

IBM AIX Enhancements and Modernization

Navdeep Dhaliwal

Ahmed Mashhour

Armin Röhl

Liviu Rosca



Power Systems



IBM Redbooks

IBM AIX Enhancements and Modernization

January 2020

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

First Edition (January 2020)

This edition applies to AIX Version 7.2 Standard Edition (product number 5765-G98), AIX Version 7.2 Enterprise Edition (product number 5765-CD3), IBM PowerVM Version 3.1 Enterprise Edition (product number 5765-VE3), IBM PowerVC Version 1.4.3 Standard Edition (product number 5765-VCS), and IBM servers that are based on POWER9 processor-based technology.

© Copyright International Business Machines Corporation 2020. All rights reserved.

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

Contents

| | |
|---|-------|
| 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 | xviii |
| Stay connected to IBM Redbooks | xix |
| Chapter 1. General enhancements | 1 |
| 1.1 Live Update function | 2 |
| 1.1.1 Live Update concepts and procedure | 2 |
| 1.1.2 Live Update modes | 4 |
| 1.1.3 Live Update management types | 4 |
| 1.1.4 Live Update methods | 4 |
| 1.1.5 AIX Live Update integration with Power Enterprise Pools | 7 |
| 1.1.6 Live Update CPU resource reduction | 11 |
| 1.1.7 Live Update across frames | 15 |
| 1.1.8 Automount File System support with Live Update | 19 |
| 1.1.9 Kerberos authentication support in Live Update | 19 |
| 1.2 Server Flash Caching | 20 |
| 1.2.1 Flash caching concepts | 20 |
| 1.2.2 Implementation modes | 21 |
| 1.3 Multipath I/O | 23 |
| 1.3.1 AIX Path Control Module | 23 |
| 1.3.2 Subsystem Device Path Control Module | 29 |
| 1.4 iSCSI software initiator | 30 |
| 1.4.1 iSCSI overview | 30 |
| 1.4.2 Configuring the initiator | 30 |
| 1.5 Network Installation Manager | 34 |
| 1.5.1 Object classes | 34 |
| 1.5.2 HTTP service | 35 |
| 1.5.3 Live Update | 36 |
| 1.5.4 The nimadm support for MultiBOS environments | 42 |
| 1.6 Logical Volume Manager | 42 |
| 1.6.1 LVM mirroring to IBM FlashSystem for enhanced performance | 42 |
| 1.6.2 LVM reclamation support | 44 |
| 1.7 JFS2 | 45 |
| 1.7.1 JFS2 defragger | 45 |
| 1.7.2 Reclaiming JFS2 space | 46 |
| 1.8 Multiple alternative disk clones | 47 |
| 1.8.1 Cloning concepts | 47 |
| 1.9 Active Memory Expansion | 53 |

| | | |
|---|---|------------|
| 1.10 | The nmon tool and current processor frequency reporting | 54 |
| 1.11 | Globalization | 57 |
| 1.11.1 | Unicode support | 57 |
| 1.11.2 | Common Locale Data Repository updates | 59 |
| 1.11.3 | International Components for Unicode for C | 60 |
| 1.12 | AIX Toolbox for Linux Applications | 60 |
| Chapter 2. Security enhancements | | 63 |
| 2.1 | AIX Trusted Execution | 64 |
| 2.2 | AIX Secure boot | 68 |
| 2.2.1 | PowerVM Secure Boot | 68 |
| 2.2.2 | AIX Secure boot implementation | 69 |
| 2.2.3 | AIX Secure boot policies and controls | 70 |
| 2.3 | AIX trusted installation and update | 71 |
| 2.3.1 | Digital signature and package signing | 72 |
| 2.3.2 | AIX package signing and digital signature catalog definition | 73 |
| 2.3.3 | AIX digital signature catalog signing process | 78 |
| 2.3.4 | Signature validation during AIX installation and update process | 79 |
| 2.3.5 | AIX trusted installation and update controls | 79 |
| 2.4 | Multifactor authentication | 83 |
| 2.4.1 | Authentication factors | 83 |
| 2.4.2 | Authentication methods | 84 |
| 2.4.3 | In-band MFA | 84 |
| 2.4.4 | Out-of-band MFA | 85 |
| 2.4.5 | Authentication on AIX systems by using RSA SecureID | 85 |
| 2.5 | Cryptographic libraries | 87 |
| 2.5.1 | OpenSSL | 87 |
| 2.5.2 | CryptoLite for C library | 87 |
| 2.6 | Address space layout randomization | 88 |
| 2.6.1 | Process address space randomized entities in AIX | 89 |
| 2.6.2 | ASLR tuning and control | 91 |
| 2.7 | Trusted shared library area support | 94 |
| Chapter 3. Networking enhancements | | 97 |
| 3.1 | Redundant link aggregation network interface backup | 98 |
| 3.2 | Shared memory communication over Remote Direct Memory Access | 99 |
| Chapter 4. Virtualization and cloud capabilities | | 101 |
| 4.1 | AIX on public cloud | 102 |
| 4.2 | IBM Power Virtualization Center | 102 |
| 4.2.1 | Introduction | 102 |
| 4.2.2 | IBM PowerVC in the virtualization and cloud software stack | 104 |
| 4.2.3 | IBM PowerVC features | 105 |
| 4.3 | IBM Cloud Automation Manager | 106 |
| 4.3.1 | Terraform | 106 |
| 4.3.2 | Configuring a cloud connection | 107 |
| 4.3.3 | Creating a template for AIX deployment | 111 |
| 4.3.4 | Deploying an AIX partition by using a template | 119 |
| 4.4 | Ansible automation and AIX | 122 |
| 4.4.1 | Installing Ansible on an AIX control node | 123 |
| 4.4.2 | AIX specific Ansible modules | 124 |
| 4.5 | Chef Infra client on AIX | 125 |
| 4.6 | Puppet Enterprise on AIX | 125 |

| | |
|---|-----|
| Chapter 5. IBM AIX and IBM PowerVM features | 127 |
| 5.1 Storage access | 128 |
| 5.2 Network access | 129 |
| 5.2.1 Dedicated adapters | 129 |
| 5.2.2 Virtual Ethernet adapters | 129 |
| 5.2.3 VIOS Shared Ethernet Adapter | 129 |
| 5.2.4 Virtual Network Interface Cards | 129 |
| 5.3 Dynamic LPAR support | 130 |
| 5.4 Virtual processors | 130 |
| 5.5 Simultaneous multi-threading and logical processors | 130 |
| 5.6 Active System Optimizer and Dynamic System Optimizer | 130 |
| 5.7 Shared storage pools | 131 |
| 5.7.1 SSP consideration and procedures | 132 |
| 5.7.2 SSP setup procedure | 133 |
| 5.8 PowerVM NovaLink | 135 |
| 5.8.1 Components of PowerVM NovaLink | 135 |
| 5.8.2 Software-defined infrastructure capabilities | 136 |
| 5.8.3 Resource Monitoring Control communication | 136 |
| 5.8.4 PowerVM NovaLink and Hardware Management Consoles | 137 |
| 5.9 Power Enterprise Pools | 137 |
| 5.9.1 Power Enterprise Pools first edition | 138 |
| 5.9.2 Power Enterprise Pools second edition | 140 |
| 5.9.3 Comparison between PEP first and second edition | 142 |
| 5.10 Linux on Power | 143 |
| 5.11 Virtual I/O Server enhancements | 143 |
| 5.11.1 Key features | 143 |
| 5.11.2 N_Port ID Virtualization | 144 |
| 5.11.3 iSCSI support | 144 |
| 5.11.4 Upgrading Virtual I/O Server to Version 3.1 | 145 |
| | |
| Chapter 6. Disaster recovery and high availability | 149 |
| 6.1 IBM VM Recovery Manager for Power Systems | 150 |
| 6.1.1 IBM VM Recovery Manager versions and lifecycle | 150 |
| 6.1.2 IBM VM Recovery Manager HA components | 151 |
| 6.1.3 IBM VM Recovery Manager DR components | 152 |
| 6.2 PowerHA | 155 |
| 6.2.1 PowerHA editions | 155 |
| 6.2.2 PowerHA versions and lifecycle | 155 |
| 6.2.3 AIX requirements for various PowerHA levels | 156 |
| 6.2.4 PowerHA licensing | 156 |
| | |
| Chapter 7. IBM AIX fundamentals | 159 |
| 7.1 Logical Volume Manager | 160 |
| 7.1.1 Introduction to the LVM | 160 |
| 7.1.2 LVM components | 161 |
| 7.1.3 Principles to optimize LVM disks | 163 |
| 7.1.4 LVM strategies of various storage types | 164 |
| 7.1.5 LVM configuration | 165 |
| 7.2 AIX Enhanced Journaled File System | 171 |
| 7.2.1 Enhanced Journaled File System functions | 171 |
| 7.2.2 JFS2 features | 172 |
| 7.3 Role-based access control | 173 |
| 7.3.1 RBAC elements | 174 |
| 7.3.2 RBAC Kernel Security Table and exact files | 175 |

| | |
|---|------------|
| 7.3.3 Customizing an RBAC role for certain tasks | 176 |
| 7.3.4 Domain RBAC | 179 |
| 7.3.5 Domain RBAC implementation scenario | 179 |
| 7.4 Encrypted File System | 182 |
| 7.4.1 EFS commands | 183 |
| 7.4.2 Sample scenario of EFS | 184 |
| 7.4.3 Integrating an EFS keystore with OpenSSH key authentication | 187 |
| 7.5 AIX Security Expert and IBM PowerSC integration | 190 |
| 7.5.1 Using AIXpert | 191 |
| 7.5.2 Using AIXpert to generate a compliance report by running the IBM PowerSC pscxpert command | 192 |
| 7.6 The AIX Auditing subsystem and Autonomic Health Advisor File System | 194 |
| 7.6.1 The AIX Auditing subsystem | 194 |
| 7.6.2 Implementing the AIX Auditing subsystem for exact events | 195 |
| 7.6.3 Autonomic Health Advisor File System | 197 |
| 7.7 MultiBOS | 199 |
| 7.7.1 Standby BOS setup | 200 |
| Related publications | 207 |
| IBM Redbooks | 207 |
| Online resources | 207 |
| Help from IBM | 210 |

Figures

| | | |
|------|---|-----|
| 1-1 | Flow diagram for PEP resource acquisition and allocation during AIX Live Update. . . . | 9 |
| 1-2 | The <code>cpu_reduction</code> parameter help text in the <code>lvupdate.template</code> file | 11 |
| 1-3 | The destination and <code>force_migration</code> parameters help in the <code>lvupdate.template</code> file. . . | 16 |
| 1-4 | The <code>nmon</code> Resources screen showing the CPU-Current-Speed metric | 56 |
| 2-1 | PowerVM Secure Boot and AIX Secure boot flow | 69 |
| 2-2 | ODM object class definition of <code>dsc_inventory</code> | 76 |
| 2-3 | Example DSC inventory stanza for the <code>bos</code> package and the <code>bos.rte</code> LPP. | 76 |
| 2-4 | ODM object class definition of <code>dsc_key</code> | 77 |
| 2-5 | ODM object class definition of <code>dsc_keystore</code> | 78 |
| 2-6 | Help message for the <code>chsignpolicy</code> command | 80 |
| 2-7 | New sample <code>SIGN_POLICY</code> control flow stanza in the <code>bosinst.template</code> file | 81 |
| 2-8 | The <code>/usr/lpp/bosinst/bosinst.README</code> information about the <code>SIGN_POLICY</code> stanza usage | 81 |
| 2-9 | Main Installation and Settings menu | 82 |
| 2-10 | Security Models submenu to configure the digital signature policy. | 82 |
| 2-11 | Help message for the <code>aslr vmo</code> tunable | 92 |
| 2-12 | Help message for the <code>aslr32 vmo</code> tunable | 93 |
| 2-13 | Help message for <code>aslr64 vmo</code> tunable | 94 |
| 2-14 | Help message for the <code>useTrustedSLAs vmo</code> tunable | 95 |
| 4-1 | IBM PowerVC integration into HMC-managed Power Systems environments. | 104 |
| 4-2 | IBM PowerVC integration into NovaLink -managed Power Systems environments. . . | 105 |
| 4-3 | Creating a cloud connection | 108 |
| 4-4 | Selecting your cloud provider for a cloud connection | 109 |
| 4-5 | Entering the configuration details for the OpenStack connection | 110 |
| 4-6 | Saving your cloud connection. | 110 |
| 4-7 | Creating a template from scratch in Cloud Automation Manager | 111 |
| 4-8 | Entering template metadata | 112 |
| 4-9 | Opening the inline editor page | 113 |
| 4-10 | Adding your Terraform template | 114 |
| 4-11 | Parameter window | 117 |
| 4-12 | Generating parameters for your template | 118 |
| 4-13 | Selecting a template to deploy | 119 |
| 4-14 | Entering the basic details for instance deployment. | 119 |
| 4-15 | Entering more details for instance deployment. | 121 |
| 4-16 | Successful instance deployment | 122 |
| 5-1 | iSCSI data flow | 145 |

Tables

| | | |
|-----|--|-----|
| 1-1 | MPIO path_status definitions | 26 |
| 1-2 | List of object classes in NIM | 34 |
| 1-3 | Management object types and description | 34 |
| 1-4 | Unicode release support and history for AIX 7.2 and related AIX 7.1 releases | 57 |
| 1-5 | CLDR release support and history for AIX 7.2 and related AIX 7.1 releases | 60 |
| 1-6 | ICU4C release support and history for AIX 7.2 and related AIX 7.1 releases | 60 |
| 2-1 | Trusted Execution different policies | 65 |
| 2-2 | New ld command options support for ASLR | 92 |
| 2-3 | New lldedit command options support for ASLR | 93 |
| 3-1 | Mode and hash mode combinations and the outgoing traffic distributions each one produces | 98 |
| 4-1 | Sample IBM PowerVC details | 107 |
| 4-2 | Cloud Automation Manager cloud connection for IBM PowerVC connectivity | 108 |
| 4-3 | AIX specific Ansible modules that are available since Ansible V2.8 | 124 |
| 5-1 | PowerVM NovaLink and Hardware Management Console capabilities | 137 |
| 5-2 | Comparison between PEP editions | 142 |
| 6-1 | Requirements for VMR HA components | 152 |
| 6-2 | Requirements for the VMR DR components | 154 |
| 7-1 | RBAC Kernel Security Tables | 175 |
| 7-2 | RBAC definition files | 175 |
| 7-3 | Listing the AIXpert security levels | 192 |
| 7-4 | Available predefined event producers | 197 |

Examples

| | | |
|------|---|----|
| 1-1 | Checking the AIX environment | 5 |
| 1-2 | Authentication between the AIX server and HMC. | 6 |
| 1-3 | Authentication between the AIX server and IBM PowerVC | 6 |
| 1-4 | Editing /var/adm/ras/liveupdate/lvupdate.data | 6 |
| 1-5 | Running geninstall -k to perform Live Update. | 7 |
| 1-6 | CPU_Reduc stanza in the liveupdate.cf file | 13 |
| 1-7 | CPU_Usage stanza in the liveupdate.cf file | 14 |
| 1-8 | Creating a cache pool and cache partition in a dedicated flash logical unit number. | 21 |
| 1-9 | Assigning a cache partition to a target device that must be cached. | 21 |
| 1-10 | Starting caching on the hdisk2 target and monitoring the status | 21 |
| 1-11 | Creating a cache pool and cache partition in a VIOS | 22 |
| 1-12 | Assigning the cache partition to a vhost in a VIOS | 22 |
| 1-13 | Assigning a cache partition to a target hdisk2 device in the AIX LPAR | 22 |
| 1-14 | Starting caching on the hdisk2 target and monitoring the status | 22 |
| 1-15 | Creating a cache pool and cache partition in an NPIV flash logical unit number | 23 |
| 1-16 | Assigning a cache partition to a target device that must be cached. | 23 |
| 1-17 | Starting caching on a hdisk2 target and monitoring the status | 23 |
| 1-18 | Using lsmpio to check the path status | 25 |
| 1-19 | Displaying parent adapters and remote ports | 26 |
| 1-20 | Sample output of MPIO detailed statistics | 27 |
| 1-21 | Dynamic queue_depth update for device | 28 |
| 1-22 | Changing the default reserve_policy to no_reserve for AIXPCM FC disk. | 28 |
| 1-23 | Changing the default algorithm to shortest_queue for the AIXPCM iSCSI disk | 29 |
| 1-24 | Disabling paths by using chpath | 29 |
| 1-25 | File-based iSCSI discovery. | 32 |
| 1-26 | ODM-based iSCSI discovery | 32 |
| 1-27 | Add an iSCSI Target Protocol Device smitty menu. | 33 |
| 1-28 | Enabling the nimhttp subsystem | 35 |
| 1-29 | Using nimhttp without using cryptographic authentication | 36 |
| 1-30 | Creating an HMC resource | 37 |
| 1-31 | Using nimquery to define the CEC object | 38 |
| 1-32 | Manually defining a CEC | 39 |
| 1-33 | Registering the NIM client by using niminit on the client partition. | 39 |
| 1-34 | Adding a management source and LPAR ID to an AIX machine | 39 |
| 1-35 | Creating a Live Update data resource | 40 |
| 1-36 | Live Update initiated from the NIM master | 41 |
| 1-37 | The defragfs command. | 45 |
| 1-38 | Checking the file system space reclamation support | 47 |
| 1-39 | Checking the system physical volumes | 48 |
| 1-40 | Initiating the cloning process to hdisk1 | 48 |
| 1-41 | Checking the new volume group that is created after the clone | 49 |
| 1-42 | Waking up the clone and putting it to sleep | 49 |
| 1-43 | Changing the root directory to access altinst_rootvg clone | 50 |
| 1-44 | Putting altinst_rootvg to sleep. | 51 |
| 1-45 | Attempting to create another clone from the mkysyb image and receiving an error. | 51 |
| 1-46 | Renaming the first altinst_rootvg clone. | 52 |
| 1-47 | Creating a second clone and renaming it | 52 |
| 1-48 | The nmon command to collect processor frequency metrics | 56 |

| | |
|---|-----|
| 1-49 Processor frequency-related statistics that are collected by the -y dfreq=on nmon argument | 56 |
| 1-50 Using yum.sh to install and configure AIX Toolbox repositories | 61 |
| 2-1 Checking the CLiC and kernel extension availability in the system | 66 |
| 2-2 Checking the TE status and enabling it | 66 |
| 2-3 Permitting the ls command according to the TSD database | 66 |
| 2-4 New untrusted ls command not running | 67 |
| 2-5 Disabling the TE integrity check | 67 |
| 2-6 Enforcing AIX Secure boot by using the chsyscfg HMC command | 71 |
| 2-7 The secure_boot attribute of the sys0 kernel device | 71 |
| 4-1 Sample provider.tf for Terraform deployment to IBM PowerVC | 106 |
| 4-2 Terraform template to create an IBM PowerVC partition | 115 |
| 4-3 Installing Ansible by using YUM on AIX | 123 |
| 5-1 Creating an SSP cluster | 134 |
| 5-2 Creating a logical unit that will be used for the client logical partition | 134 |
| 5-3 Mapping the logical unit to a virtual host with a VTD name | 134 |
| 5-4 Adding a second VIOS node | 134 |
| 5-5 Increasing dynamically the logical unit size | 134 |
| 5-6 Extracting mksysb from VIOS V3.1 Flash Media | 146 |
| 5-7 Extracting mksysb from the VIOS V3.1 DVD installation media | 146 |
| 7-1 Extending the volume group by adding a disk for mirroring | 165 |
| 7-2 Using the mirrorvg SMIT menu | 165 |
| 7-3 Setting the preferred read copy by using mklv or changing it by using chlv | 166 |
| 7-4 Checking the free PPs | 167 |
| 7-5 Using the mklv command to create raw LVs | 167 |
| 7-6 Unmounting a file system by using umount | 168 |
| 7-7 Unmounting a file system by using SMIT | 168 |
| 7-8 Deactivating the volume group by using the varyoffvg command | 168 |
| 7-9 Deleting the volume group by running the exportvg command | 168 |
| 7-10 Deleting the disk by running the rmdev command | 169 |
| 7-11 Listing the hdisk2 information by using lspv hdisk2 | 169 |
| 7-12 Listing the volume group information | 169 |
| 7-13 Exporting the volume group | 170 |
| 7-14 Removing the disk by using the rmdev command | 170 |
| 7-15 Checking disk growth | 170 |
| 7-16 Turning off the bad block relocation policy | 170 |
| 7-17 Checking the mklv command full path and whether it has security attributes | 176 |
| 7-18 Checking whether any role is associated to the access authorization | 176 |
| 7-19 Creating a role by running the mkrole command | 176 |
| 7-20 Running the setkst command to update the kernel security tables | 176 |
| 7-21 Running the chuser command to add the new role | 177 |
| 7-22 Listing roles that are assigned to the current logged in user | 177 |
| 7-23 Checking whether enhanced_RBAC is enabled | 177 |
| 7-24 Enabling enhanced_RBAC and restarting | 177 |
| 7-25 Running the mkauth command to create an authorization | 177 |
| 7-26 Viewing the authorization | 178 |
| 7-27 Running tracepriv for a certain command | 178 |
| 7-28 Running the setsecattr command to bind security attributes to the authorization | 178 |
| 7-29 Viewing the complete authorization that is associated with the command | 178 |
| 7-30 Running the mkrole command | 178 |
| 7-31 Adding the new role to the user roles | 178 |
| 7-32 Updating the kernel security table by running the setkst command | 179 |
| 7-33 Checking the enhanced_RBAC status | 180 |

| | | |
|------|--|-----|
| 7-34 | Creating a user for the job of mounting the file systems | 180 |
| 7-35 | Checking the authorizations that are associated with the mount command | 180 |
| 7-36 | Checking the authorizations that are associated with the umount command | 180 |
| 7-37 | Creating a role with both authorizations and listing the role | 180 |
| 7-38 | Assigning the newly created role to the designated user | 180 |
| 7-39 | Updating the Kernel Security Table | 181 |
| 7-40 | Testing the role of the user | 181 |
| 7-41 | Creating RBAC domains | 181 |
| 7-42 | Checking the file system | 181 |
| 7-43 | Assigning a domain to a file system | 181 |
| 7-44 | Assigning only the dom1 domain to the test user | 182 |
| 7-45 | Updating the Kernel Security Table | 182 |
| 7-46 | Attempting to mount the file systems | 182 |
| 7-47 | Enabling EFS by running efsenable | 184 |
| 7-48 | Checking the EFS directories | 184 |
| 7-49 | Creating EFSs | 184 |
| 7-50 | Creating the users example | 185 |
| 7-51 | Creating the EFS directory and setting the inheritance | 185 |
| 7-52 | Checking the EFS enablement | 185 |
| 7-53 | Loading the EFS keystore into the shell | 186 |
| 7-54 | Listing the encrypted files | 186 |
| 7-55 | Listing the encrypted file attributes | 186 |
| 7-56 | Encrypting individual file | 186 |
| 7-57 | Enabling public key authentication | 187 |
| 7-58 | Enabling EFS login for the client and server | 187 |
| 7-59 | Restarting sshd on both the client and server machines | 187 |
| 7-60 | Creating a client user and generating a public key | 188 |
| 7-61 | Copying public keys on to the server into authorized_keys | 188 |
| 7-62 | Creating a keystore and insert the public key cookie | 189 |
| 7-63 | Verifying the login authentication | 190 |
| 7-64 | Setting a specific AIXpert security level | 191 |
| 7-65 | Implementing AIXpert low-level security in AIX | 192 |
| 7-66 | Applying low-level security with verbose mode | 192 |
| 7-67 | Checking the rule in the XML file and in /etc/security/user | 193 |
| 7-68 | Checking the security settings against the previously applied set of rules | 193 |
| 7-69 | Generating a compliance report by running pcsxpert | 193 |
| 7-70 | Checking the .txt version compliance report | 193 |
| 7-71 | Changing the audit mode in the /etc/security/audit/config file | 195 |
| 7-72 | Defining a class that has the event name in the audit config file | 195 |
| 7-73 | Adding the class to a specific user | 195 |
| 7-74 | Adding the auditstream command for redirection to an output file | 196 |
| 7-75 | Defining an audit object to monitor its execution | 196 |
| 7-76 | Defining an event format | 196 |
| 7-77 | Recycling the audit subsystem | 196 |
| 7-78 | Stopping the audit and monitoring the events | 197 |
| 7-79 | Mounting the /aha file system | 198 |
| 7-80 | Editing the aha-pl.inp file to monitor /tmp file system | 198 |
| 7-81 | Mounting the aha file system and start monitoring | 199 |
| 7-82 | Previewing the standby BOS instance creation | 201 |
| 7-83 | Creating the MultiBOS instance | 202 |
| 7-84 | Checking the standby logical volumes | 203 |
| 7-85 | Setting the boot list to point to the standby instance first | 203 |
| 7-86 | Restarting the system and checking which instance was used to start | 203 |

| | | |
|------|---|-----|
| 7-87 | Updating the current active and standby boot images | 204 |
| 7-88 | Checking the MultiBOS unique tagging | 204 |
| 7-89 | Removing the MultiBOS instance | 205 |

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.

| | | |
|-----------------------------|---------------------|---|
| AIX® | Micro-Partitioning® | Redbooks (logo)  ® |
| Db2® | POWER® | Storwize® |
| DB2® | POWER6™ | System Storage™ |
| DS8000® | POWER7® | SystemMirror® |
| Global Technology Services® | POWER8® | Terraform® |
| HyperSwap® | POWER9™ | Tivoli® |
| IBM® | PowerHA® | WebSphere® |
| IBM Cloud™ | PowerPC® | XIV® |
| IBM FlashSystem® | PowerVM® | z/OS® |
| IBM Spectrum® | Redbooks® | |

The following terms are trademarks of other companies:

Connect:Direct, are trademarks or registered trademarks of IBM International Group B.V., an IBM Company.

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

- ▶ General AIX enhancements
- ▶ AIX Live Update individually or by using Network Installation Manager (NIM)
- ▶ AIX security features and integration
- ▶ AIX networking enhancements
- ▶ IBM PowerVC integration and features for cloud environments
- ▶ AIX deployment that uses IBM Terraform® and IBM Cloud Automation Manager
- ▶ AIX automation that uses configuration management tools
- ▶ AIX integration with Kubernetes
- ▶ PowerVM enhancements and features
- ▶ Latest disaster recovery (DR) solutions
- ▶ AIX Logical Volume Manager (LVM) and Enhanced Journaled File System (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 Bachelor of Computer Science degree. His areas of expertise include IBM PowerVM, IBM Power Systems, AIX, Linux, Chef, Ansible, and Cloud technologies.

Ahmed Mashhour is a Power Systems Global subject matter expert (SME) at IBM Egypt. 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. He has authored several publications inside and outside IBM, including co-authoring other IBM Redbooks® publications. He has hosted IBM AIX, PowerVM, PowerHA®, and IBM Spectrum® Scale classes more than 70 times around the world.

Armin Röhl works as a Power Systems IT specialist in Germany. He has 24 years of experience in Power Systems and AIX pre-sales technical support. He holds a degree in experimental physics from the University of Hamburg, Germany. He co-authored the IBM AIX Version 4.3.3, the IBM AIX 5L Version 5.0, the IBM AIX 5L Version 5.3, the IBM AIX Version 6.1, and the IBM AIX 7.1 Differences Guide IBM Redbooks publications, and the IBM Power System E950 and the IBM Power System E980 IBM Redpapers.

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, AIX, and 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.

The project that produced this publication was managed by:

Scott Vetter, PMP

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

Amit Agarwal, Javier Bazán, Damien Bergamini, Petra Buehrer, Ben Castillo, Rahul Chandrakar, Jim Cunningham, Bert Dufranse, Paul B Finley, Nigel Griffiths, Srinivas Gundurao, James L Hall, Axel Ismirlian, Joefon Jann, Sivakumar Krishnasamy, Su Liu, Aurelien Lun-Sin, Ann Lund, Christina Morel, Michael Passaloukos, Amartey Pearson, Stephen Peckham, Gilles Quillard, Xiaohan Qin, 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:
<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>
- ▶ 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.

This chapter describes the following topics:

- ▶ Live Update function
- ▶ Server Flash Caching
- ▶ Multipath I/O
- ▶ iSCSI software initiator
- ▶ Network Installation Manager
- ▶ Logical Volume Manager
- ▶ JFS2
- ▶ Multiple alternative disk clones
- ▶ Active Memory Expansion
- ▶ The nmon tool and current processor frequency reporting
- ▶ Globalization
- ▶ AIX Toolbox for Linux Applications

1.1 Live Update function

The AIX operating system (OS) has the Live Update function, which eliminates the workload downtime that is associated with an AIX system restart that is required by AIX releases before Version 7.2 when fixes to the AIX kernel are deployed.

AIX Live Update enables the following functions:

- ▶ The workloads on the system continue running in a Live Update operation. The workloads can use the interim fixes after the Live Update operation.
- ▶ AIX 7.2 Live Update aims to achieve zero downtime while updating OS patches, service packs, and technology levels (TLs) without disrupting business-critical workloads.
- ▶ Live Update can save organizations on substantial costs and help them avoid a data breach while applying critical security patches, which might happen due to not completing a maintenance window on time.

1.1.1 Live Update concepts and procedure

In this section, we explain the Live Update concepts and modes.

The following keywords that are used for Live Update are defined as follows:

| | |
|----------------------------------|---|
| Original partition | The logical partition (LPAR) where the operation is started. |
| Surrogate partition | Another LPAR that is used by Live Update. |
| Checkpointing a workload | Freezing a running process and saving its current state. |
| Live Application Mobility | Checkpointing processes on an LPAR and restarting them later on another LPAR. |

Tip: If you plan to install updates by using the Live Update function, before you begin the installation, it is a best practice 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 alternative disk copy. The updates that are installed by using the Live Update function are always committed, so you cannot reject the updates later.

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

1. If a service pack (SP) or a technology level (TL) update is the only intended scope of a Live Update operation, the updates are applied and committed first on the original partition. If any interim fixes are specified along with the SP or TL update, the interim fixes are also installed and committed on the original partition as part of the first process step.
2. The root volume group of the original partition (orig-rootvg) is cloned by an alterate disk copy operation.
3. If only interim fix installation is within the scope of the Live Update operation, the interim fixes are applied on the cloned volume group. This step is not needed if the interim fix installation is part of a SP or TL update because the interim fixes installation would have than been covered in the first process step before the orig-rootvg was cloned.
4. Further customization is performed on the cloned volume group with the intend to export the volume group on the original partition and to use it as boot root volume group (surr-boot-rootvg) for the surrogate partition which is created in the next step.

5. The provisioning of a surrogate partition is initiated by the original partition and facilitated by the managing HMC of the environment.
6. After the surrogate partition is started and while the workloads are still running on the original partition, an additional copy of the original's root volume group is created. The creation of a new mirror copy for the orig-rootvg is initiated by the original partition shortly before the checkpoint / mobility phase is entered.
7. All relevant workload processes are checkpointed by the Metacluster Checkpoint and Restart (MCR) technology.
8. After the checkpoint / mobility phase has been entered, and consequently no workload is making any changes to the rootvg on the original partition (orig-rootvg), the mirror copies are synchronized and one mirror is split off. One copy is retained on the original partition whereas the other copy of the orig-rootvg is transferred to the surrogate partition. On the surrogate the mirror copy of the orig-rootvg is imported and its file systems are used to back a chroot environment. If only interim fixes and no SP or TL update are within the scope of the LU operation the ifix updates have to be applied to the mirror copy of the orig-rootvg disks during the checkpoint blackout period. This process is initiated by the surrogate and the mirror copy of the orig-rootvg is referred to as surr-mir-rootvg as soon as the volume group has been imported (and optionally updated) on the surrogate.
9. The checkpointed workload processes are moved to the surrogate partition during the mobility phase.
10. Workloads resume on the surrogate partition in a chroot environment on the surr-mir-rootvg (the updated copy of the orig-rootvg). During this process, the workloads continue to run without being stopped, although a short blackout time occurs when these workloads are suspended.
11. If the Live Update operation fails after step 1 on page 2, the updates and interim fixes that are installed on the system in these steps are not uninstalled. If the cause of the Live Update failure is corrected, you can attempt the Live Update operation again instead of restarting the original LPAR. In this scenario, updates or interim fixes are not specified for the Live Update operation because the updates are already installed.

Live Update has the following characteristics:

- ▶ The Live Update feature is intended for applying interim fixes that contain kernel changes or kernel extension changes.
- ▶ Service pack and TL installations that require a restart.
- ▶ The package type that is installed by Live Update might contain other files (for example, commands and libraries), and the Live Update feature does not change anything about the way these files are applied. For example, a shared library is modified on the file system, but any running processes continue 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 7.2 with the 7200-01 TL 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 VG (rootvg) is retained after the Live Update operation is complete. Thus, if you have installed only interim fixes with the Live Update function and 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 VG (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 OS. After these updates or fixes

are installed, you can use the Live Update function to load the updated kernel software without restarting the system.

1.1.2 Live Update modes

Live Updates has two modes: preview mode or automated mode.

Preview mode

In preview mode, the estimation of the total operation time, application blackout time, and 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 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 Update operation completes.

1.1.3 Live Update management types

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

HMC-based Live Update operation

If the LPAR is managed by an HMC, you must authenticate to the HMC by using the `hmcauth` command or by defining an HMC object through NIM.

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

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

IBM PowerVC based Live Update operation

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

When you run the Live 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.data` file. If multiple profiles are associated with the LPAR, only the active profile is maintained by the Live Update operation. The other profiles are not preserved after the Live Update operation is complete. The virtual adapter ID values, also known as slot numbers, might change during the Live Update operation.

1.1.4 Live Update methods

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

Live Update by using NIM

Performing the Live Update operation by using NIM requires preparation. You can start the AIX Live Update operation on the target machine by using the NIM master or a NIM client.

For more information, see 1.5, “Network Installation Manager” on page 34.

Live Update by running the `geninstall` command

You can always run Live Update by running the `geninstall` command, either by pointing to the TL, service pack, or the interim fix, or by updating the AIX and then verifying that Live Update does not make you restart the AIX server.

To run Live Update by running the `geninstall` command, complete the following steps:

1. Check your current AIX server and confirm that you have at least two free disks with a proper multipathing setup, as shown in Example 1-1. As a best practice, have three disks to run `alt_disk_copy`.

Example 1-1 Checking the AIX environment

```
# 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_sndspace 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 fscsi0
Enabled hdisk1 fscsi0
Enabled hdisk2 fscsi0
Enabled hdisk0 fscsi1
Enabled hdisk1 fscsi1
Enabled hdisk2 fscsi1
Enabled hdisk0 fscsi1
Enabled hdisk1 fscsi1
Enabled hdisk2 fscsi1
```

As a best practice, use the `alt_disk_copy` command to create a clone beforehand. For more information about creating the rootvg clone, see 1.8, “Multiple alternative disk clones” on page 47.

2. Authenticate between the AIX server and the controlling environment, which is either HMC or IBM PowerVC, as shown in Example 1-2 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 IBM 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
```

3. After confirming the authentication, make sure that you create the Live Update data file from the `lvupdate` template file, which is at the following path:

```
/var/adm/ras/liveupdate/lvupdate.template
```

Copy it to the same location with the following file name:

```
/var/adm/ras/liveupdate/lvupdate.data
```

As shown in Example 1-4, we set `nhdisk` to `hdisk2`, which is the disk that we use to create a copy of the original rootvg and start the surrogate partition, and we set `mhdisk` to `hdisk3`, which is the disk we use for the mirrored rootvg on the surrogate partition. The HMC `lpar_id` is the new surrogate LPAR ID, but it can be left blank, which lets the system select the next available ID for the new surrogate partition and the HMC or IBM PowerVC authentication that was validated in Example 1-3.

Example 1-4 shows editing the `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
```

4. Run Live Update either for a TL, service pack, or an interim fix by running `geninstall -k`, as shown in Example 1-5.

Example 1-5 Running geninstall -k to perform Live Update

```
# oslevel -s  
7200-03-01-1838  
# geninstall -Yk -d /patches/aix/aix72t13sp3 update_all
```

Example 1-5 shows an update for the AIX servers 7200-02-01 - 7200-03-03 without restarting the AIX server.

If you already updated the AIX server with either a TL, service pack, or interim fix by running either **smitty update_all** or **emgr** and you have not restarted, you can activate the updates without restarting by running **geninstall -k**.

Note: The **geninstall -k** command does not update RPM packages. RPM packages should be updated before performing a Live Update operation.

1.1.5 AIX Live Update integration with Power Enterprise Pools

Performing a Live Update operation requires enough available CPU and memory resources to clone and start the surrogate LPAR on the central electronics complex (CEC) while the original LPAR is running. In the initial implementation of AIX Live Update up to AIX 7.2 TL 1, the Live Update operation fails if there are insufficient resources in the pool of free processor or memory resources (free pool) of the system.

AIX 7.2 TL 2 adds Power Enterprise Pool (PEP) Capacity on Demand (CoD) (PEP CoD) technology with AIX Live Update. This enhancement uses Mobile CoD resources from the classical PEP technology during the Live Update operation when the required resources are insufficient on the CEC where the AIX LU operation is initiated. The PEP feature is available only in an HMC-managed environment and is not under the control of IBM PowerVC.

A *PEP* is a group of systems that can share Mobile CoD processor and memory resources. Mobile CoD resource activations can be moved among the systems in a pool with HMC commands. These operations provide flexibility in managing large workloads in a pool of systems and helps to rebalance the resources to respond to business needs. This feature is useful for providing continuous application availability during maintenance. The workloads can be easily moved to alternative systems, and so can the processor activations and memory activations. Disaster recovery (DR) planning also is more manageable with the ability to move activations where and when they are needed.

There are three different types of PEP that are available:

- ▶ PEP for IBM Power System 770 (9117-MMD), IBM Power System E870 (9119-MME), IBM Power System E870C (9080-MME), and IBM Power System E880C (9080-MHE)
- ▶ PEP for IBM Power System 780 (9179-MHD), IBM Power System 795 (9119-FHB), IBM Power System E880 (9119-MHE), Power E870C (9080-MME), and Power E880C (9080-MHE)
- ▶ PEP for Power E870 (9119-MME), Power E870C (9080-MME), Power E880 (9119-MHE), Power E880C (9080-MHE), and IBM Power System E980 (9080-M9S)

For more information about the PEP technology, see [Power Systems Capacity on Demand](#).

Classical PEP resources are integrated into the AIX Live Update operation and do not require any specific configuration activity. During Live Update, for each step that uses server resources, if there are no more available resources in the free pool on the server, a request is sent to the HMC through the Resources Pool Provisioning (RPP) C++ library (librpp)

component to activate mobile memory or processor resources that are available from the PEP.

In the AIX Live Update preview step, the available processor and memory resources on the server are evaluated. If they are insufficient, then a new check is initiated through the librpp component to verify whether the resources can be acquired from the PEP. The preview step does not activate the resources from the free pool, but rather verifies their availability status. If the check fails, then Live Update stops with an appropriate error message. If the check succeeds, a new message is printed to indicate that the Mobile CoD resources from the PEP will be used during the Live Update.

If possible, the surrogate LPAR is configured according to the needed processor and memory value of the original LPAR, but the implementation considers the available resources as sufficient if they are equal to or larger than the configured processor or memory minimum value of the active LPAR profile. The available values are applied to the surrogate LPAR if the resources are not allocatable.

Immediately before the HMC initiates the cloning of the original partition, the evaluation of the available processor and memory resources on the server is repeated. If they are not sufficient, a new operation of acquiring resources is started through the librpp component. If the resource activation operation fails again, more attempts are done by a specific retry mechanism. If the activate operation succeeds, then the Live Update process can continue.

The activation of the Mobile CoD resources is not a strict reservation of resources for the clone operation. It is an acquisition into the free pool that is available for all the partitions that are defined on the server. The real acquisition of the resources is run by the HMC during the clone step.

Figure 1-1 shows the flow diagram for PEP resource acquisition to the pool of free resources of a system followed by the resource allocation during the AIX Live Update clone phase.

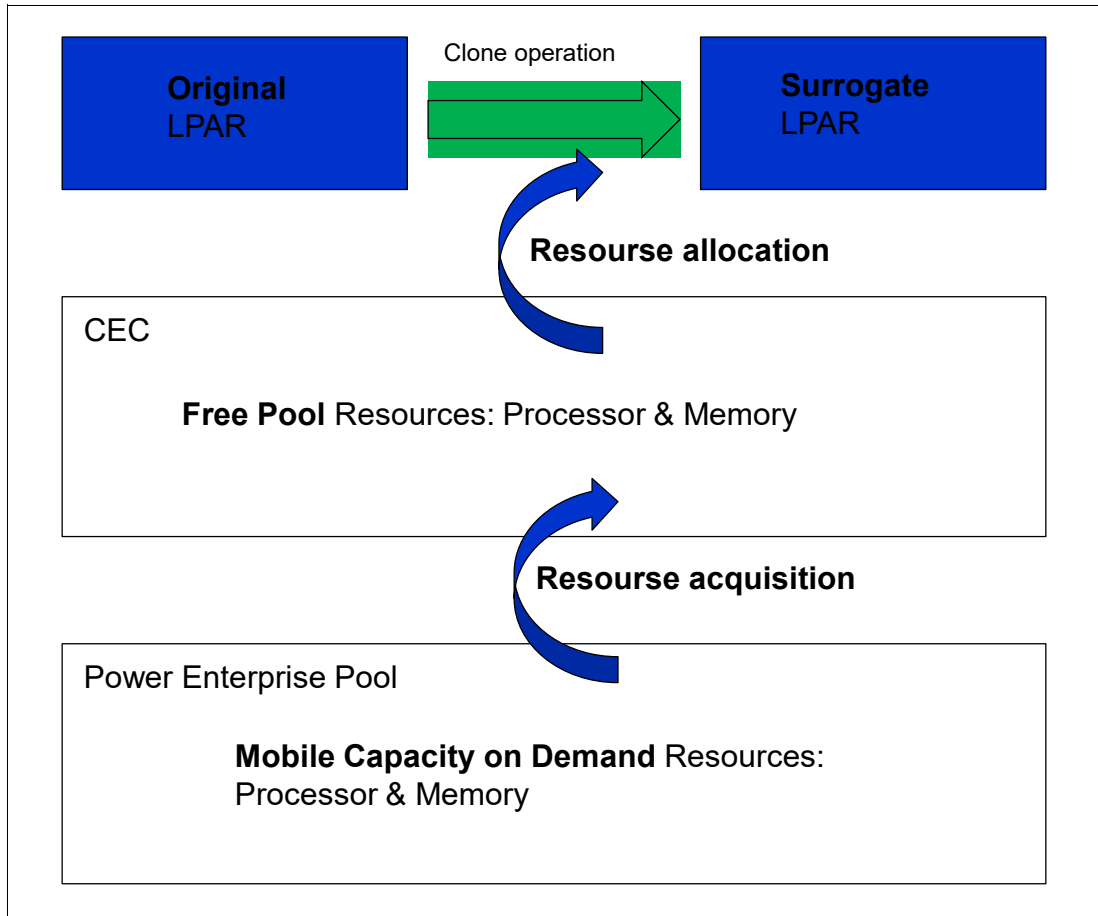


Figure 1-1 Flow diagram for PEP resource acquisition and allocation during AIX Live Update

In some cases, the Live Update operation might fail during the surrogate boot phase due to a lack of resources. In those cases, a new request to acquire resources from the PEP is done through `librpp`. Similar to the clone process step, a retry mechanism covers the situation if the resource acquisition fails again. If the acquire operation succeeds, then Live Update can continue.

The retry mechanism for PEP resource requests is a rescue mechanism to prevent the lack of resources during the clone phase of the original partition or during the boot phase of the surrogate partition. The retry mechanism accommodates situations where several operations are competing concurrently for the available resources (for example, parallel AIX Live Update operations).

The PEP resources are not used when the number of available resources on the CEC where the AIX Live Update operation is initiated is greater than the size of the LPAR to be updated with AIX Live Update.

If the available resources on the CEC are not sufficient to establish a surrogate partition, Live Update tries to fulfill the resource requirements by using the free resources of the PEP. If the free PEP resources are not sufficient, an operation to acquire PEP CoD resources from the pool members is initiated.

The operation of acquiring PEP CoD resources for temporary use during the AIX Live Update operation considers all mobile processor and memory resources that are distributed over all systems that are members of the pool. The operation examines the pool members until the resource requirements are met, which implies that not all pool members must provide resources for the initiated Live Update operation. For each member, except for the CEC on which the AIX Live Update operation runs, the acquire operation ensures the following items:

- ▶ That each CEC's mobile resources that are free or in use but not in the unreturned state are released into the pool to be available for the Live Update operation.
- ▶ That the mobile resources that are acquired and used by the Live Update operation are tagged as unreturned resources, but that the partitions may continue to use them during the grace period inherent to the PEP.
- ▶ That the amount of the mobile resources that are acquired (activated or deactivated) for each CEC are saved in a dedicated resource stanza file (`epcod.cf`) to restore the initial CEC resources state at the end of a Live Update operation.

The new stanza configuration file at `/var/adm/ras/liveupdate/epcod.cf` ensures that only the PEP CoD resources that are activated during the Live Update operation are released after the deletion of the original partition or in the cleanup phase after an encountered error. The file is updated by the original partition during the clone and the boot operations for each PEP acquisition and is transferred to the surrogate LPAR by using the `lvup_sock_helper` mechanism after the second boot of the surrogate. The surrogate LPAR uses this file to restore, by using a reverse setting operation (a rollback of the acquired mobile resources), the mobile resource for each CEC that is impacted by the PEP acquisition. A message is displayed if the resource restoration cannot be completed, which can occur when the resources that are temporarily used by AIX Live Update are allocated by other LPARs of the CEC before they can be released to the PEP pool and pool members.

AIX Live Update with Power Enterprise Pools 2.0

A classical PEP provides a convenient solution to the temporary resource requirements during an AIX Live Update operation. It uses the pool's inherent grace period of 48 hours to activate the extra resources that are needed while staying in compliance with the PEP rules. But in the context of the classical PEP implementation, it is necessary to activate resources that are available only in a powered-off state if the processor and memory requirements cannot be fulfilled by free active resources.

An even more efficient way to provide processor and memory resources for AIX Live Update is by using the Power Enterprise Pools 2.0 (PEP 2.0) technology, which has been available since May 2019 exclusively for pools that are built of Power E980 servers. With PEP 2.0, all processor and memory resources are instantaneously available for use because both processors and memory blocks are constantly in the power-on state.

For more information about PEPs 2.0 with utility capacity, see [IBM Knowledge Center](#).

1.1.6 Live Update CPU resource reduction

If there are enough free processor resources, AIX Live Update tries to assign the surrogate LPAR as many processing units as are assigned to the original LPAR. For a shared processor LPAR, if only the minimum number of processing units that is specified in the original LPAR's current configuration are available, the surrogate LPAR is created with processor resources that are specified by the minimum processing unit value of the original LPAR. After the original LPAR is deleted at the end of the Live Update operation, an attempt is made to add processing units to the surrogate LPAR to match what was the original LPAR's assigned capacity. The Live Update operation fails if there is not sufficient processor capacity within the pool of free processor resources (free pool) of the local system and if the requirement also cannot be satisfied either through Mobile CoD processors that are acquired from a PEP or by using Live Partition Mobility (LPM) to move the LPAR in question to a remote system.

To expand the constellations under which the Live Update operation succeeds, AIX 7.2 TL 3 offers an enhanced resource management option that you can use to reduce the processor resource requirement. This new feature is supported in HMC and IBM PowerVC managed environments, and it is controlled by the **cpu_reduction** parameter of the general stanza within the Live Update configuration file `/var/adm/ras/liveupdate/lvupdate.data`. The `lvupdate.data` file provides input parameters to the **geninstall** command that are used to initiate a Live Update operation. The `lvupdate.template` file in the `/var/adm/ras/liveupdate` directory provides configuration guidance to system administrators. As the file name suggests, this file is used as template to create a customized `lvupdate.data` file.

Figure 1-2 shows the help information for the **cpu_reduction** parameter that is included in the `/var/adm/ras/liveupdate/lvupdate.template` file.

```
[ ... omitted lines ... ]
#
# general:
#     kext_check = <yes | no> Blank defaults to yes. If no, the live update
#     operation will be attempted regardless as to whether all the loaded
#     kernel extensions are determined to be safe or not.
#     cpu_reduction = <yes | no> Blank or omitted defaults to no. If yes, the
#     live update operation will, in the absence of sufficient processing
#     capacity to create and boot the surrogate normally, determine whether
#     removing some capacity from the original would result in sufficient
#     available capacity, and if so, will automatically remove
#     the capacity from the original (later to restore the surrogate to
#     full capacity once the original is deleted).
#
[ ... omitted lines ... ]
```

Figure 1-2 The `cpu_reduction` parameter help text in the `lvupdate.template` file

A Live Update operation with CPU reduction enabled (`cpu_reduction = yes`) traverses three phases to reduce the processor resources that are required:

1. Information from the original LPAR and the original LPAR's hosting local system is gathered. For a Live Update operation in combination with LPM in an IBM PowerVC managed environment, the information is collected for the LPM target system.
2. Based on the information that is acquired in phase 1, a reduction plan is prepared and written to the `/var/adm/ras/liveupdate/liveupdate.cf` configuration file. For the HMC-based Live Update, the reduction plan accounts for the available Mobile CoD resources of a configured PEP.
3. The actual CPU reduction dynamic reconfiguration (dynamic LPAR (DLPAR)) operation is performed. The reduction plan ensures that enough processing units and virtual processors are initially assigned to the surrogate LPAR so that DLPAR add operations in the surrogate LPAR can be deferred until after the original LPAR is deleted.

Note: The combination of CPU reduction and LPM that is used by Live Update (LU) across frames entails various possible scenarios. The Live Update across frames operation is governed by a **destination** and a **force_migration** attribute in the `lvupdate.data` configuration file. You can use the **destination** parameter to run the Live Update locally, on a particular remote host, or on any remote host that is determined by IBM PowerVC management policies. The **force_migration** attribute with a specified destination (if the attribute is present and not blank) can specify either to favor a local Live Update or to enforce the execution on a remote frame. If the Live Update across frames configuration is set with `destination = ANY` and `force_migration = no`, the system processes the following priority list and acts:

1. The local host is checked for whether LU can proceed without CPU reduction.
2. Remote hosts are checked for whether LU can proceed without CPU reduction.
3. The local host is checked for whether LU can proceed with CPU reduction.
4. Remote hosts are checked for whether LU can proceed with CPU reduction.

Other combinations of the **destination** and **force_migration** parameter settings are less generic and can be intuitively interpreted. Section 1.1.7, "Live Update across frames" on page 15 provides more information about the combination of LPM with Live Update operations.

Phases 1 and 2 are performed as part of the Live Update preview process, which provides the system administrator with detailed information about the CPU reduction that is intended for the original LPAR. The third phase is conducted only when the Live Update operation is run.

The following LPAR-specific information is collected during the first phase:

- ▶ Minimum and assigned number of virtual processors
- ▶ Minimum and assigned number of processing units
- ▶ Minimum required processing units per virtual processor
- ▶ Shared pool ID, where -1 indicates dedicated processors, 0 refers to the default processor pool, and numbers larger than 0 are used for user-defined shared pools

In a HMC-managed environment, the available processing units in the free pool and the number of processing units that can potentially be acquired for the local system through Mobile CoD from a PEP are recorded.

All of these values are written to the `liveupdate.cf` configuration file under the `CPU_Reduc` stanza, as shown in Example 1-6.

Example 1-6 CPU_Reduc stanza in the liveupdate.cf file

```
CPU_Reduc:
  do = "yes"
  assignedVirtualProcessors = "4"
  assignedProcessingUnits = "30"
  minimumVirtualProcessors = "1"
  minimumProcessingUnits = "10"
  processingUnitsPerVirtualProcessor = "5"
  capacityIncrement = "1"
  sharedPoolId = "1"
  sharedPoolHeadroom = 170
  maximumPoolProcessingUnits = 200
  assignedPoolProcessingUnits = 30
  availableMobileProcessors = "0"
  availableProcessingUnits = "20"
  acquirableProcessingUnits = "20"
  processingUnitsShortfall = "0"
  processingUnitsReduction = "0"
  reducedProcessingUnits = "30"
  virtualProcessorsReduction = "0"
  reducedVirtualProcessors = "4"
```

The value of the `do` attribute is copied from the `cpu_reduction` attribute in the `lvupdate.data` file. The following 12 attributes reflect the information that is collected in phase 1 on page 12. The attributes `acquirableProcessingUnits` and `processingUnitsShortfall` are intermediate values that are used to compute the remaining four attributes: `processingUnitsReduction`, `reducedProcessingUnits`, `virtualProcessorsReduction`, and `reducedVirtualProcessors`. The four last named attributes determine the virtual processor and processing unit characteristics of the original LPAR after the CPU reduction is performed by Live Update in phase 3 on page 12.

The `acquirableProcessingUnits` attribute is computed by adding up what is available in the system free pool and what can be gained from Mobile CoD without exceeding any shared processor pool maximum processing units. The `processingUnitsShortfall` attribute is a number between 0 and the minimum required processing units that are reduced by the acquirable processing units. The minimum processing units are needed to start the surrogate LPAR. If the Live Update operation cannot get this minimum value, then you have a shortfall.

The `processingUnitsReduction` attribute designates how much processor capacity must be reduced from the original LPAR. Similarly, `virtualProcessorReduction` indicates how many virtual processors must be removed from the original LPAR. The `reducedProcessingUnits` and `reducedVirtualProcessors` attributes are then computed as the original LPAR's assigned capacity minus `processingUnitsReduction` and assigned virtual processors minus `virtualProcessorReduction`. After taking away the `processingUnitsReduction` capacity and `virtualProcessorReduction` virtual processors, the original LPAR is left with `reducedProcessingUnits` and `reducedVirtualProcessors` resources.

The output of the reduction plan that is created during the second phase is written to the `CPU_Usage` stanza in the `liveupdate.cf` configuration file, as shown in Example 1-7. This stanza describes the base parameter that determines the surrogate LPAR's boot characteristics.

Example 1-7 CPU_Usage stanza in the liveupdate.cf file

```
CPU_Usage:
    isBootable = "yes"
    virtualProcessors = "20"
    minimumProcessingUnits = "100"
    maximumProcessingUnits = "350"
```

The reduction plan attempts to maintain two key requirements:

1. Minimize how much the original LPAR is reduced, even if that means removing some of its virtual processors.
2. The number of virtual processors between the original and the surrogate LPARs must match to preserve processor bindings.

The reduction plan is designed in a way so that no extra CPUs need to be added to the surrogate LPAR until the original LPAR is shut down and deleted, and its CPUs freed. This situation means that the surrogate LPAR may start with less capacity than what is assigned to the original LPAR. The assigned processor capacity in the surrogate is selected in a range that is determined by the `minimumProcessingUnits` and `maximumProcessingUnits` parameters of the `CPU_Usage` stanza.

If CPU reduction is not permitted (either explicitly by setting `cpu_reduction = no` in the `lvupdate.data` file or by leaving this setting unspecified (default)) or when there is enough processor capacity available in the free pool of the system, the following *default plan* is employed:

- ▶ For dedicated processor LPARs, the surrogate LPAR is started with the same capacity as the original LPAR.
- ▶ For shared processor LPARs, the surrogate LPAR is started with at least enough capacity to support the same number of virtual processors that are assigned to the original LPAR.

If CPU reduction is permitted by setting `cpu_reduction = yes` in the `lvupdate.data` file, the reduction plan determines at first whether enough capacity is available to be removed from the original LPAR relative to what is needed (**processingUnitsShortfall**). If the outcome of this check is positive, `isBootable` is set to `yes` in the `CPU_Usage` stanza, which means that the surrogate LPAR can be configured and started. Under these conditions, either a dedicated or a shared CPU reduction plan is created:

Dedicated CPUs plan First, the plan determines the mean value of the original LPAR's assigned virtual processor number and the number of processors that are supported by the acquirable resources. The virtual processor number for dedicated processor LPARs is equivalent to the number of assigned physical processors, and the acquirable processor resources are a combination of the processor capacity of the free pool and the capacity that is available through Mobile CoD expressed in units of whole processor cores. The CPU reduction of the original LPAR (**virtualProcessorsReduction**) is now computed in a way that the processor resources are equally distributed between the original and the surrogate LPAR. After the dedicated CPU plans are employed, both LPARs run with a number of virtual and physical processors as defined by the mean value.

Shared CPUs plan

In the case of shared processor LPARs, the processing capacity of the surrogate LPAR is determined to be equal or above the original LPAR's defined minimum and to be sufficient to support enough virtual processors to accommodate the processing capacity of the (reduced) original LPAR during the Live Update operation. No additional processing capacity must be added to the surrogate LPAR until the original LPAR is shut down and deleted, and its CPUs freed.

The actual CPU reduction is the next step after the reduction plan is prepared. It has two parts to it:

1. Reducing the original LPAR's processing capacity and number of virtual processors.
2. Creating the surrogate LPAR with the correct profile setting.

The CPU reduction is done before the Live Update operation itself is registered as a dynamic reconfiguration event. In the context of Live Update with LPM, the CPU reduction is employed after LPM successfully moves the original LPAR to the target system.

If the AIX Live Update operation completed without errors, the original LPAR is deleted and the surrogate LPAR configuration is dynamically changed to exhibit the same processing units and virtual processors characteristics as the original LPAR had before the Live Update was initiated.

While a Live Update operation is running, external operations can claim the resources that were expected to be available either to start the surrogate partition or to restore the surrogate partition to its full capacity. In the former case, the Live Update stops. In the latter case, the surrogate partition runs with reduced capacity until the system administrator intervenes to restore it to full capacity manually.

1.1.7 Live Update across frames

The availability of CPU and memory resources on the managed system where the original partition resides is an essential requirement for the adoption of the AIX Live Update feature. On highly used systems, there are potentially not enough free CPU or memory resources to allow the creation of the surrogate partition, or not enough to run several Live Update operations concurrently.

Beginning with AIX 7.2 TL 3, the Live Update operation can be run on an alternative remote host if memory or processor resources are insufficient on the local host. This new feature enhances the automation characteristics of the Live Update operation by using the IBM PowerVC Live Migration technology.

Live Update initiates an LPM operation to move the original partition to an alternative host where it completes the concurrent OS software update before a second LPM operation transfers the new and updated surrogate partition back to the original host.

Note: AIX Live Update across frames is only available for IBM PowerVC controlled environments. Live Update across frames also requires the departure and the destination hosts to be managed by the same IBM PowerVC server. Environments that are exclusively managed by HMCs do not provide the required automation support to orchestrate the partition migration operation with Live Update.

The new feature is activated and controlled by the **destination** and **force_migration** parameters of the **pvc** stanza within the Live Update configuration file `/var/adm/ras/liveupdate/lvupdate.data`. The `lvupdate.data` file provides input parameters to the **geninstall** command, which is used to initiate a Live Update operation. The `lvupdate.template` file in the `/var/adm/ras/liveupdate` directory provides configuration guidance to system administrators. As the file name indicates, this file is used as template to create a customized `lvupdate.data` file.

Figure 1-3 shows the help information for the **destination** and **force_migration** parameters as included in the `/var/adm/ras/liveupdate/lvupdate.template` file.

```
#
# pvc:
[ ... omitted lines ... ]
#     destination = < IBM PowerVC host name | ANY > This attribute is used to
#     specify the host on which the Live Update operation will be executed
#     if the resources are insufficient on the local host, or if the
#     force_migration attribute is specified with a value equals to yes.
#     If the attribute is set to ANY, the destination will be selected
#     according to the placement policy of the host group in IBM PowerVC.
#     If this parameter or its value is not specified, the Live Update
#     operation is executed locally.
#     force_migration = < yes | no > When set to yes, the Live Update
#     operation is executed on the host specified by the destination
#     attribute even if the resources are sufficient locally.
#     This parameter is optional and can be set to yes only when a
#     destination is specified.
#
```

Figure 1-3 The **destination** and **force_migration** parameters help in the `lvupdate.template` file

The resource requirements to support a Live Update operation differ depending on the use of the CPU resource reduction feature, the availability of Mobile CoD resources, and the utilization of IBM PowerVC Live Migration. The CPU resource reduction feature helps to diminish the processor resources requirements by creating the surrogate partition with a minimum number of CPUs. (For more information about CPU resource reduction, see 1.1.6, “Live Update CPU resource reduction” on page 11.) Mobile CoD resources may be acquired to support Live Update if the local system is an active member of a PEP. (For more information about the use of PEP with Live Update, see 1.1.5, “AIX Live Update integration with Power Enterprise Pools” on page 7.)

The additional resource requirements to run Live Update on a local host are as follows:

- ▶ The same amount of memory as assigned to the original partition is needed to support the surrogate partition. The memory requirement can be satisfied by free memory resources (free pool resources) of the system or through Mobile CoD.
- ▶ For dedicated processor LPARs where CPU reduction is not permitted, the surrogate partition is started with the same processor cores as the original partition. The resources can be provided through the free processor resources (free pool resources) of the system or Mobile CoD.

- ▶ For shared processor LPARs where CPU reduction is not permitted, the surrogate LPAR is started with at least enough capacity to support the same number of virtual processors as are assigned to the original partition. If supported by the free pool or Mobile CoD, the surrogate runs with the same amount or processing units as are assigned to the original partition.
- ▶ For dedicated processor LPARs where CPU reduction is applied, the acquirable processor capacity expressed in units of whole processor cores determines the additional resource consumption.
- ▶ For shared processor LPARs where CPU reduction is applied, the processing capacity of the surrogate partition is determined to be equal or above the original partition's defined minimum and sufficient to support enough virtual processors to accommodate the processing capacity of the (reduced) original partition during the Live Update operation.

The resource requirements to run Live Update on a remote host are the same as described above, but with the additional requirement to support the original partition with the associated memory and processor resources on the remote host.

During the preview phase, Live Update reads these new pvc attributes (**destination** and **force_migration**) that are specified in the `lvupdate.data` file to check their validity and determine whether an LPM operation must be done. If a decision is made in favor of LPM, a new entry, `need_migration = 1`, is added to the pvc stanza in the `liveupdate.cf` configuration file to indicate that a Live Migration must be performed. The following conditions apply to the Live Update preview phase:

- ▶ If the **force_migration** attribute is set to an invalid value or set to yes without specifying a destination, Live Update fails during the parsing of the `lvupdate.data` file with a message indicating that the attribute is invalid.
- ▶ If the **destination** attribute is set to an invalid value (a destination that is not known by IBM PowerVC), Live Update fails during the configuration checks and notes that the destination host is not managed by IBM PowerVC.
- ▶ If the CPU/memory resources are insufficient on the local host and no destination host or local host is specified, Live Update fails during the configuration checks.
- ▶ If a valid destination name (besides the **ANY** keyword) is specified and either the CPU/memory resources are insufficient on the local host or the **force_migration** attribute is set to yes, Live Update fails during the configuration checks if the destination host does not meet the requirements of a valid remote destination as defined below. Otherwise, `need_migration = 1` is stored in the pvc stanza in the Live Update configuration file.
- ▶ If the CPU/memory resources are sufficient on the local host and either the **force_migration** attribute is not set to yes or the specified destination is the local host, `need_migration = 0` is stored in the pvc stanza in the Live Update configuration file.
- ▶ If the **ANY** keyword is specified as the destination and either the CPU/memory resources are insufficient on the local host or the **force_migration** attribute is set to yes, Live Update tries to find a valid destination host satisfying the requirements defined below. If a valid destination host cannot be found, Live Update fails during the configuration checks. Otherwise, `need_migration = 1` is stored in the pvc stanza in the Live Update configuration file.

A remote host is considered as a valid LPM destination if the following conditions are met:

- ▶ Enough resource capacity to host both the original and surrogate partitions is available:
 - Sufficient memory resources are available on the destination host.
 - Sufficient processing units are available on the destination host.
- ▶ The requirements to support LPM operations are fulfilled:
 - The original partition and the destination host are in the same IBM PowerVC host group.
 - PowerVM Enterprise Edition is activated on both the original and destination hosts.
 - The destination host supports the processor compatibility mode of the original partition.
 - The original and destination hosts have the same logical-memory block (LMB) size.
 - The original and destination hosts are compatible from a storage perspective (verified by the IBM PowerVC inherent verification methodology).
 - The original and destination hosts are compatible from a network perspective (verified by the IBM PowerVC inherent verification methodology for each network port).

Note: The decision to move to a different host or to stay on the local host is made during the preview phase based on the CPU and memory resources that are available then. This decision is final and cannot change, even if resources become available or are taken away between the preview and the point where those resources are used by Live Update. Therefore, if Live Update decides to move to a remote host, then this is indicated during the preview by a specific message, and the migration phases are displayed on the Live Update output.

When the **destination** attribute is set in the `lvupdate.data` file to the special uppercase value **ANY**, then the destination host is automatically selected within the same *host group* as the original host. IBM PowerVC supports host groups as a means of deploying or migrating partitions without having to specify a specific destination host. When the target is a host group rather than a specific host, IBM PowerVC automatically chooses a host within the host group based on the *placement policy* of the host group. The following six placement policies are applied to host groups in IBM PowerVC:

| | |
|------------------------------------|--|
| Stripping | The host with the lowest number of partitions is selected. |
| Packing | The host with the greatest number of partitions is selected. |
| CPU allocation balance | The host that would end up with the smallest percentage of its processing units allocated is selected. |
| Memory allocation balance | The host that would end up with the smallest percentage of its memory allocated is selected. |
| CPU utilization balanced | The host with the lowest CPU utilization is selected. |
| Memory utilization balanced | The host with the lowest memory utilization is selected. |

Although IBM PowerVC can select a host within a host group, it would do so while ignoring the fact that Live Update creates an extra surrogate partition on the destination host. Therefore, the destination host is selected by Live Update. Live Update accepts the IBM PowerVC placement policy of the host group, but also accounts for the additional CPU and memory resources that are required for the surrogate partition.

The Live Update across frames feature is fully supported by NIM. Moving to a different host if resources are insufficient is achieved by specifying a destination host in the `live_update_data` NIM resource. The partition relocation can be optionally forced by specifying `force_migration = yes` in the `live_update_data` resource. Because the original partition is moved to a different host only during the Live Update operation, this change is not reflected in the NIM database.

1.1.8 Automount File System support with Live Update

AIX 7.2 TL 4 introduces support for active Automount File System (AutoFS) mounts while a Live Update operation is run. This feature removes a significant limitation of the Live Update feature and expands the configuration environments that are supported for concurrent OS software updates.

Beginning with AIX 7.2 TL 4, AutoFS mounts are checkpointed on the original partition and restored on the surrogate partition while the network activity is suspended during a Live Update operation. The implementation covers all supported AutoFS map types, which include the following items:

- ▶ Direct maps
- ▶ Indirect maps
- ▶ Executable maps
- ▶ Host maps
- ▶ Network Information Services (NIS) / Lightweight Directory Access Protocol (LDAP) maps

AutoFS mounts and unmounts are permitted during the entire duration of a Live Update operation. However, if an AutoFS is accessed during Live Update and the addressed Network File System (NFS) file system is not already mounted, the Live Update operation might fail. In this way, the AutoFS mount request is given precedence over Live Update to ensure the functioning of critical applications relying on the AutoFS function.

1.1.9 Kerberos authentication support in Live Update

Kerberos is a computer-network authentication protocol that is widely used to support single sign-on infrastructures. AIX provides both the Kerberos client and server functions through the IBM Network Authentication Service (IBM NAS) product that is distributed through the AIX Expansion Pack.

AIX 7.2 TL 4 enhances the Live Update implementation to cover environments with active Kerberos NAS services. The supported server configurations and client services are listed below:

- ▶ Valid NAS server configurations that are supported in the context of Live Update:
 - NAS server configurations with a local file system repository (Kerberos legacy database)
 - NAS server configurations with an LDAP repository
- ▶ Valid NAS client Kerberos enabled services that are supported by Live Update:
 - AIX system user authentication
 - Remote commands: **rlogin**, **rcp**, **rsh**, **telnet**, and **ftp**
 - Secure Shell (SSH) and Secure Copy (SCP)
 - NFS server and client services that are based on the NFS protocol Version 2, 3, and 4, and the Kerberos security methods **krb5**, **krb5i**, and **krb5p**

The following limitations apply:

- ▶ Only the IBM NAS Version 1.6.0.4 server and client is supported (Kerberos V5).
- ▶ For the IBM NAS server with an LDAP repository setup, if this LDAP repository is on the same host, only IBM Tivoli Directory Server V6.3.30 or greater is supported.
- ▶ IPsec is unsupported by Live Update, so IPsec with Kerberos is not supported.
- ▶ Kerberos configuration changes during Live Update are not supported.
- ▶ NFS configuration changes during Live Update are not supported.

If NFS-mounted file systems with Kerberos security are on the LPAR, any required NFS daemons must be active while running a Live Update operation. In particular, the **gssd** daemon must be active and started from the system resource controller (SRC) or Live Update cannot mount the NFS file systems on the surrogate and fail.

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 a 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 storage area network (SAN)-based.

It is an emerging technology because of its advantages, and one of which is faster access to data. Enterprise-class customers can have faster access to data in their production environment. To use a flash cache, you must have a set of flash drives and solid-state drive (SSD) capable adapters that are connected to the flash drives.

1.2.1 Flash caching concepts

The caching function can be enabled dynamically while the workload is running because the act of starting to cache does not require the workload to be brought down to a quiescent state. The caching is also transparent to the workload. Consider the following characteristics:

- ▶ The cache devices can be server-attached flash (built-in SSD drives in the server), flash devices that are directly attached by 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 that is requested is not found in the cache or if it is a write request, it falls through to the original storage.

Flash cache terminology

Flash cache has several terms and concepts that form its structure:

| | |
|---------------------|---|
| Cache device | Any SSD or flash storage that is used for caching. |
| Cache pool | A group of cache devices that is used only for storage caching. It provides simplified management of multiple flash disk devices. Initially, only a single cache pool is supported, but more 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 used 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 Virtual I/O Server (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 decides on what blocks in the storage must 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 about how the cache device is provisioned to the AIX LPAR. These modes are described in the following sections.

Dedicated

In dedicated mode, the cache device is directly provisioned to the AIX LPAR. A cache pool must 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.

Because the cache device is dedicated to this LPAR, the LPAR cannot use LPM. If the LPAR must be migrated to another service, the caching must be manually stopped, and the cache device unconfigured in preparation for the migration.

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

1. Create a cache pool and a cache partition, as shown in Example 1-8

Example 1-8 Creating a cache pool and cache partition in a dedicated flash logical unit number

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

2. Assign the cache partition to target devices `hdisk2` in which we want to cache, as described in Example 1-9.

Example 1-9 Assigning a cache partition to a target device that must be cached

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

3. Start caching on the target device and monitor the status, as shown in Example 1-10.

Example 1-10 Starting caching on the `hdisk2` target and monitoring the status

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

Virtual SCSI mode

In the virtual SCSI (vSCSI) 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 the device is discovered on the AIX LPAR, the partition can be used to cache a target device. Because this cache device is virtual, the partition can be migrated to another server.

Before starting 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. After the migration is completed, caching is automatically started on the destination. In this case, the cache starts 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 the VIOS, complete the following steps:

1. Create a cache pool and a cache partition on VIOS server side, as shown in Example 1-11.

Example 1-11 Creating a cache pool and cache partition in a VIOS

```
# 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 vhost in VIOS, as described in Example 1-12.

Example 1-12 Assigning the cache partition to a vhost in a VIOS

```
# cache_mgt partition assign -P part1 -v vhost0
```

In the AIX LPAR, complete the following steps:

1. Assign the cache partition to a target device in the AIX LPAR, as described in Example 1-13.

Example 1-13 Assigning a cache partition to a target hdisk2 device in the 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 1-14.

Example 1-14 Starting caching on the hdisk2 target and monitoring the status

```
# cache_mgt cache start -t hdisk2
```

```
# cache_mgt monitor get -h -s
```

N_Port ID Virtualization (NPIV) mode

In this mode, the cache device is available as a virtual Fibre Channel (FC) device on the AIX LPAR. A cache pool can be created on the AIX LPAR on which only one cache partition may be created. The cache partition can be used to cache any number of target devices on this LPAR. Because the cache device is available from the SAN, the LPAR can be migrated to another server. The cache device must be made visible on the target system. Caching can continue to occur throughout the migration process, and the cache continues to stay populated after the migration completes.

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

1. Create a cache pool and a cache partition, as shown in Example 1-15.

Example 1-15 Creating a cache pool and cache partition in an NPIV flash logical unit number

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

2. Assign the cache partition to target device `hdisk2` in which we want to cache, as described in Example 1-16.

Example 1-16 Assigning a cache partition to a target device that must be cached

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

3. Start caching on the target device and monitor the status, as shown in Example 1-17.

Example 1-17 Starting caching on a `hdisk2` target and monitoring the status

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

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, iSCSI, and FC disk device drivers and their device methods support MPIO disk devices. In this section, we describe the AIX Path Control Module (AIXPCM).

1.3.1 AIX Path Control Module

The native 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 usable for sending I/O.
- ▶ Enable a path that was previously marked failed because of a temporary path fault.
- ▶ Check unused paths that would be used if a failover occurred.

AIXPCM benefits from being integrated into the OS, which removes the need for separate maintenance activities that are related to updating the PCM. By being integrated, your regular OS maintenance results in the PCM being updated, which simplifies the management of your AIX environments.

When using the default AIXPCM, you can 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 can be defined on a per device basis if required.

I/O algorithms

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

- | | |
|-----------------------|--|
| fail_over | In this mode, I/O is sent over one enabled path, and another path is used only 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 FC 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 is 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 is used. In this mode, path priority is ignored, and paths are selected based on the number 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 that have a state of open. A device that is not in use does 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 are enabled for the device, which includes paths that failed. |
| Failed | In this mode, the healthcheck command is sent to all paths that are in a 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, which includes paths that are in enabled and failed states. 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 0 - 3600, and it represents the time in seconds between polling. If a value of 0 is specified, it 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 might 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:

| | |
|---------------------------|---|
| <code>no_reserve</code> | This policy does not apply any reservation on the target device allowing initiators (paths) on the same system, and on other systems, access to the target device. This is the recommended policy for devices where disks are shared between hosts and devices that have the <code>shortest_queue</code> or <code>round_robin</code> algorithms configured. |
| <code>single_path</code> | This is the default policy when using AIXPCM. This policy places an SCSI2 reserve on a target device so that the device can be accessed only 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. |
| <code>PR_exclusive</code> | This policy applies an SCSI3 persistent-reserve with exclusive-host methodology on the device when the device is opened to exclusively lock it to a single host. A <code>PR_key_value</code> attribute must also be set on the device when using this mode to uniquely identify the host. |
| <code>PR_shared</code> | This policy applies an 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 <code>PR_key_value</code> 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 value depends on multiple factors, such as your storage systems throughput and latency, and application requirements.

Checking paths by using AIXPCM

The main command that is used to display information about your MPIO configuration is the `lsmpio` command, which is shown in Example 1-18. In this example, we used the `-l` flag to specify the disk that we wanted to see the status of. By omitting this flag, you see the status of all MPIO disks.

Example 1-18 Using `lsmpio` to check the path status

```
# lsmpio -l hdisk0
name    path_id  status  path_status  parent  connection
-----
hdisk0  0        Enabled Non          fscsi0  5005076802360f79,0
hdisk0  1        Enabled Se1,Opt    fscsi0  5005076802360f7a,0
hdisk0  2        Enabled Non          fscsi1  5005076802260f79,0
hdisk0  3        Enabled Opt          fscsi1  5005076802260f7a,0
```

Alternatively, use the `lsmpio -o` command to check the status of paths. This command differs from running the `lsmpio` command by itself because it directs the AIX disk driver to attempt to access all MPIO paths that are associated with the specified MPIO disk, including paths that are marked as failed when the MPIO disk was last closed. By using the `lsmpio -o` command, you receive the current status of the MPIO disk paths.

The `path_status` column provides information about preferred paths and path failures.

The possible values for **path_status** are described in Table 1-1.

Table 1-1 MPIO path_status definitions

| The path_status value | Description |
|-----------------------|---|
| Opt | 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. |
| Non | 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. |
| Act | 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. |
| Pas | 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. |
| Sel | Indicates that the path is being selected for I/O operations at the time when the lsmpio command is run. |
| Rsv | 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. |
| Fai | Indicates that the path experienced a failure. It is possible for a path to have a status value of <code>Enabled</code> and still have a path_status value of <code>Fai</code> . This scenario indicates that operations that are sent on this path are failing, but AIX MPIO has not marked the path as <i>Failed</i> . In some cases, AIX MPIO leaves one path to the device in the <code>Enabled</code> state, even when all paths are experiencing errors. |
| Deg | 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. |
| Clo | Indicates that the path is closed. If all paths to a device are closed, the device is considered to be closed. If only some paths are closed, then those paths might have experienced errors during the last time the device was opened. The AIX MPIO periodically attempts to recover closed paths until the device path is open. |

To see connectivity information for an FC adapter, you can use the **lsmpio -are** command. This command shows the connectivity status for each of your FC adapters along with the remote ports they are connected to on your storage. This command also provides you with error statistics for the connection to each remote port, as shown in Example 1-19.

Example 1-19 Displaying parent adapters and remote ports

```
# lsmpio -are
Adapter Driver: fscsil -> 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 that are 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 and what type of error was detected by the AIXPCM.

Example 1-20 shows a sample output of the detailed statistics that are provided when using AIXPCM.

Example 1-20 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 needed values by using the **chdev** command. If the device is in use, you must use the **-U** flag 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 1-21, where we update the **queue_depth** to 32. The same method can be used to update the **algorithm** attribute.

Example 1-21 Dynamic queue_depth update for device

```

# 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 can have their attributes set by using the **chdef** command so that these disks have your value set for the attribute.

To change the default value for an FC device by using AIX PCM, you can run the command **chdef -t mpioosdisk -c disk -s fcp -a <attribute>=<value>**, as shown in Example 1-22.

Example 1-22 Changing the default reserve_policy to no_reserve for AIXPCM FC disk

```

# 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 1-23.

Example 1-23 Changing the default algorithm to `shortest_queue` for the AIXPCM iSCSI disk

```
# 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 by running the command `chpath -l <device> -p <parent adapter> -s <disable|enable>`.

Example 1-24 demonstrates disabling all paths on `fscsi0` for `hdisk0`. Using `chpath` to disable a path disables future I/O over that path *only*, not I/O that is in progress.

Example 1-24 Disabling paths by using `chpath`

```
# 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
hdisk0  0        Disabled Non          fscsi0  5005076802360f79,0
hdisk0  1        Disabled Opt       fscsi0  5005076802360f7a,0
hdisk0  2        Enabled  Non          fscsi1  5005076802260f79,0
hdisk0  3        Enabled  Sel,Opt     fscsi1  5005076802260f7a,0
```

1.3.2 Subsystem Device Path Control Module

Previously, when connecting to IBM SAN Volume Controller and Storwize® storage devices the recommended storage driver was Subsystem Device Path Control Module (SDDPCM). On IBM POWER9™ processor-based systems, SDDPCM is no longer supported, so as a best practice, use AIXPCM on them, and on IBM SAN Volume Controller and Storwize devices running microcode levels V7.6.1 and above.

AIXPCM now includes all significant functions that are available within SDDPCM.

For more information about how to migrate from SDDPCM to AIXPCM, see [How To Migrate SDDPCM to AIXPCM](#).

Note: When migrating from SDDPCM to AIXPCM, your device attributes reset to the system defaults, including the values for `queue_depth`, `algorithm`, and `reserve_policy`. If you were using the `load_balance` algorithm before migration, the `shortest_queue` algorithm from AIXPCM provides the same function. 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 by using any network adapter. In this section, we describe what iSCSI is and how to use iSCSI on AIX.

1.4.1 iSCSI overview

iSCSI is a protocol that uses the 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 [RFC3270 - Internet Small Computer Systems Interface \(iSCSI\)](#).

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 Command Descriptor Block (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 a globally unique iSCSI name that 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 that have the necessary AIX Object Data Management (ODM) definitions. For more information about the improvements to native MPIO, see 1.3, “Multipath I/O” on page 23.

For more information about iSCSI, see Chapter 2, “Introduction to iSCSI in IBM Storwize storage systems”, in *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 notice a new device that is labeled `iscsi0` on your AIX system. This device is the *iSCSI Protocol Device*, and it is 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 by using `smitty` or by running the fast path `smitty chgiscsisw`.

Initiator name

The iSCSI `initiator_name` is an attribute that 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 following syntax:

```
iqn.<hostname>.hostid.<hostid>
```

The values for `hostname` and `hostid` are not dynamic, and are set only at the initial configuration of the device. The `hostname` value is the host name of the AIX system, and `hostid` is the globally unique identifier for the AIX host that is 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 you set the name manually, ensure that the name is globally unique.

Protocol features

AIX 7.2 TL 4 introduces new SCSI protocol feature support that you can use to modify the behavior of data transfers:

| | |
|-----------------------|--|
| initial_r2t | Indicates whether an iSCSI initial Ready to Transfer (R2T) Protocol Data Unit (PDU) is required for each write. This value can be set to either yes or no. The default value of this attribute is yes. |
| immediate_data | Turn on or off support for iSCSI immediate data for this device. This is an even more aggressive form of <code>InitialR2T=no</code> that allows the write data to be sent on the same PDU that sends the SCSI WRITE CDB instead of sending a SCSI CDB PDU followed by a Data Out PDU. This value can be set to either yes or no. The default value for this attribute is no. |

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`, which determines how many times the iSCSI initiator attempts to recover from network errors. We have two possible settings for the recovery policy:

| | |
|---------------------|--|
| delayed_fail | This is the default policy that is 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 so that AIX can switch over more quickly to working paths if network outages occur due to one path for an iSCSI target. |

Maximum targets that are 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 know the targets that 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 that are available for the iSCSI software initiator:

| | |
|-------------|--|
| file | Information about targets is specified in a configuration file that is pointed to by the <code>disc_filename</code> attribute on the device. This is the default discovery mode for the iSCSI initiator and the default <code>disc_filename</code> value is <code>/etc/iscsi/targets</code> . This file contains examples of 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 also must set the attributes **isns_srvports** and **isns_srvnames**. The **isns_srvports** attribute is used to define the IP addresses of your iSNS servers, and **isns_srvports** defines the port on which they are listening.
- 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.

Example 1-25 shows 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` (use more secure passwords than this example).

Example 1-25 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
# echo '9.114.181.38 3260 iqn.1986-03.com.ibm:2145.cluster9.114.181.189.node2
"abcd1234"' >> /etc/iscsi/targets
# cfgmgr -l iscsi0
# lsdev -p iscsi0
hdisk3 Available MPI0 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 the discovery mode is set to **odm**, the configuration for the targets can be listed, removed, and added by using the **lscsi**, **rmiscsi**, and **mkiscsi** commands. Example 1-26 demonstrates adding a target with the discovery policy set to **odm** and by using the **mkiscsi** command.

Example 1-26 ODM-based iSCSI discovery

```
# lsdev -p iscsi0
# lsattr -El iscsi0 -a disc_policy
disc_policy file Discovery Policy True
# chdev -l iscsi0 -a disc_policy=odm
iscsi0 changed
# lsattr -El iscsi0 -a disc_policy
disc_policy odm Discovery Policy True
# mkiscsi -l iscsi0 -g static \
> -t iqn.1986-03.com.ibm:2145.cluster9.114.181.189.node2 \
> -n 3260 \
> -i 9.114.181.38 \
> -p "abcd1234"
# lscsi
```

```

iscsi0      9.114.181.38    3260
iqn.1986-03.com.ibm:2145.cluster9.114.181.189.node2
# cfgmgr -l iscsi0
# lsdev -p iscsi0
hdisk3 Available MPI0 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 your environment had multiple paths to the iSCSI target, repeat the process to discover each additional path.

Creating multiple iSCSI Protocol Devices

AIX 7.2 TL 4 also introduces support for creating multiple iSCSI Protocol Devices. By creating multiple iSCSI Protocol Devices, you can improve utilization of hardware by allowing for more concurrent processing. By using multiple iSCSI initiators, you can also increase the number of TCP/IP sessions that is used for iSCSI sessions. By creating multiple iSCSI initiators, you can use different discovery policies and attributes for each initiator based on your individual requirements.

To create a second iSCSI Protocol Device, run the `mkdev -c driver -s node -t iscsi -a initiator_name="aixlpar"` command, where `aixlpar` should be replaced by a unique initiator name.

You can also create iSCSI Protocol Devices by using the fast path `smitty addiscsiw`, as shown in Example 1-27.

Example 1-27 Add an iSCSI Target Protocol Device smitty menu

Add an iSCSI Target Protocol Device

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

| | | |
|---|----------------------|----|
| | [Entry Fields] | |
| iSCSI Initiator Name | [] | |
| Discovery Policy | file | + |
| Configuration file | [/etc/iscsi/targets] | / |
| Support ImmediateData | no | + |
| Require InitialR2T | yes | + |
| iSCSI Network Error Recovery Policy | delayed_fail | + |
| Maximum Targets Allowed | [16] | +# |
| Maximum transfer size | [0x80000] | +# |
| Maximum number of commands to queue to driver | [200] | +# |
| iSNS Servers IP Addresses | [auto] | |
| iSNS Servers Port Numbers | [] | |
| Apply change to DATABASE only | no | + |

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

1.5 Network Installation Manager

NIM is an application that can run on AIX partitions, which provides an environment for installation and management AIX file sets. This application works on a client/server model and can be used to manage a NIM client's file sets and perform base operating installations on systems. The fundamental concepts of NIM and its usage are described in *NIM from A to Z in AIX 5L*, SG24-7296. This section covers the key enhancements that were made available since its release up to AIX 7.2 TL3 SP3.

1.5.1 Object classes

The NIM database is stored in the ODM repository. The resources are broken into four classes with the default installation of AIX 7.2 and are further extended to a fifth class that is 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 their purpose.

Table 1-2 List of object classes in NIM

| Class | Description |
|-------------------------|--|
| groups | Class of objects that represents a group of machines or other resources. |
| machines | Class of objects that represents machines. |
| management ^a | Class of objects that represents machine control points. |
| networks | Class of objects that represents networks. |
| resources | Class of object that represents installation resources. |

a. This class is visible on a NIM master only if the DSM file set `dsm.core` is installed. This file set is available on the AIX Installation Base Media on AIX 7.1 and above.

Management object class

When you install the DSM `dsm.core` file set, an extra set of resources may be defined that you can use to define control points for your LPARs. The management class objects that are available are shown in Table 1-3.

Table 1-3 Management object types and description

| Object name | Description |
|------------------------|--|
| <code>nas_filer</code> | Defines an object that represents a network attached file system device. |
| <code>hmc</code> | Defines an object that represents an HMC device. |
| <code>cec</code> | Defines an object that represents a CEC. |
| <code>vios</code> | Defines an object that represents a VIOS. |
| <code>ivm</code> | Defines an object that represents the Integrated Virtualization Manager (IVM). |
| <code>bcmm</code> | Defines an object that represents a Blade Center Management Module. |
| <code>powervc</code> | Defines an object that represents an IBM PowerVC management console. |

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

An example of this command is shown in Example 1-30 on page 37 when creating an HMC object.

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

1. Create the HMC object, as shown in Example 1-30 on page 37.
2. Define the CEC object, as shown in Example 1-31 on page 38.
3. Define the AIX partition, as shown in Example 1-33 on page 39.
4. Associate the AIX partition with the CEC it is on and set its LPAR ID, as shown in Example 1-34 on page 39.

By defining this association, you can perform operations on a machine such as a Live Update, which requires 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 is available. The `nimhttp` service is defined in `/etc/services`, and the `nimhttp` daemon listens for requests over port 4901. When the `nimhttp` service is active, NIM clients attempt to access supported resources on the NIM server by using the HTTP service on the port that is listed for the `nimhttp` service in the `/etc/services` file. If HTTP access fails or if the access is denied, an access failover attempt to the NFS client occurs. The NIM resources that support HTTP access are as follows:

- ▶ `file_res`
- ▶ `fix_bundle`
- ▶ `installp_bundle`
- ▶ `lpp_source`
- ▶ `script`

Enabling HTTP service

To enable the NIM HTTP service, run the `nimconfig -h` command, as shown in Example 1-28.

Example 1-28 Enabling the `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        nimhttp        10354954      active
```

After the service starts for the first time, the configuration file for the daemon is created in `/httpd.conf`. The log file for the HTTP service is in `/var/adm/ras/nimhttp.log`.

If NIM is not configured to use cryptographic authentication, the configured **nimhttp** service and all resources that support the HTTP service are accessible directly through `http://<ip>:4901`, as shown in Example 1-29. If you want to prevent access through this URL, enable cryptographic authentication in NIM.

Example 1-29 Using nimhttp without using 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 Sockets Layer (SSL) communication, control traffic is encrypted when you use the NIM HTTP service. After enabling the cryptographic authentication by using the **nimconfig -c** command, you must stop and start the **nimhttp** service if it is already running by running **stopsrc -s nimhttp** and then restart the service by running **startsrc -s nimhttp** for the changes to take effect.

Note: If the **nimhttp** service is already running and you run **nimconfig -c** to regenerate certificates, you must first stop the **nimhttp** service by using the **stopsrc -s nimhttp** command and then restart it after regenerating the certificates by using the **startsrc -s nimhttp** command.

To configure the NIM client, you also must run the **nimclient -c** command on the NIM client and ensure that the communication method on the client is configured for **nimsh**.

1.5.3 Live Update

With the introduction of Live Update (as described in 1.1, “Live Update function” on page 2), NIM also supports performing updates by using Live Update. This section describes how to prepare your environment to perform NIM updates by using Live Update.

Preparing for Live Update

To prepare your NIM server so that you can use Live Update, you must define many NIM resources:

- ▶ An object representing the HMC
- ▶ An object representing the CEC
- ▶ An object representing the AIX partition
- ▶ An object representing the Live Update configuration

Note: To add the HMC and CEC object type, you must ensure that you have installed the DSM packages `dsm.core` file set. This file set is not installed by default on AIX, but is available on the AIX base installation media. For more information regarding DSM, see section 5.2, “Distributed System Management” in *IBM AIX Version 7.1 Differences Guide*, SG24-7910.

Defining the Hardware Management Console

To define the HMC, complete the following steps:

1. Generate an encrypted password file to access the HMC by using the `dpasswd -f EncryptedPasswordFilePath -U hmcLogin -P hmcPassword` command.

Note: As a best practice, exclude the `-P` flag from the `dpasswd` command so that the command prompts for a password to prevent your HMC password from appearing in your shell history as clear text.

2. After generating the encrypted password file, define the HMC object by running the following command:

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

3. Run the `dkeyexch -f password_file -I hmc -H host` command to exchange keys between the NIM master server and the HMC. When you use SSH to access the HMC from the NIM master, you will not be prompted for a password.

Example 1-30 demonstrates the creation of an HMC object for a device with the following information:

- ▶ Host name: hmc01
- ▶ Password: abc123
- ▶ Gateway: 9.47.79.254
- ▶ Subnet mask: 255.255.240.0
- ▶ Encrypted password file path: /export/hmc/hmc_passwd

Example 1-30 Creating an HMC 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  
Reenter 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  
type = hmc
```

```

passwd_file = /export/hmc/hmc_passwd
mgmt_port   = 12443
if1         = vlan2230 hmc01 0
Cstate      = ready for a NIM operation
prev_state  =
Mstate      = currently running
# dkeyexch -f /export/hmc/hmc_passwd -I hmc -H hmc01
OpenSSH_7.5p1, OpenSSL 1.0.2r 26 Feb 2019

```

Defining the CEC

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

Using *nimquery* to define an object

To define the CEC object in one step, run the **nimquery -a hmc=hmcObjectName -d** command.

This command discovers all managed systems that are attached to the HMC, registers them as CEC management class objects, and automatically names them. Example 1-31 demonstrates this function and the resultant objects that are defined in NIM.

Example 1-31 Using *nimquery* to define the CEC object

```
# 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 CEC management object is to define manually the object by using the following command:

```

# 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 must obtain the CEC serial number, the hardware type, hardware model, name of the HMC object that you created, and the object name of the CEC object that you want to define.

You can also use the `nimquery` command to obtain the details of the CECs that you want to add to the system by omitting the `-d` flag. This syntax prints the details of the CECs but does not automatically add them to NIM. Alternatively, you can obtain the details directly from the HMC by running the `lssyscfg -r sys -F name,type_model,serial_num` command.

By defining a CEC object manually, you may select your own name for the CEC, as shown in Example 1-32. In this example, the HMC resource name is `hmc01` and is named `TESTSYS`.

Example 1-32 Manually defining a CEC

```
# 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 CEC, define the NIM client. Registering stand-alone clients in NIM is covered in detail in *NIM from A to Z in AIX 5L*, SG24-7296. If the AIX server you want to update is already running, you perform this step by logging in to the client partition and by using the `niminit` command. In Example 1-33, we use the `niminit` command to register and define our client to the NIM master with the host name `nimlpar` by using the `nimsh` protocol.

Example 1-33 Registering the NIM client by using niminit on the 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, update the `mgmt_source` and identity attributes for the stand-alone machine object. The `mgmt_source` is the CEC management object that the partition is on, and the identity is the partition ID of the AIX client partition. In Example 1-34, we log in to the NIM master and update an existing NIM clients `mgmt_source` and identity attributes. In this case, `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 1-34 Adding a 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
  connect    = nimsh
  platform   = chrp
  netboot_kernel = up
  if1        = vlan2230 aixlpar FA3953FB9220 ent0
  cable_type1 = bnc
```

```
mgmt_profile1 = hmc01 5 8286-41A_212B8BW
Cstate        = ready for a NIM operation
prev_state    = ready for a NIM operation
Mstate        = currently running
cpuid         = 00FA2B8B4C00
```

Defining the Live Update configuration resource

To use Live Update, you need a `lvupdate.data` file. You have two options:

- ▶ The file can be on the virtual client.
- ▶ The file can be defined as a resource in NIM and used when triggering a Live Update.

A template of this file can be found in `/var/adm/ras/liveupdate/lvupdate.template` and can be used as a basis for the configuration of your `lvupdate.data` file for the AIX LPAR on which you want to use Live Update. Details regarding Live Update configuration are available in 1.1, “Live Update function” on page 2.

After you create your `lvupdate.data` file, create the NIM resource by using the following command:

```
nim -o define -t live_update_data -a location=<path to file> -a server=<server that hosts resource>
```

Example 1-35 shows the creation of the Live Update data resource by using a configured `lvupdate.data` file that is named `aixlpar_lvupdate.data`.

Example 1-35 Creating a Live Update data resource

```
# 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 Live Update

After the required resources that are related to your virtual partition are defined in your NIM master server, you may use your **lpp_source** resources to perform Live Update on your AIX partitions. These updates can be initiated either from the NIM client or from the NIM master.

When you use the NIM master to do customization (by using `nim -o cust -a live_update=yes`), it triggers the NIM client to initiate a Live Update-based `installp`. If you created a `live_update_data` resource and want to use it for your NIM installation, specify the following command:

```
nim -o cust -a live_update=yes live_update_data=lvupdate_data_resource
```

In Example 1-36, we perform a Live Update by using a `live_update_data` resource that is named `aixlpar_live_update_data` and perform an `update_all` customization to a server. If you omit the `live_update_data` value, NIM uses the `lvupdate.data` file that is located locally on the client itself.

Example 1-36 Live Update initiated from the NIM 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 Update operation begins in 10 seconds.  
  
Initializing Live Update on original LPAR.  
  
Validating original LPAR environment.  
  
Beginning Live Update operation on original LPAR.  
  
Requesting resources required for Live Update.  
.....  
Notifying applications of impending Live 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 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 Live Update from the NIM client is similar to one that is initiated by the NIM master; the only difference is that instead of the `nim` command, you run the `nimclient` command.

1.5.4 The `nimadm` support for MultiBOS environments

With the `multibos` command, root-level administrator can 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 action creates two bootable sets of BOS on a rootvg.

Before AIX 7.2, the `nimadm` command did not support the migration of AIX instances that were configured for MultiBOS. With AIX 7.2, the `nimadm` command successfully performs preservation or migration installations of AIX instances to the next level of AIX that is available.

1.6 Logical Volume Manager

In this section, we describe the recent enhancements to the Logical Volume Manager (LVM) in AIX.

1.6.1 LVM mirroring to IBM FlashSystem for enhanced performance

In many cases, customers choose to include IBM FlashSystem® storage as the preferred read LVM mirror on AIX, which provides enhanced I/O read performance and resiliency.

An AIX administrator can add the hdisks to the VG, mirror the data, and set the logical volume scheduling policy to parallel/sequential, which indicates that writes are done to both copies in parallel and reads come from the first LVM copy, which should be on the IBM FlashSystem server.

Setting the preferred read to a specific copy of a mirrored logical volume is done by running `mk1v -R`, which sets the read preference to the copy of the logical volume. If the `-R` option is specified and 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 0 - 3. The default value is 0. To change the read preferences, run `ch1v -R`, which changes the preferred read copy of the logical volume. Always read from the preferred copy if the preferred copy is available. If the preferred copy is not available, then the reads follow the scheduling policy of the logical volume. The `PreferredRead` variable can be set to a value 0 - 3. Setting the `PreferredRead` variable to 0 disables the preferred read copy of the logical volume.

So, you can speed up an application by accelerating the read I/Os. Implementing preferred read with the IBM FlashSystem provides an easy way to deploy the IBM FlashSystem in an existing environment. The data is secured by writing it to two different storage systems. Data is read at IBM FlashSystem speed because it is always read from the IBM FlashSystem. This implementation does not change the existing infrastructure concepts, for example, data security, replication, backup, and DR.

On AIX, preferred read is implemented by the AIX LVM. The following steps show you how to create a preferred read configuration with the IBM FlashSystem. The steps assume that the AIX server is cabled and zoned correctly.

1. Create a file system on a HDD.
2. Add the IBM FlashSystem as a mirrored copy to this file system.
3. Set the correct read and write policy.
4. Set the preferred read to the IBM FlashSystem.

You may also set the scheduling policy for reads and writes to the storage systems. If you use mirrored logical volumes, the following scheduling policies for writing to disk can be set for a logical volume with multiple copies:

Sequential scheduling policy

Performs writes to multiple copies or mirrors in order. The multiple physical partitions (PPs) representing the mirrored copies of a single LPAR are designated primary, secondary, and tertiary. In sequential scheduling, the PPs are written to in sequence. The system waits for the write operation for one PP to complete before starting the write operation for the next one. When all write operations are complete for all mirrors, the write operation is complete.

Parallel scheduling policy

Simultaneously starts the write operation for all the PPs in an LPAR. When the write operation to the PP that takes the longest to complete finishes, the write operation is complete. Specifying mirrored logical volumes with a parallel scheduling policy might improve I/O read-operation performance because multiple copies allow the system to direct the read operation to the least busy disk for this logical volume.

Parallel write with sequential read scheduling policy

Simultaneously starts the write operation for all the PPs in an LPAR. The primary copy of the read is always read first. If that read operation is unsuccessful, the next copy is read. During the read retry operation on the next copy, the failed primary copy is corrected by the LVM with a hardware relocation, which patches the bad block for future access.

Parallel write with round-robin read scheduling policy

Simultaneously starts the write operation for all the PPs in an LPAR. Reads are switched back and forth among the mirrored copies.

You must set the policy to parallel write with sequential read to get the preferred read performance of the IBM FlashSystem. With this option, you get these functions:

- ▶ Write operations are done in parallel to all copies of the mirror.
- ▶ Read operations are always performed on the primary copy of the devices in the mirror set.

Note: Downtime for the file system is required when you change the LVM scheduler policy.

Mirroring synchronization for allocated Enhanced Journaled File System blocks

While working with mirrored copies, it is a best practice to check the mirroring copies synchronization among the logical volumes copies regardless of the preferred reading copy set or preferred scheduling policy.

The **syncvg** command synchronizes the PPs, which are copies of the original PP that are not current. A new feature is introduced in **syncvg** command that can instruct the **syncvg** command to avoid syncing PPs that are not allocated by the file system. If the Enhanced Journaled File System (JFS2) is mounted, the resync operation of the LVM resynchronizes the blocks that are allocated only by the JFS2.

The **-j** option achieves this operation, and is accompanied either by a **y** or **n** attribute. Because **-jy** resynchronizes the blocks that are allocated only by the JFS2, **-jn** resynchronizes all of the blocks regardless of the JFS2 block allocations, and it is the default value.

1.6.2 LVM reclamation support

In AIX 7.2 7200-01 TL or later, the LVM supports space reclamation for physical volumes (PVs) that can reclaim space. It works as follows:

- ▶ LVM informs the disk driver, which informs the storage subsystem that the partition space is no longer in use and that the storage subsystem can reclaim the allocated space.
- ▶ The disk driver helps LVM detect the space reclamation capability of the PV.
- ▶ LVM and file system configuration commands such as the **rm1v** command, **rm1vcopy** command, and the **chfs** (shrink fs) command initiate the space reclamation for the partitions after they are freed.
- ▶ LVM detects the PV's space reclamation capability when it opens the volume during the execution of the **varyonvg** or **extendvg** command.
- ▶ LVM also tries to detect it while the VG is online.
- ▶ If the state change detection requires a PV to be reopened, the administrator must run the **varyoffvg** command and then run the **varyonvg** command for the VG.

VGs that are created before AIX 7.2 TL 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 is automatically reclaimed for the partitions that are freed after installation of AIX 7.2 TL 1.

The LVM process to reclaim the space runs in the background after a command such as **rm1v** completes execution.

Take note:

- ▶ If the system fails 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 function is available from the storage subsystem to reclaim the freed space from a PV. Each storage subsystem expects the reclaim request to be aligned on a 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 a partition is not possible because the reclaim size is not aligned with the physical blocks of the partition.

Some storage subsystems support the reclamation 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 reclamation 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 -r** command to get information about the space reclamation requests that are generated by the LVM.

Disabling reclamation

The reclamation feature can be disabled by running the **ioo dk_1bp_enabled** command.

You can use the **dk_1bp_enabled** parameter to enable or disable support for the Logical Block Provisioning (LBP) (thin-provisioning) in the AIX OS.

When disabled, AIX does not attempt to release the blocks that are not used from a thin-provisioned disk. A value of 0 disables the LBP support. A value of 1 enables the LBP support. The default value is 1.

For more information, see [IBM Knowledge Center](#).

1.7 JFS2

This section describes the recent enhancements for the JFS2 file system.

1.7.1 JFS2 defragger

If a file system is created with a fragment size smaller than 4 KB, it becomes necessary after a period to query the number 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 the **smitty dejfs2** command or the **defragfs** command, as shown in Example 1-37. Some free space must be available for the defragmentation procedure to be used. The file system must be mounted for read/write.

Example 1-37 The defragfs command

```
# 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.
```

The **defragfs** command attempts to coalesce free space in the file system 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 might result in improved I/O performance when accessing the files. What it does is relocate and combine data extents for each file in the file system. This process prioritizes file organization over file system free space contiguity.

1.7.2 Reclaiming JFS2 space

JFS2 space reclamation is used to return space to the disk to enact thin provisioning. 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 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, you 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 provides only a user command to run this option. The actual work of the reclamation is done by the LVM. This option is supported only on JFS2.

There is a new attribute for the **chfs** command to reclaim unused space in JFS2 for the **-a** option without shrinking the file system:

```
chfs -a reclaim=[normal | fast]
```

If the normal mode is chosen, this command packs the file system as much as possible. Then, it looks for the biggest contiguous chunk of free space and reclaims as much of it as it can. If the fast mode is chosen, this command looks for the biggest contiguous chunk of free space (without first packing the file system) and then reclaims as much of it as it can.

The **chfs** command cannot estimate exactly how much free space is recovered. To get an estimate, run **lvmstat -v VGNAME -r** after running the **chfs** command to provide a rough estimate of how much space was reclaimed.

Points to consider

Consider the following points regarding file system reclamation:

- ▶ Although file system space reclamation frees disk partitions for reuse, it does not alter the actual file system size.
- ▶ File system space reclamation is recommended during periods of low I/O activity.

- ▶ File system space reclamation can be run only on a file system whose disks all support the reclamation operation. If all the disks on a VG support the reclamation operation, the `lvmstat -v VGNAME -r` command displays `on` in the `reclaim` column, as shown in Example 1-38.

Example 1-38 Checking the file system space reclamation 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
```

- ▶ File system reclamation cannot be run while Live Update is running, and Live Update cannot start if this command is in progress.
- ▶ File system reclamation cannot be used on a file system that is read-only.
- ▶ File system reclamation cannot be used concurrently with any file system resize operation like `shrinkfs` or `extendfs`.
- ▶ The amount of free space that is reclaimed depends on the number of free partitions in the file system and the underlying storage's reclaim block size.

For more information, see [IBM Knowledge Center](#).

1.8 Multiple alternative disk clones

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

Cloning the `rootvg` to an alternative disk has many advantages:

- ▶ You have 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.
- ▶ When applying maintenance or TL updates, a copy of the `rootvg` is made to an alternative disk and then updates are applied to that copy. The system runs uninterrupted during this time. When it is restart, the system starts from the newly updated `rootvg` for testing. If the updates cause problems, the `old_rootvg` can be retrieved by resetting the bootlist and then restarting.

1.8.1 Cloning concepts

The `alt_disk_copy` command allows users to copy the current `rootvg` to an alternative disk and to update the OS to the next maintenance or TL without taking down the machine for an extended period, which mitigates outage risk. To do this task, create a copy of the current `rootvg` on an alternative disk and simultaneously apply software updates. If necessary, run the `bootlist` command after the new disk starts. The bootlist can be changed to fall back to the older maintenance or TL of the OS.

If you start by using the new alternative disk, the former `rootvg` VG shows up in a `lspv` command listing as `old_rootvg`, and it includes all disks in the original `rootvg`. This former `rootvg` VG is set to *not* vary-on at restart, and it should be removed only by running the `alt_rootvg_op -X old_rootvg` or `alt_disk_install -X old_rootvg` commands.

Cloning operation procedure

To do the cloning operation, complete the following steps.

1. Create an `/image.data` file that is based on the current rootvg's configuration. A customized `image.data` file can be used.
2. Create an alternative rootvg (`altinst_rootvg`).
3. Create the logical volumes and file systems with the `alt_inst` prefix.
4. Generate a backup file list from the rootvg, and if an `exclude.list` file is present, those files are excluded from the list.
5. Copy the final list to the `altinst_rootvg` file systems.
6. If specified, the `installp` command installs updates, fixes, or new file sets into the alternative file system.
7. The file systems are then unmounted, and the logical volumes and file systems are renamed.
8. 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 the current rootvg is common, and it uses the `alt_disk_copy` command to do the job.

Cloning from the mksysb image is performed by running `alt_disk_mksysb` command.

Creating a single clone from the current rootvg

To clone the rootvg into another disk, complete the following steps:

1. Check the system PVs, as shown in Example 1-39.

Example 1-39 Checking the system physical volumes

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

2. Initiate the cloning process by running the `alt_disk_copy` command to the target `hdisk1`, as shown in Example 1-40.

Example 1-40 Initiating the cloning process to hdisk1

```
# 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

```

3. Checking the system after the clone operation, you see a new VG that is named `altinst_rootvg`, as shown in Example 1-41.

Example 1-41 Checking the new volume group that is created after the clone

```

# lspv
hdisk0      00f6db0a6c7aece5      rootvg      active
hdisk1      00fa2b8bff7af30e      altinst_rootvg
hdisk2      00fa2b8bfff8b56b      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` VG is by using a *wake-up operation*. A VG wake-up can be done on the non-booted VG.

The wake-up operation puts the VG in a `post-alt_disk_install phase1` state, which means that the current root file system remains as `/alt_inst`, which also applies for the rest of the file systems in `altinst_rootvg`.

Note: Any cloned VG that has the wake-up operation ran against is renamed to `altinst_rootvg`. When data access is no longer needed, the VG can be put to sleep. So, if you renamed your clone and woke it up, you must rename with the original name.

Example 1-42 shows how to wake up the new clone and how to put it to sleep. For the wake-up operation, we run `alt_rootvg_op -W`.

Example 1-42 Waking up the clone and putting it to sleep

```

# lspv
hdisk0      00f6db0a6c7aece5      rootvg      active
hdisk1      00fa2b8bff7af30e      altinst_rootvg
hdisk2      00fa2b8bfff8b56b      None
hdisk3      00fa2b8bfff8b600      None

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

# lspv
hdisk0      00f6db0a6c7aece5      rootvg      active
hdisk1      00fa2b8bff7af30e      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_hd3      1.00  1.00  1%     61  1% /alt_inst/tmp
/dev/alt_hd2      4.00  2.12  48%   40625  8% /alt_inst/usr

```

```
/dev/alt_hd9var 2.00 1.93 4% 1120 1% /alt_inst/var
/dev/alt_livedump 0.25 0.25 1% 4 1%
/alt_inst/var/adm/ras/livedump
```

If we changed to the /alt_inst file systems, we can see the data of the altinst_rootvg clone, but we cannot run a command to reflect altinst_rootvg unless we change the root directory temporarily by running the **chroot** command, as shown in Example 1-43. To confirm this change, look for the new shell process in Example 1-43.

Example 1-43 Changing the 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

# exit

# ps
      PID    TTY  TIME CMD
  4587858 pts/0  0:00 -ksh
  7799196 pts/0  0:00 ps

# lsuser -a ALL
root
daemon
bin
guest
newuser
```

After you access altinst_rootvg, you do not find the user newuser that was created in the original rootvg, which means it was created after the clone.

Important notes to consider:

- ▶ The clone installation does not allow a wake-up to occur on a VG with a greater OS version unless the **FORCE** environment variable is set to yes.
- ▶ If a **FORCE** wake-up is attempted on a VG that contains a more recent version of the running OS and the waking VG was a system rootvg, several errors occur.
- ▶ When data access to the clone is no longer needed, the VG should be put to sleep.
- ▶ The VG that experiences a wake-up must be put to sleep before it can be started and used as the rootvg because starting while the VG is awake disrupts the ODM.

Setting the volume into sleep state can be done by running `alt_rootvg_op -S`, as shown in Example 1-44.

Example 1-44 Putting altinst_rootvg to sleep

```
# 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...

# lspv
hdisk0          00f6db0a6c7aece5          rootvg          active
hdisk1          00fa2b8bff7af30e          altinst_rootvg
hdisk2          00fa2b8bff8b56b          None
hdisk3          00fa2b8bff8b600          None
# df -g | grep alt_inst
#
```

Creating another clone from an mksysb image

You can always create another clone either from the current rootvg or from the mksysb image. However, the default operation of cloning uses the same name of `altinst_rootvg`, so if you attempt to create the clone either from the current rootvg or from the mksysb image, the process fails, as shown in Example 1-45.

Example 1-45 Attempting to create another clone from the mksysb image and receiving an error

```
# 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.
```

To fix this error, rename the current `altinst_rootvg` clone and then create your clones by completing the following steps:

1. To rename the current clone, run `alt_rootvg_op -v`, as shown in Example 1-46.

Example 1-46 Renaming the first altinst_rootvg clone

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

# alt_rootvg_op -v 1st_clone -d hdisk1

# lspv
hdisk0      00f6db0a6c7aece5      rootvg      active
hdisk1      00fa2b8bff7af30e      1st_clone
hdisk2      00fa2b8bff8b56b      None
hdisk3      00fa2b8bff8b600      None
```

2. Create a second clone and rename it too, as shown in Example 1-47.

Example 1-47 Creating a second clone and renaming 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 alternative 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      00fa2b8bff7af30e      1st_clone
hdisk2      00fa2b8bff8b56b      altinst_rootvg active
hdisk3      00fa2b8bff8b600      None

# alt_rootvg_op -v 2nd_clone -d hdisk2
# lspv
hdisk0      00f6db0a6c7aece5      rootvg      active
hdisk1      00fa2b8bff7af30e      1st_clone
hdisk2      00fa2b8bff8b56b      2nd_clone
hdisk3      00fa2b8bff8b600      None
```

3. The second clone is successfully created, and there is no limitation for the number of instances that can be created, and there is no conditions regarding from which source the clone is created.

While creating multiple clones, consider the following points:

- ▶ You cannot wake up two rootvgs concurrently, but you can create as many clones as you need.
- ▶ The running OS must be a higher version or equal to the OS version of the VG that undergoes the wake-up operation.

For more information, see the following resources:

- ▶ [Multiple Alternate rootvg Criteria](#)
- ▶ [Managing multiple instances of altinst_rootvg and applying fixes to them](#)

1.9 Active Memory Expansion

Active Memory Expansion (AME) is a feature that is available on Power Systems servers that you can use to expand the amount of memory that is available to an AIX LPAR beyond the limits that are specified in the partition profile. AME compresses memory pages to provide more memory for a partition.

AME is available starting with IBM POWER7® processor-based systems and AIX V6.1 TL 4 Service Pack 2. At the time of writing, AIX is the only OS that can use this feature. In-memory data compression is managed by the OS, and this compression is transparent to applications and users.

Compression and decompression of memory content can provide memory expansion with percentages that can exceed 100%. So, a partition can 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 that are allocated to the partition to compress and decompress the memory contents of the partition. AME provides a tradeoff 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 a smaller amount of memory. AIX has a command that is named **amepat** (AME Planning and Advisory Tool) that you can use to estimate the compression rate for an individual workload.

For more information about **amepat**, see [IBM Knowledge Center](#).

In the initial implementation of AME, only 4 K and 16 million memory pages are subject to compression. The following types of memory page are not compressed:

- ▶ Pinned memory pages.
- ▶ File pages that are cached in memory.
- ▶ Pages that are already compressed.

A new unrestricted **vmo** tunable was added to enable all supported page sizes. Here is its syntax:

```
vmo -ro ame_mpsize_support=1
```

Starting with AIX 7.2 TL 1, compression of 64 K memory pages is available on IBM POWER8® processor-based systems with firmware FW860 or later.

For more information about AME, see the following resources:

- ▶ *IBM PowerVM Virtualization Managing and Monitoring*, SG24-7590
- ▶ *IBM PowerVM Virtualization Introduction and Configuration*, SG24-7940
- ▶ [IBM Knowledge Center](#)

1.10 The nmon tool and current processor frequency reporting

The ability of Power Systems servers to change processor frequencies depending on system configuration settings and workload characteristics to improve energy efficiency and system performance was introduced by the EnergyScale technology for IBM POWER6™ processor-based systems. With each new processor generation, the EnergyScale technology is enhanced and expanded. Particularly for the Maximum Performance mode with POWER9 processor-based systems, the need for a convenient processor frequency reporting tool became indispensable. To put the related enhancement to the `nmon` tool in perspective, a brief overview of the processor frequency control policies for POWER8 and POWER9 processor-based systems might help you to better understand the relevance of current processor frequency reporting in your data center environment.

EnergyScale functions that control processor frequency policies on POWER8 processor-based systems are as follows:

| | |
|--------------------------------------|---|
| Idle Power Saver mode | When a system is considered to be idle, the processor frequency is reduced to the supported minimum level. |
| Fixed Maximum Frequency mode | The system runs at its maximum frequency regardless of workload if the power and thermal limits allow it. |
| Static Power Saver mode | Provides predictable performance with power savings by reducing the CPU frequency by a fixed amount. |
| Fixed Frequency Override mode | Allows a user to specify a specific frequency between the minimum and maximum values that are allowed for the system to run at. |
| Dynamic Power Saver mode | Allows a system to apply algorithms to dynamically adjust the processor core frequency to optimize system performance (<i>favor performance</i>) or to balance energy consumption and performance (<i>favor power</i>). |

If the Dynamic Power Saver mode is enabled through the Asynchronous Systems Management Interface (ASMI) the processor frequency changes dynamically over time in response to the system's workload characteristics.

Note: POWER8 processor-based systems run by default with a constant nominal frequency. On some server models, the Idle Power Saver mode is enabled by default. But this mode determines only the minimum frequency that is applied if the system runs idle. All other modes must be explicitly enabled by the system administrator.

The announcement of POWER9 processor-based systems introduced new features for EnergyScale, including new variable processor frequency modes that provide a significant performance boost beyond the static nominal frequency. The following EnergyScale functions are available for POWER9 processor-based systems to control processor frequencies:

- | | |
|---------------------------------|---|
| Idle Power Saver mode | When a system is considered to be idle, the processor frequency is reduced to the supported minimum level. |
| Static Power Save mode | The system runs at the minimal static frequency all the time regardless of the workload. |
| Static Nominal mode | The system runs at the fixed nominal frequency. The nominal frequency is the guaranteed frequency that the system achieves when running within the specified environmental parameters for a server model (meaning under the maximum ambient temperature and elevation). This mode is enabled when selecting Disable all modes in the Power and Performance Mode Setup menu of the ASMI tool. The option of Disable all modes was the default for all systems before POWER9 processor-based systems. While this mode provides a constant frequency, it does not maximize the performance of the system. |
| Dynamic Performance mode | The system generally runs above the nominal frequency and might even get to the maximum frequency if the workloads are light enough or many cores are not being used. The determining factor for what frequency the CPU runs at in Dynamic Performance mode is power. The system limits the socket power draw to a base wattage (this varies somewhat by chip and system). When in Dynamic Performance mode, the frequency varies above the static nominal value depending on available power headroom within the limits that are imposed by the nominal power draw of the sockets. |
| Maximum Performance mode | This mode takes advantage of lower active core counts and normal utilization workloads (just like Dynamic Performance mode). However, Maximum Performance mode allows the system to reach the maximum frequency under more conditions, thus providing maximum performance. As in the Dynamic Performance mode, the same constraint of power exists, but the system takes extra steps to extend the limit. When in Maximum Performance mode, the voltage regulators allow the sockets to draw more power than in any of the other modes. The Maximum Performance mode may, if necessary, increase the speed of the server air-moving devices (fans) to remain at the highest frequency under higher CPU workloads. This potentially leads to a higher acoustic level in the data center or office environment. |

Note: The Maximum Performance mode is the default for all POWER9 processor-based systems except for the IBM Power System S914 server. (The default for the Power S914 system is the Dynamic Performance mode.) The frequencies that are used by a POWER9 processor-based system in the Maximum Performance mode is published as the typical frequency range specification for each server.

One method for measuring the fluctuating core frequency in AIX is to use the `mpstat -E 1 1` or the `lparstat -E 1 1` command. The `mpstat -E 1 1` command provides individual core frequencies, and `lparstat -E 1 1` averages frequencies across all the cores in the LPAR.

Beginning with AIX 7.2 TL 4, the `nmon` tool reports the static nominal processor frequency as *CPU speed* (in MHz) and the current processor frequency as *CPU-Current-Speed* (in MHz) value. If you use `nmon` in text screen, type the keyword `r` to get access to the Resources submenu where the processor frequency-related values are listed as shown in Figure 1-4. The CPU-Current-Speed metric is updated dynamically whenever the system changes the processor frequency according to the EnergyScale performance and efficiency optimization algorithm.

```

-topas nmon -c=CPU Host=vm206 Refresh=2 secs 11:42.12
Resources
OS has 24 PowerPC POWER9 (64 bit) CPUs with 24 CPUs active SMT=8
CPU Speed 3300.0 MHz SerialNumber=7804930 MachineType=IBM,9009-42A
Logical partition=Dynamic HMC-LPAR-Number&Name=16,vm206-d20127ba-00000076
AIX Version=7.2.4.0 TL04 Kernel=64 bit Multi-Processor
Power Saving=Dynamic-Performance CPU-Current-Speed 3937.4 MHz
Hardware-Type (NIM)=CHRP=Common H/W Reference Platform Bus-Type=PCI
CPU Architecture =PowerPC Implementation=POWER9 SubProcessor Mode=Unknown
CPU Level 1 Cache is Combined Instruction=32768 bytes & Data=32768 bytes
Level 2 Cache size=not available Node=vm206
Event= 0 --- --- SerialNo Old=--- Current=C04930 When=---

```

Figure 1-4 The `nmon` Resources screen showing the CPU-Current-Speed metric

To save the current processor frequency to the `nmon` output file (`.nmon`), system administrators must add the parameter `-y dfreq=on` to the `nmon` command-line arguments.

Example 1-48 shows a `nmon` command or crontab entry that initiates the collection of processor frequency metrics, including the CPU-Current-Speed.

Example 1-48 The `nmon` command to collect processor frequency metrics

```
nmon -f -m /home/nigel/nmondata -s 60 -c 1440 -y dfreq=on
```

The processor frequency metrics that are collected in the output file by the `-y dfreq=on` command argument are shown in Example 1-49. The open source graphing tools like `nmon Analyzer` and `nmonchart` were updated to graph these new processor frequency statistics when they are found in the `nmon` output file.

Example 1-49 Processor frequency-related statistics that are collected by the `-y dfreq=on` `nmon` argument

```

# grep ^CPUMHZ vm206_191113_1146.nmon
CPUMHZ,CPU Frequency, nominalMHz, currentMHz
CPUMHZ,T0001,3300.0,3937.9
CPUMHZ,T0002,3300.0,3937.2
CPUMHZ,T0003,3300.0,3937.3
CPUMHZ,T0004,3300.0,3937.5
CPUMHZ,T0005,3300.0,3937.4
CPUMHZ,T0006,3300.0,3937.2
CPUMHZ,T0007,3300.0,3937.2
CPUMHZ,T0008,3300.0,3937.2
CPUMHZ,T0009,3300.0,3937.2
[ ... omitted lines ... ]

```

1.11 Globalization

One important aspect of a modern OS relates to an up-to-date implementation of globalization features. AIX is committed to this strategic development imperative and provides regular updates and enhancements in the following globalization areas:

- ▶ Unicode standard
- ▶ Common Locale Data Repository (CLDR) definition
- ▶ International Components for Unicode (ICU) for C libraries
- ▶ New locale support

The following sections give an overview of the AIX enhancements in each of the named areas regarding the AIX 7.1 and AIX 7.2 TLs since the introduction of AIX Version 7.2.

1.11.1 Unicode support

Unicode is a code set standard for supporting worldwide information processing requirements, and it is used to display all languages throughout the world. In AIX, Unicode is a foundational element of the OS because it is used as internal data exchange standard for all data processing operations. Therefore, it is a best practice to keep the AIX OS Unicode-compliant and provide required enhancements and additions with every TL update.

Table 1-4 shows the supported Unicode version regarding the AIX 7.1 and AIX 7.2 TLs since the introduction of AIX Version 7.2 in December 2015.

Table 1-4 Unicode release support and history for AIX 7.2 and related AIX 7.1 releases

| Unicode version | AIX 7.2 TL | AIX 7.1 TL |
|-----------------|------------|------------|
| 7.0 | 7200-00 | 7100-04 |
| 8.0 | 7200-01 | NA |
| 9.0 | 7200-02 | NA |
| 10.0 | 7200-03 | NA |
| 11.0 | 7200-04 | NA |

Unicode 7.0 in AIX 7.2 and AIX 7.1 TL 4

In December 2015, the AIX 7.2 base release and AIX 7.1 TL 4 introduced support for Unicode V7.0. With Version 7.0, the Unicode Consortium defined 23 new scripts, 2834 new code points, and many new symbols, including more than 100 emoji.

The following list shows the newly supported scripts that are provided in AIX 7.1 with 7100-04 TL and in AIX 7.2 with 7200-00 base level through Unicode 7.0:

- ▶ Bassa Vah
- ▶ Caucasian Albanian
- ▶ Duployan
- ▶ Elbasan
- ▶ Grantha
- ▶ Khojki
- ▶ Khudawadi
- ▶ Linear A
- ▶ Mahajani
- ▶ Manichaean
- ▶ Mende Kikakui

- ▶ Modi
- ▶ Mro
- ▶ Nabataean
- ▶ Old North Arabian
- ▶ Old Permic
- ▶ Pahawh Hmong
- ▶ Palmyrene
- ▶ Pau Cin Hau
- ▶ Psalter Pahlavi
- ▶ Siddham
- ▶ Tirhuta
- ▶ Warang Citi

Among the more than 100 emoji, you find the prominent *slightly smiling face* and the *slightly frowning face*. For examples of the graphical representation of any of the emojis in Unicode 7.0, see [Unicode Version 7](#).

For more information regarding Unicode 7.0, see [Unicode 7.0 Versioned Charts Index](#).

Unicode 8.0 in AIX 7.2 TL 1

In November 2016, AIX 7.2 TL 1 introduced support for Unicode 8.0. With Version 8.0, the Unicode Consortium defined six new scripts, 7716 new code points, 37 new emoji, and 5 skin tone emoji modifiers.

The following list shows the six newly supported scripts that are provided in AIX 7.2 7200-01 TL through Unicode 8.0:

- ▶ Ahom
- ▶ Anatolian Hieroglyphs
- ▶ Hatran
- ▶ Multani
- ▶ Old Hungarian
- ▶ Sutton SignWriting

Among the 37 emoji, you find the well-known *thinking face* and the *face with rolling eyes*. Examples of the graphical representation of any of the emojis in Unicode 8.0 can be found at [Unicode Version 8](#).

For more information regarding Unicode 8.0, see [Unicode 8.0.0](#).

Unicode 9.0 in AIX 7.2 TL2

In October 2017, AIX 7.2 TL 2 introduced support for Unicode 9.0. With Version 9.0, the Unicode Consortium defined six new scripts, 7500 new code points, and 72 new emoji.

The following list shows the six newly supported scripts that are provided in AIX 7.2 7200-02 TL through Unicode 9.0:

- ▶ Adlam
- ▶ Bhaiksuki
- ▶ Marchen
- ▶ Newa
- ▶ Osage,
- ▶ Tangut

Among the 72 new emoji, you find the *rolling on the floor laughing face* and the *lying face*, and symbols for the avocado and potato. Examples of the graphical representation of any of the emojis in Unicode 9.0 can be found at [Unicode Version 9](#).

For more information regarding Unicode 9.0, see [Unicode 9.0.0](#).

Unicode 10.0 in AIX 7.2 TL3

In November 2018, AIX 7.2 TL 3 introduced support for Unicode 10.0. With Version 10.0, the Unicode Consortium defined four new scripts, 8518 new code points, and 56 new emoji.

The following list shows the four newly supported scripts that are provided in AIX 7.2 7200-03 TL through Unicode 10.0:

- ▶ Masaram Gondi
- ▶ Nushu
- ▶ Soyombo
- ▶ Zanabazar Square

Among the 56 new emoji, you find the *face with raised eyebrow* (Colbert) emoji, the Bitcoin symbol, and symbols for broccoli and the Tyrannosaurus Rex. Examples of the graphical representation of any of the emojis in Unicode 10.0 can be found at [Unicode Version 10](#).

For more information regarding Unicode 10.0, see [Unicode 10.0.0](#).

Unicode 11.0 in AIX 7.2 TL4

In November 2019, AIX 7.2 TL 4 introduced support for Unicode 11.0. With Version 11.0, the Unicode Consortium defined seven new scripts, 684 new code points, and 66 new emoji.

The following list shows the seven newly supported scripts that are provided in AIX 7.2 7200-04 TL through Unicode 11.0:

- ▶ Dogra
- ▶ Gunjala Gondi
- ▶ Hanifi Rohingya
- ▶ Makasar
- ▶ Medefaidrin
- ▶ Old Sogdian
- ▶ Sogdian

Among the 66 emoji, you find the *partying face* and the *woozy face* but also symbols for a superhero and a raccoon. Examples of the graphical representation of any of the emojis in Unicode 11.0 can be found at [Unicode Version 11](#).

For more information regarding Unicode 11.0, see [Unicode 11.0.0](#).

1.11.2 Common Locale Data Repository updates

CLDR local definitions are promoted by the Unicode CLDR project for consistent culture conversions in the context of software globalization. The AIX OS supports the CLDR standard and updates the local definitions regularly in accordance with the latest CLDR releases.

Table 1-5 shows the supported CLDR releases regarding the AIX 7.1 and AIX 7.2 TLs since the introduction of AIX Version 7.2 in December 2015.

Table 1-5 CLDR release support and history for AIX 7.2 and related AIX 7.1 releases

| CLDR release | AIX 7.2 TL | AIX 7.1 TL |
|--------------|------------|------------|
| 26.0 | 7200-00 | 7100-04 |
| 29.0 | 7200-01 | NA |
| 30.0.3 | 7200-02 | NA |
| 32.0.1 | 7200-03 | NA |
| 34.0.0 | 7200-04 | NA |

For more information about the Unicode CLDR project, see [Unicode CLDR Project](#).

1.11.3 International Components for Unicode for C

ICU is the premier software globalization package. It provides robust features that allow programmers to effectively work with Unicode data and create globalized applications. In this context, ICU4C is a set of C/C++ libraries that provide Unicode and globalization standards-compliant text handling services in support for software applications.

ICU is part of the strategic imperatives of AIX to provide customers with a current version of the ICU4C libraries. To achieve this goal, AIX development follows tightly the release activities of the ICU organization.

Table 1-6 shows the supported ICU4C versions regarding the AIX 7.1 and AIX 7.2 TLs since the introduction of AIX Version 7.2 in December 2015.

Table 1-6 ICU4C release support and history for AIX 7.2 and related AIX 7.1 releases

| ICU4C version | AIX 7.2 TL | AIX 7.1 TL |
|---------------|------------|------------|
| 55.1 | 7200-00 | 7100-04 |
| 57.1 | 7200-01 | NA |
| 58.2 | 7200-02 | NA |
| 60.2 | 7200-03 | NA |
| 63.1 | 7200-04 | NA |

For more information about the ICU project, see [ICU-TC Home Page](#).

1.12 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 by using the RPM package manager. It also contains popular open source libraries and software that are commonly used in Linux based distributions. It also comes with GNU tools that are required to build and package Linux applications to use on AIX. The software that is provided in this repository comes *as-is* and without support.

For an overview of the AIX Toolbox for Linux Applications and licensing and installation details, see [AIX Toolbox for Linux Applications](#).

Note: Using the AIX Toolbox for Linux Applications repositories and other repositories might result in dependency issues or other unexpected errors.

To simplify the installation and usage of the packages that are available on AIX Toolbox for Linux Applications, AIX may use the YUM package management utility to access the toolbox. The YUM package manager handles the management of dependencies and installs the prerequisite packages for you so you do not need to resolve them manually. For more information about how to install and configure YUM on AIX to access the AIX Toolbox for Linux Applications, see [the readme file for YUM](#).

If your AIX partition is connected to the internet, the process involves downloading and running the `yum.sh` script that is available in the AIX Toolbox, as shown in Example 1-50.

Example 1-50 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                ##### [ 88%]
16:python-devel-2.7.10-1     ##### [ 94%]
17:python-tools-2.7.10-1    ##### [100%]

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
```

After the YUM repositories are configured and installed, you can quickly install any package that is available on the repository by running `yum install <package name>`.



Security enhancements

This chapter describes recent IBM AIX security enhancements.

This chapter describes the following topics:

- ▶ AIX Trusted Execution
- ▶ AIX Secure boot
- ▶ Multifactor authentication
- ▶ Cryptographic libraries
- ▶ Address space layout randomization

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 the TE mechanism, the system can be configured to check the integrity of the trusted object, such as:

- ▶ A command
- ▶ A binary file
- ▶ A library
- ▶ A configuration file
- ▶ A shell script in a 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 with some security critical files, which results in the system becoming vulnerable and exploitable. The central idea behind the set of features under TE is prevention of such activities, or in the worst case to identify whether any such incident happens to the system.

Using the functions that are provided by TE, the system administrator can decide on the actual set of executable files that are allowed to run or the set of kernel extensions that are allowed to be loaded. You can also use TE to audit the security state of the system and identify files that changed, which increases the trusted level of the system and makes 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 the Trusted Signature Database (TSD)
- ▶ Auditing integrity of the TSD
- ▶ Configuring Security Policies
- ▶ Trusted Execution Path (TEP) and Trusted Library Path (TLP)

Similar to the Trusted Computing Base (TCB), there is a database that stores critical security parameters of trusted files that are on the system. This database, which is named *Trusted Signature Database (TSD)*, is in the `/etc/security/tsd/tsd.dat` file.

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

- ▶ Kernel (operating system (OS)).
- ▶ 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 `1s` 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 that is stored in the TSD. A file can be marked as trusted by adding its definition to the TSD by 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 TSD), the TE feature can monitor its integrity for every access. TE can continuously monitor the system and can detect tampering on any trusted file (by a malicious user or application) that is present on the system at run time (for example, at load time). If the file is found to be tampered with, TE can take corrective actions based on pre-configured policies, such as disallow execution or access to the file, or log an error.

To enable or disable different security policies that are used with the TE mechanism, use the **trustchk** command. You can specify the policies that are shown in Table 2-1.

Table 2-1 Trusted Execution different policies

| Policy | STOP_UNTRUSTED | STOP_ON_CHKFAIL |
|------------|--|--|
| CHKEXEC | Stops the loading of executable files that do not belong to TSD. | Stops the loading of executable files whose hash values do not match the TSD values. |
| CHKSHLIBS | Stops loading of shared libraries that do not belong to TSD. | Stops loading of shared libraries whose hash values do not match the TSD values. |
| CHKSCRIPTS | Stops loading of shell scripts that do not belong to TSD. | Stops loading of shell scripts whose hash values do not match the TSD values. |
| CHKKERNEXT | Stops loading of kernel extensions that do not belong to TSD. | Stops loading of kernel extensions whose hash values do not match the TSD values. |

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

The TEP defines a list of directories that contain the trusted executable files. After TEP verification is enabled, the system loader allows only binary files in the specified paths to run.

The TLP has the same function except that it is used to define the directories that contain trusted libraries of the system. After TLP is enabled, the system loader allows only the libraries from this path to be linked to the binary files.

The **trustchk** command can be used to enable or disable the TEP or TLP and set the colon separated path list for both by using TEP and TLP command-line attributes for the **trustchk** command.

In order for TE to work, the CryptoLite for C (CLiC) and kernel extension must be installed and loaded on your system. These file sets are included with the AIX Expansion Pack and are provided at no additional charge.

To check whether these items are installed on your system and loaded into the kernel, run the **lslpp** command to check, and then run the **genkex** or **lke** internal **kdb** command to ensure that the clicxext kernel extension is loaded into the system, as shown in Example 2-1.

Example 2-1 Checking the CLiC and kernel extension availability in the system

```
# lslpp -l "cllc*"
  Fileset                                Level  State      Description
  -----
Path: /usr/lib/objrepos
  clic.rte.kernext                       4.10.0.2  COMMITTED  CryptoLite for C Kernel
  clic.rte.lib                            4.10.0.2  COMMITTED  CryptoLite for C Library

Path: /etc/objrepos
  clic.rte.kernext                       4.10.0.2  COMMITTED  CryptoLite for C Kernel
  clic.rte.lib                            4.10.0.2  COMMITTED  CryptoLite for C Library

# genkex | grep clic
f1000000c021f000    55000 /usr/lib/drivers/crypto/clicxext

# echo "lke" | kdb | grep clic
59 F1000905DEA00 F1000C021F000 00055000 02090252 /usr/lib/drivers/crypto/clicxext
```

Here is an example of how to turn on the runtime integrity check by configuring TE, CHKEEXEC, and STOP_ON_CHKFAIL. Complete the following steps:

1. Check the status of the policies, turn *on* these policies, and then check the status again, as shown in Example 2-2.

Example 2-2 Checking the TE status and enabling it

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

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

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

2. If you run the **ls** command, it should run without problems because the entry is in the TSD database and no changes to its checksum occurred, so the command execution is permitted, as shown in Example 2-3.

Example 2-3 Permitting the ls command according to the TSD database

```
# ls
.SPOT                bin                mnt
.Xdefaults           bosinst.data      ok2
.bash_history        bxpup.sh          opt
.bash_profile        cd0               perz1images
.bashrc              core              proc
.dbxhist             data              sample.sh
```

| | | |
|--------|-----|----------|
| .kshrc | db | sbin |
| .mwmrc | dev | smit.log |

3. Replace the `/usr/bin/ls` binary file with the `/usr/bin/chmod` binary file. Then, run the `/usr/bin/ls` command. This time, the execution fails because it is not trusted any more, as shown in Example 2-5.

Example 2-4 New untrusted ls command not running

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

# ls
ksh: ls: 0403-006 Execute permission denied
```

The runtime integrity check is working on the system. If there are any trojans or other viruses that changed any binary file, then their run times are denied.

4. To disable the runtime integrity check, run the same command with the OFF value, as shown in Example 2-5.

Example 2-5 Disabling the TE integrity check

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

# /usr/sbin/trustchk -p TE CHKEEXEC STOP_ON_CHKFAIL
TE=OFF
CHKEEXEC=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          perzimages
.bashrc       core         proc
.dbxhist      data         sample.sh
.kshrc        db           sbin
.mwmrc        dev          smit.log
```

The TE can be used in the real-time execution of the trusted commands. Whenever a trusted command is entered at the command line, a hash is calculated by the OS and the hash of the trusted command is extracted from the TSD database. Only when both hash values match may the command run.

2.2 AIX Secure boot

The current security guidelines of the Common Criteria (CC) Protection Profile for General Purpose Operating Systems (GPOSPP) Version 4.2.1 of the National Information Assurance Partnership (NIAP) recommend that OSs must verify the integrity of the bootchain up through the OS kernel (and potentially other executable code) before its execution by using either a digital signature that uses a hardware-protected asymmetric key or by using a hardware-protected hash. This security requirement is described in section “FPT_TST_EXT.1 Boot Integrity” in [Protection Profile for General Purpose Operating Systems](#).

IBM POWER9 processor-based hardware and firmware includes new PowerVM features to provide a more secure platform for cloud deployments. Specifically, a new secure initial program load (IPL) process, commonly referred to as IBM PowerVM Secure Boot, allows only appropriately signed firmware components to run on the system processors. Each component of the firmware stack, including HostBoot, the IBM POWER® Hypervisor (PHYP), and the Partition Firmware (PFW) is signed by the platform manufacturer and verified as part of the IPL process.

The AIX Secure boot feature was implemented in AIX 7.2 Technology Level (TL) 3, and it extends the chain of trust that is initialized by PowerVM Secure Boot to AIX client logical partitions (LPARs) by digitally verifying the AIX codes: the OS boot loader, the kernel, and the runtime environment, including device drivers, kernel extensions, applications, and shared libraries. The AIX Secure boot feature meets the boot integrity requirements that are defined by GPOSPP Version 4.2.1.

Before more details about the AIX Secure boot feature are explained in 2.2.2, “AIX Secure boot implementation” on page 69, a brief overview about the underlying POWER9 secure IPL technology is provided in 2.2.1, “PowerVM Secure Boot” on page 68.

2.2.1 PowerVM Secure Boot

PowerVM Secure Boot on POWER9 processor-based systems implements a host processor-based chain of trust. The chain starts with an implicitly trusted component with other components being authenticated and integrity checked before being run on the host processor cores. A verification code from the locked processor Serial Electrically Erasable Programmable ROM (SEEPROM) validates the initial firmware load. The firmware verifies the cryptographic signatures of all subsequent “to be trusted” firmware that is loaded for execution on the POWER9 cores. On a POWER9 processor-based system, SEEPROM security switches are set in Self-Boot Engine (SBE) code and locked down on the system manufacturing line to provide the basis for the hardware enforcement of the secure IPL flow. The entire trusted firmware stack is authenticated by using signed images and runs in trusted memory locations.

Figure 2-1 shows the boot flow for a PowerVM Secure Boot followed by an AIX Secure boot process.

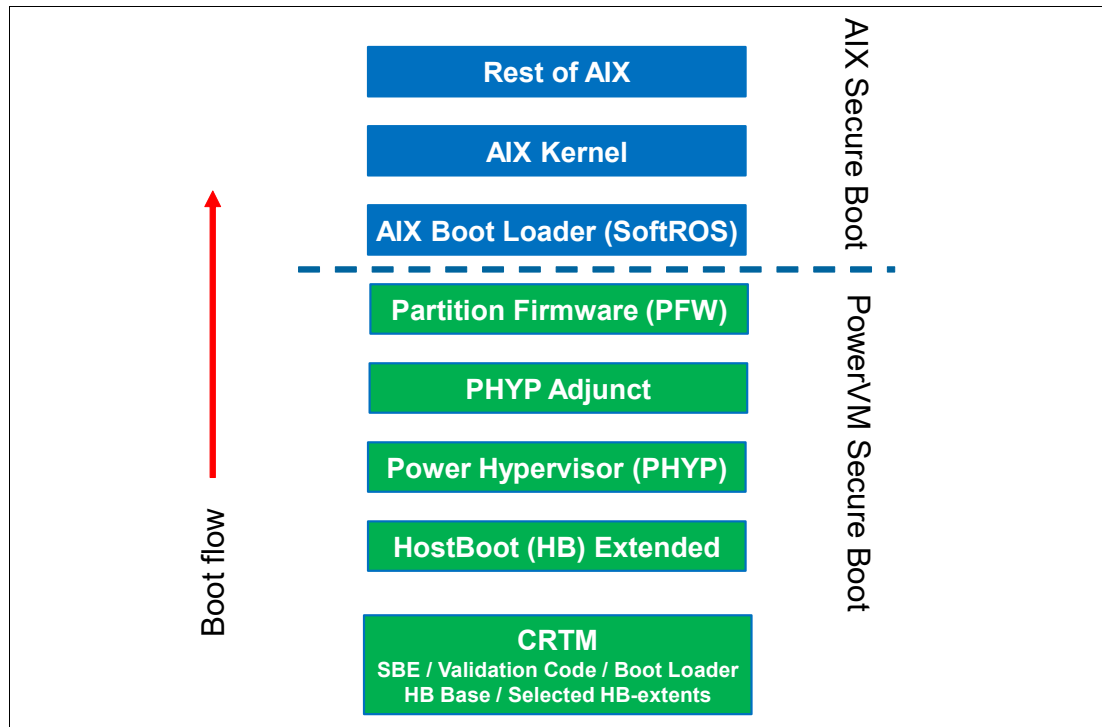


Figure 2-1 PowerVM Secure Boot and AIX Secure boot flow

PowerVM Secure Boot establishes the SBE, verification codes, a mini-boot loader, HostBoot Base code (including a portion of HostBoot Extended code), and hashes of platform / hardware public keys as the Core Root of Trust for Measurement (CRTM), which is saved in the system's SEEPROM module. Initiated by the Flexible Service Processor (FSP), the SBE starts fetching initialization code from on-module secure non-volatile locked memory. With this step, the chain of trust starts and is extended to include more HostBoot Extended code, the PHYP, selected adjunct partitions (physical Trusted Platform Module (pTPM), virtual Trusted Platform Module (vTPM), HostBoot run time, and encryption adjuncts) and the PFW. Coupled with the processor hardware security support, this trust domain ensures that it is not feasible to silently display or alter customer data through any hardware or firmware mechanisms. If the verification fails, the system immediately stops under hardware control to prevent running untrusted or rogue code.

With the successful verification and running of the PFW, the PowerVM Secure Boot process completes and the system is ready to leave the secure IPL domain and start the AIX Secure boot process.

2.2.2 AIX Secure boot implementation

The starting point of AIX Secure boot is running the PFW whose integrity and authenticity has been verified as good. The PFW can be viewed as first stage boot loader on a POWER9 processor-based system. The following steps take place:

1. The PFW loads microcode drivers from the bootable devices, and then loads the OS boot loader and jumps to the entry point of the boot loader.
2. In the secure boot mode, both microcode drivers and boot loader are validated before they are allowed to run.

3. The boot loader decompresses and verifies the kernel. After successful validation, the AIX boot loader loads and starts the kernel.
4. The kernel performs various initialization and setup operations for the processors, virtual memory management, interrupt processing, the RAM file system, the scheduler, and others.
5. The kernel starts the first user space process, which is the init process. The init process is responsible for starting the rest of system run time, which includes loading drivers and configuring devices, mounting the real root file system, and other tasks.
6. All relevant codes, including the kernel; the init process and other applications; the kernel extensions; device drivers; and shared libraries, are verified before being loaded into memory and run.
7. The AIX Secure boot feature checks the integrity of the boot and initialization codes until all entries of the inittab file are parsed and run.

To support the AIX Secure boot process, the AIX boot image was modified to include the digital signatures of the boot loader and the kernel. Also, a subset of the TSD was added to the RAM file system of the boot image to enable signature and integrity verification for all components that are loaded after the kernel initialization.

The TSD traditionally supports the AIX TE feature. When enabled, the kernel execution logic and program loader check at run time the hash of a given binary file, script, shared library, or kernel extension before the respective object is loaded into memory. The hash value is compared against the TSD, which is set up during software installation. The hash and signature of essential AIX applications are computed in the build process and included in each package. If AIX TE is configured, it is enabled after the first boot phase completes.

To implement AIX Secure boot, the kernel is enhanced to support cryptographic functions, and a tailored TSD in the RAM file system enables the TE function at the earliest possible stage of the boot process, particularly before the init process starts. During the secure boot process, the TE logic checks the code integrity by viewing the associated hash and validates the related signature. If the secure boot flow reaches the point where the RAM file system is replaced by the regular file systems, the untailed TSD is reloaded from the /usr file system to the kernel security tables.

By using a combination of AIX Secure boot and AIX TE, it is possible to build an end-to-end security domain that protects the OS effectively from any malicious code.

AIX Secure boot is compatible with other Power Systems specific technologies, like PowerVM Trusted Boot, Live Partition Mobility (LPM), AIX Live Update, and Simplified Remote Restart (SRR).

2.2.3 AIX Secure boot policies and controls

AIX Secure boot is controlled by a partition-specific attribute that is configurable only by using the GUI or the command-line interface (CLI) of the Hardware Management Console (HMC). Five different secure boot policies are supported:

- | | |
|-----------------|--|
| Policy_0 | Disabled: Policy 0 disables AIX Secure boot and designates the default configuration setting. |
| Policy_1 | Enabled: Policy 1 enables AIX Secure boot in audit mode. If errors are encountered, they are logged in the <code>/var/adm/ras/secureboot.log</code> file, but the boot process is not interrupted. |

- Policy_2** Enforced: Policy 2 enforces AIX Secure boot, which means that the boot process is stopped and the LED code 0x328 is shown on the HMC if any signature verification failed.
- Policy_3** Augmented enforcement (I): Policy 3 augments Policy 2 with two more rules: Programs and libraries that are not included in the TSD are not loaded, and write access to /dev/*mem devices is disabled.
- Policy_4** Augmented enforcement (II): Policy 4 augments Policy 3 by disabling the kernel debugger (KDB).

The HMC **chsyscfg** command sets the secure boot policy by using the newly introduced **secure_boot** parameter. The numerical policy designations (0, 1, 2, 3, and 4) are used to qualify the **secure_boot** parameter. Using the parameter requires a restart of the partition for any secure boot policy change to take effect. Example 2-6 shows how a system administrator might configure an enforced AIX Secure boot for LPAR fvtzep2-lp16 on the managed system fctzep2 by using the **chsyscfg** command.

Example 2-6 Enforcing AIX Secure boot by using the chsyscfg HMC command

```
hscroot> chsyscfg -r lpar -m fvtzep2 -i "name=fvtzep2-lp16, secure_boot=2"
```

The HMC **lssyscfg** command shows the currently active secure policy setting as value of the **curr_secure_boot** parameter. The **pend_secure_boot** parameter value that is also shown by the **lssyscfg** command indicates the secure boot policy that is enforced during the next partition boot.

With the HMC GUI, you can configure the secure boot policy by using the General Properties menu of a defined partition. You can select only policy 0 (**Disabled**), 1 (**Enabled and Log only**), or 2 (**Enabled and Enforced**).

From the command-line shell of the AIX OS domain, the secure boot policy setting can be displayed but not changed. The secure boot policy is shown by the **secure_boot** attribute value of the sys0 system device, as shown in Example 2-7. In this example, the partition is running with secure boot disabled.

Example 2-7 The secure_boot attribute of the sys0 kernel device

```
# lsattr -E -l sys0 -a secure_boot -H
```

| attribute | value | description | user_settable |
|-------------|--------------------|------------------|---------------|
| secure_boot | Policy_0(Disabled) | Secure Boot Mode | False |

2.3 AIX trusted installation and update

The current security guidelines of the CC GPOSPP Version 4.2.1 of the NIAP recommend that OSs must cryptographically verify updates to themselves by using a digital signature before installation. This security requirement is detailed in “FPT_TUD_EXT.1.2 Trusted Update” in [Protection Profile for General Purpose Operating Systems](#).

Code (package) signing and verification is the process of digitally signing a program executable file, script, or package to ensure that the software the system is running has not been altered since it was signed. Code signing helps protect against corrupted artifacts, process breakdown (accidentally delivering the wrong item), and any malicious intent.

Code signing and verification works as follows:

1. In addition to writing the code, the author runs a hash function with the code as the input, which produces a digest.
2. The digest is signed with the author's private key, which produces the signature.
3. The code, signature, and hash function are delivered to the verifier.
4. The verifier produces the digest from the code by using the same hash function, and then uses the public key to decrypt the signature. If both digests match, then the verifier can be confident that the code has not been tampered with.

This section explains how this concept applies to signing AIX packages with a digital signature and verifying the packages during an OS installation and update process beginning with AIX 7.2 TL 4:

- ▶ Section 2.3.1, “Digital signature and package signing” on page 72 section provides an introduction to the digital signing technology and its applicability to the AIX OS.
- ▶ Section 2.3.2, “AIX package signing and digital signature catalog definition” on page 73 provides more detailed information about the implementation specifics. Concepts regarding the security requirements (policies) and the related *digital signature catalog* (DSC) are explained. The definition of the relevant AIX Object Data Management (ODM) classes for AIX trusted installation and update are also provided.
- ▶ Section 2.3.3, “AIX digital signature catalog signing process” on page 78 provides an outline of the required process steps that are instantiated by the IBM AIX development organization to support this new security feature.
- ▶ Section 2.3.4, “Signature validation during AIX installation and update process” on page 79 show the process steps that must be run by the AIX OS.
- ▶ Section 2.3.5, “AIX trusted installation and update controls” on page 79 covers practical aspects that are important to system administrators.

2.3.1 Digital signature and package signing

This section frames the context around what a digital signature entails and what level of asset assurance it provides. Specifically, we describe the purpose of package signing in AIX and the levels of assurance that are covered by the product component.

Digital signatures are like electronic “fingerprints”. In the form of a coded message, the digital signature securely associates a signer with a document in a recorded transaction. Digital signatures use a standard accepted format that is called Public Key Infrastructure (PKI) to provide the highest levels of security and universal acceptance.

Digital signatures, like handwritten signatures, are unique to each signer. Digital signature solution providers follow the specifics of the PKI protocol, which requires the provider to use a mathematical algorithm to generate two long numbers that are called *keys*. One key is *public*, and the other key is *private*.

When a signer electronically signs a document, the signature is created by using the signer's private key, which is always securely kept by the signer. The mathematical algorithm acts like a cipher by creating data that matches the signed document, called a *hash*, and encrypting that data. The resulting encrypted data is the digital signature. The signature is also marked with the time that the document was signed. If the document changes after signing, the digital signature is invalidated.

To protect the integrity of the signature, PKI requires that the keys be created, conducted, and saved in a secure manner, and often requires the services of a reliable certificate authority (CA). Digital signature providers must meet PKI requirements for safe digital signing.

Package signing is a security technology that IBM uses to certify that AIX development created an OS-related package. After a package is signed, the system can detect any change to the package content. The benefits of package signing are:

- ▶ Ensure that the package content has not been altered since it was signed. AIX systems can detect any form of changes to a package, whether it is intentional (by a malicious attacker) or accidental (for example, when a file is corrupted). When a package signature is intact, the AIX system is sure that the content is as the signer intended.
- ▶ Identify packages as coming from a specific source. The package signature includes cryptographic information that unambiguously points to a particular author (IBM AIX development). This information is included in the certificates that are published by the AIX package builder.
- ▶ Determine whether a package is trustworthy for a specific purpose. Among other things, an AIX package builder can use a digital signature to state that an updated version of a product should be considered by the AIX system to be the same product as the previous release version.

Package signing is just one component of a complete security solution that works in concert with other technologies, for example, AIX secure boot (2.2, “AIX Secure boot” on page 68) and AIX TE (2.1, “AIX Trusted Execution” on page 64). In the context of a comprehensive security strategy it is also important to know about the limitations of package signing:

- ▶ Package signing does not ensure that a package is free of security vulnerabilities.
- ▶ Package signing does not ensure that a software program will not load unsafe or altered code during execution.
- ▶ No copy protection technology is provided because package signing does not in any way hide or obscure the content of the signed package.

2.3.2 AIX package signing and digital signature catalog definition

Beginning with AIX 7.2 TL 4, AIX development delivers signed OS software packages in the `installp` format. The package signature consists of three parts:

| | |
|--------------------------|---|
| Seal | A seal is a collection of checksums or hashes of the various parts of a package that is created by the package signing software. The package seal is used at verification time to detect alterations. |
| Digital signature | The package signing software encrypts the seal by using the signer's identity to create a digital signature, which ensures the seal's integrity. |
| Sign requirements | These requirements are the security policies and rules governing the verification of the package signature. Some are inherent to the verifier (depending on its goals). Others are specified by the signer and sealed with the rest of the code (for example, version, release, modification, and fix (VRMF) level designation, message-digest algorithm, and keystore origin). |

Seal

The underlying AIX package signing program generates the seal by running different parts of the source through a one-way hashing algorithm. This algorithm produces a series of digests, or checksums, which are short strings of digits that are unique to a particular input block, but that cannot be used to reconstruct the original input.

A verifying entity (a user program) that has both the package under evaluation, and the corresponding collection of hashes, runs the same hashing algorithm on the package in the same way as the signer, and compares the results to the original stored hashes to see whether anything changed. Even a small modification in the code results in a different digest, which indicates tampering or corruption. However, this verification is only as reliable as the reliability of the stored hash. The digital signature ensures this reliability.

Digital signature

A digital signature uses public key cryptography to ensure data integrity. Like a signature that is written with ink on paper, a digital signature can be used to identify and authenticate the signer. However, a digital signature is more difficult to forge, and goes one step further: It can ensure that the signed data has not been altered. This is somewhat like designing a paper check or money order in such a way that if someone alters the written amount of money, the check becomes visibly invalid.

In the context of AIX package signing, the signing program creates the digital signature by encrypting the seal's hashes with the signer's private key. Because only the signer possesses the private key, only the signer may perform this encryption. It is this collection of encrypted hashes that the signer stores in the package envelope (or a catalog file) along with the matching certificate, which collectively represents the digital signature.

To verify the signature, the verifying program computes the same set of hashes across the various blocks of packaged data. It then uses the signer's public key, which is embedded in a certificate, to decrypt the encrypted hashes that came with the package, thus obtaining the original hash as computed by the signer. If the two hashes match, the data has not been modified since being signed by an author in possession of the signer's private key.

Usually, the digital certificate is signed by a trusted CA. If not, the verifier can be sure of the certificate's stability from one release to another but not of its origin. In relation to AIX packaging, a CA is not required; instead, a self-signed certificate is generated that contains the private key that is used when signing AIX packages during asset creation (build).

Sign requirements and security policies

Sign requirements, also referred to as security policies, are the set of established rules AIX is using to evaluate a package signature. The AIX system doing the evaluation decides what sign requirements to apply at evaluation time, depending on its goals.

Sign requirements that are specified by the signer and included as part of the package signature process are known as *internal requirements*, which are available to the system verifying a package signature, but the system may choose to use them or not (that is, disallow during diskless initialization or INUCLIENTS). Because the seal covers the sign requirements, the internal requirements are also certain to be intact if the package signature is valid.

The most important internal requirement is the DSC requirement. This rule tells an evaluating AIX system how to identify a particular package or image asset. Any two packages that have (and successfully verify against) the same DSC are considered to be the same package, which allows the AIX build process to publish a new release version of a package that is verifiable before the package installation. As an example, the DSC for AIX 7.2.2.0 would contain the same appropriate signatures for AIX package versions of AIX 7.2.0.0 previously included in the AIX 7.2 release. When a new version of the AIX package is published, if a new version of the `bos.rte.install` package has the same DSC, the package can be verified before installation, even if the binary executable file is different than the currently installed (included) executable file.

The signing methodology automatically builds a designated signature requirement by using the name of the program, VRMF designation, and keystore data that is found in its (system) DSC catalog file. As part of the signing process, the DSC catalog includes the private key and certificate signing request that is used for signing the build package. The resulting output is then stored in the newly constructed DSC for the VRMF designation in a build process. The constructed DSC is then included with the production-ready release of the AIX product.

Digital signature catalog keystore protection

The usage of PKI-based solutions continues to grow as corporations use digital certificates as an authentication factor for users and machines, but the encryption can be undermined if proper key management is not enforced.

Every time a digital certificate is issued, whether from a CA or self-signed, a private/public key pair must be generated. A best practice is that the private key should remain secure and private (not available to others). If anyone obtains the private key, depending on the certificate type, the actor could sign applications or packages by using IBM as the CA.

The options for private key protection and storage that are used by AIX are described in the following sections.

Several OSs provide certificate or keystores, which are software-based databases that store public/private key pairs as part of a certificate locally on the machine. This type of key storage is popular because many applications are then written to look in a specific location automatically rather than manually looking for the certificate file each time it is used.

Another appeal of this option is that it is fairly easy to customize, such as enable/disable private key exportability, enable strong private key protection (prompts for password every time the certificate is used), and the ability to create backups if the private key is exportable.

AIX 7.2 TL 3 implemented a software-based database to facilitate key storage with the following considerations:

- ▶ Even if a private key is marked as non-exportable, there are utilities that can circumvent this protection (non-exportability is not 100% guaranteed). As a precaution, AIX does not include a private certificate with the default keystore.
- ▶ If anyone accesses an AIX account and a package signing private key did not have strong private key protection enabled (no password is required to use the certificate), they could reuse the private certificate. As a precaution, AIX uses a strong (five words or longer) passphrase from the Diceware method.
- ▶ If the private key is marked as exportable, anyone that uses the system would be able to export the private key. Even if the private key were protection-enabled, a user would not have to enter the password to export it. So, AIX enforces control access rules for key management.

Public-Key Cryptography Standards #12 (PKCS#12) defines an archive file format for storing cryptography objects in a single file. It is widely used for files containing a public/private key pair. Unlike the locally stored OS and browser keystores, these files can be stored anywhere, including remote servers, and are always password-protected (meaning any time that the private key is requested, a password must be entered). Another appeal is that because these are files, users can easily distribute copies if multiple organizations (or people) must use the certificate.

Storing the file on a remote server or version control system is problematic because extra care must be taken to restrict access. The private key password is still required for the copied file to be used effectively, which is another reason to use strong passphrase passwords with more than five words containing mixed-case letters, numbers, and special characters.

The AIX DSC contains PKCS#12 keystore signatures for all signed package files. The signatures are distributed through ODM entries of the new `dsc_inventory` class. Figure 2-2 shows the definition of the new object class.

```
# odmshow dsc_inventory
class dsc_inventory {
    char pkg_name[145];           /* offset: 0xc ( 12) */
    char lpp_name[145];         /* offset: 0x9d ( 157) */
    short ver;                   /* offset: 0x12e ( 302) */
    short rel;                   /* offset: 0x130 ( 304) */
    short mod;                   /* offset: 0x132 ( 306) */
    short fix;                   /* offset: 0x134 ( 308) */
    char ftype[4];              /* offset: 0x136 ( 310) */
    vchar signature[1024];      /* offset: 0x13c ( 316) */
    vchar timestamp[64];       /* offset: 0x140 ( 320) */
    link dsc_key dsc_key id key[11]; /* offset: 0x144 ( 324) */
};
[ ... omitted lines ...]
```

Figure 2-2 ODM object class definition of `dsc_inventory`

Figure 2-3 shows the `dsc_inventory` stanza for the `bos.rte` product as part of the `bos` package in AIX 7.2 TL 4.

```
odmget dsc_inventory

[ ... omitted lines ... ]
dsc_inventory:
    pkg_name = "bos"
    lpp_name = "bos.rte"
    ver = 7
    rel = 2
    mod = 4
    fix = 0
    ftype = "S"
    signature =
"dzNvegbA1WYoQDDyDEqMn4UteS4TogFoH6DRpvs2bVTN/NVB6R2aoyxo9eTb116gE2vLNYmWlhGZGj
oMXBuIeu22Id1yd16Nrcmv6UFS9ge51cSzvGjXoT4BxXcj/1wViNSoyUJ6xFR9KoJr02WTMTUS06ub8
Tbh2aTxo81A2D4/Ah894EuBN8Q9Q9E6WbKuwE38Ho2XnDndz/bGkiaH4ovCuVGQohS9HgpXPOMMqUQ5
2gmp4o607u7S3fwEYqe5L5ZyoX6DzL12Z64hSY9b0X4wBjI805xnpCu21fy12R48muBUz50YD546cBt
iNiY2zSeXVfndIXe1TwJ6Gyn+3Q=="
    timestamp = "Mon_Oct_21_10-46-24_2019"
    key = "01"
[ ... omitted lines ... ]
```

Figure 2-3 Example DSC inventory stanza for the `bos` package and the `bos.rte` LPP

The variables of the structure that is shown in Figure 2-3 on page 76 have the following meanings:

| | |
|-----------|--|
| pkg_name | Contains the package (included) name for the file set record. |
| lpp_name | Contains the package (file set) name being represented by the digital signature. |
| ver | Contains the package version number. |
| rel | Contains the package release number. |
| mod | Contains the package modification number. |
| fix | Contains the package fix number. |
| ftype | Designates whether the package contains a base or an updated image. |
| signature | Holds the digital signature of the named package. |
| timestamp | Stores the date of image creation. |
| key | Specifies an audit chain identifying the <i>key</i> object that is used for signing verification |

Each signed package is added to the `dsc_inventory` along with information on the key certificate that is used during the signing process. This level of description allows for multiple entries to exist for a package name.

The implementation of an AIX trusted installation and update also defines the new object class `dsc_key` for certificates and keys. Figure 2-4 shows the associated characteristics as shown by the `odmshow dsc_key` command. The variables of the structure that are shown in Figure 2-4 have the following meanings:

| | |
|----------|--|
| id | Contains the unique integer that is assigned to a key object. |
| type | Specifies the type of key asset (certificate or public/private key). |
| alias | Common name that is used to describe or identify the object. |
| location | Absolute file path specifying where the key asset is stored. |
| modulus | Value string that is used for mapping “certificate” and “key” origination. |
| hash | Describes the cryptographic hash function that is used for signing. |
| Keystore | Specifies an audit chain identifying the “keystore” object of origination. |

```
# odmshow dsc_key
class dsc_key {
    long id; /* offset: 0xc ( 12) */
    char type[32]; /* offset: 0x10 ( 16) */
    char alias[256]; /* offset: 0x30 ( 48) */
    char location[256]; /* offset: 0x130 ( 304) */
    vchar modulus[128]; /* offset: 0x230 ( 560) */
    vchar hash[32]; /* offset: 0x234 ( 564) */
    link dsc_keystore dsc_keystore id keystore[11]; /* offset: 0x238 ( 568) */
}
[ ... omitted lines ... ]
```

Figure 2-4 ODM object class definition of `dsc_key`

Each key asset is added to a keystore file and included with each release of AIX. The file is defined to the AIX system by using a new object class `dsc_keystore` for keystore management and identification. Figure 2-5 shows the associated characteristics as shown by the `odmshow dsc_key` command.

```
# odmshow dsc_keystore
class dsc_keystore {
    long id; /* offset: 0xc ( 12) */
    char type[32]; /* offset: 0x10 ( 16) */
    char alias[256]; /* offset: 0x30 ( 48) */
    char location[256]; /* offset: 0x130 ( 304) */
};
[ ... omitted lines ... ]
```

Figure 2-5 ODM object class definition of `dsc_keystore`

The variables of the structure that are shown in Figure 2-5 have the following meanings:

| | |
|-----------------------|--|
| <code>id</code> | Contains the unique integer that is assigned to a keystore object. |
| <code>type</code> | Specifies the type of keystore format (pkcs12). |
| <code>alias</code> | Common name that is used to describe or identify the object. |
| <code>location</code> | Absolute file path specifying where the keystore asset is stored. |

The default keystore contains the default signing certificate that is intended for use by the AIX package verification tools and is secured with a general passphrase.

2.3.3 AIX digital signature catalog signing process

IBM AIX development implemented the DSC signing by completing the following steps:

1. Public / private key creation:
 - a. Create a public / private key pair and a corresponding self-signed public key certificate by running the `openssl` command.
 - b. The private key is placed in a secured keystore that is only accessible by the AIX development team.
 - c. The public key certificate of the private key is exported and placed in the default keystore for inclusion into the `bos.rte.install` file set.
2. Package signing:
 - a. Generate a package signature for each AIX package with the private key as input to special build tools.
 - b. The package signatures are stored in the corresponding signature file.
 - c. The signature file is preserved by the IBM AIX development organization.
3. DSC publishing:
 - a. All signatures are added as records to the `dsc_inventory` ODM database file.
 - b. The record entries include the package name, the location of the signature and hash to store, and the VRMF designation of the package.
 - c. The complete `dsc_inventory` file is saved for inclusion into the `bos.rte.install` file set.

2.3.4 Signature validation during AIX installation and update process

The following process steps are completed to conduct signature validation during the installation or update of OS file sets:

1. Verification key identification and retrieval:
 - a. During the file set installation or update process, the `installp / geninstall` command calls the internal package verification program (`/usr/sbin/pkgverify`), which in turn queries the `dsc_inventory` ODM class to find the matching record for the file sets in question.
 - b. If the record is found, the `pkgverify` program obtains the content of the signature field, which concludes the verification key identification and retrieval process step.
2. Package signature validation:
 - a. The package verification program performs the needed verification and returns the output to the calling `installp` program.
 - b. Depending on the system-wide signature policy setting of the kernel, provisions are taken to enforce the needed level of assurance. (For more information about the `sys0 signpolicy` attribute, see “Runtime control of the trusted installation and update” on page 80.)

On verification success, the file sets are marked as `SIGNED` for all of the available signature policy settings: low, medium, and high.

On verification failure, the following actions are taken:

Signature policy low The file set is marked as `UNTRUSTED`, but the installation is allowed to complete.

Signature policy medium This policy expects a user response through the supported force operation. If the installation is forced, the failure is handled as specified by the low signature policy setting. Otherwise, the installation is marked as failed.

Signature policy high The installation is marked as failed. The signature policy high is not supported for a new and complete overwrite or preservation installation of the base operating system (BOS) because catastrophic system states might be provoked.

- c. Only the verification results are recorded and stored during package installation. The failure of the installation process itself is not permitted at this level of assurance settings. In other words, when an installation begins, it must finish regardless of the verification result to ensure a defined state of the system.

2.3.5 AIX trusted installation and update controls

A system administrator has two options to control the digital signature policy settings:

- ▶ In the context of a complete overwrite or preservation BOS installation, the policy is defined with the help of BOS menus or a tailored configuration of the `bosinst.data` file.
- ▶ At run time, a new, dynamically changeable kernel attribute of the `sys0` device allows you to change the trusted installation and update policy according to the security guidelines of the organization.

Runtime control of the trusted installation and update

Since the release of AIX 7.2 TL 3, the runtime behavior of an AIX system regarding the trusted installation and update feature is governed by a new `signpolicy` attribute of the `sys0` kernel device. This attribute can be dynamically modified by the newly implemented `chsignpolicy` command. Figure 2-6 shows the help message of the `chsignpolicy` command, which gives an indication of the available attribute values.

```
# chsignpolicy -h
Usage: chsignpolicy [ -R | -p | -s {low|medium|high|none} ]
```

Figure 2-6 Help message for the `chsignpolicy` command

The `chsignpolicy` command flags provide the following choices:

| | |
|------------------------------|--|
| <code>chsignpolicy -R</code> | List range of policy options. |
| <code>chsignpolicy -p</code> | Prints the current policy option setting. |
| <code>chsignpolicy -s</code> | Sets the policy option for future installation operations. |

The `chsignpolicy` policy options govern the installation and update process as follows:

| | |
|-------------------------------------|---|
| <code>chsignpolicy -s none</code> | The installation process does not check for the signature for any package during the installation and update. |
| <code>chsignpolicy -s low</code> | If the signature verification fails, the file set is marked as UNTRUSTED and a warning message is issued, but the installation is allowed to complete. |
| <code>chsignpolicy -s medium</code> | If the signature verification fails, this policy expects a user response to confirm a force installation. If the operation is enforced, the failure is handled as specified by the low signature policy setting. Otherwise, the installation is marked as failed. |
| <code>chsignpolicy -s high</code> | If the signature verification fails, the installation is marked as failed. The signature policy high is not supported for the installation or update of the BOS installation because catastrophic system states might be provoked. |

Digital signature policy setting configuration at installation time

Beginning with AIX 7.2 TL 4, the installation process allows you to set the digital signature policy either by configuring the `bosinst.data` file or by using the BOS installation menus.

Traditionally, the `/usr/lpp/bosinst/bosinst.template` file provides guidance to configure a `bosinst.data` file for an automated non-prompted installation of the OS. Figure 2-7 shows the new `SIGN_POLICY` stanza in the `CONTROL_FLOW STANZA` section. More information about the `SIGN_POLICY` stanza is provided by the `/usr/lpp/bosinst/bosinst.template.README` files, as shown in Figure 2-8.

```
# vi /usr/lpp/bosinst/bosinst.template

[ ... omitted lines ... ]

#   TRUSTED_AIX = no,yes (defaults to no, enables Multilevel security)
#   TRUSTED_AIX_LSPP = no,yes (defaults to no, Trusted AIX in LSPP/EAL4+ mode)
#   TRUSTED_AIX_SYSMGT = yes,no (defaults to yes, installs Java with Tr.AIX
#   SECURE_BY_DEFAULT = no,yes (defaults to no, a minimal AIX Security
#                       Expert install
#   SIGN_POLICY = none,low,medium,high (defaults to none, sets digital
#                       signature policy for installing packages)
#   ADAPTER_SEARCH_LIST = a space separated list of adapters, that can be
#                       used to limit the disk selection choices.
#
#   PLEASE READ /usr/lpp/bosinst/bosinst.template.README for more information
```

Figure 2-7 New sample `SIGN_POLICY` control flow stanza in the `bosinst.template` file

```
# vi /usr/lpp/bosinst/bosinst.template.README

[ ... omitted lines ... ]

#   SIGN_POLICY:
#   Determines the digital signature security policy setting for
#   installing AIX packages.
#   When set to none, it does not check for any signatures on packages
#   when they are installed.
#   When set to low, it will install the package and issue a warning
#   message if the signature is invalid.
#   When set to medium, the user will be required to confirm the
#   installation of the package if its signature is invalid.
#   When set to high, the package will not be installed if its
#   signature is invalid.
#
[ ... omitted lines ... ]
```

Figure 2-8 The `/usr/lpp/bosinst/bosinst.README` information about the `SIGN_POLICY` stanza usage

During an attended BOS installation, system administrators may tailor the installation process with the help of the BOS installation menus. The main Installation and Settings BOS menu allows you to select the Security Model under option 3, as shown in Figure 2-9. Selecting option 3 brings you to the Security Models submenu (Figure 2-10), where the new Digital Signature Policy option allows you to choose between the values none, low, medium, and high.

```
Installation and Settings

Either type 0 and press Enter to install with current settings, or type the
number of the setting you want to change and press Enter.

    1 System Settings:
      Method of Installation.....New and Complete Overwrite
      Disk Where You Want to Install....hdisk2...

    2 Primary Language Environment Settings (AFTER Install):
      Cultural Convention.....English (United States)
      Language .....English (United States)
      Keyboard .....en.IS08859-1
      Keyboard Type.....Default

    3 Security Model.....Default
    4 More Options (Software install options)
    5 Select Edition.....standard

>>> 0 Install with the current settings listed above.
```

Figure 2-9 Main Installation and Settings menu

```
Security Models

Type the number of your choice and press Enter.

    1. Trusted AIX..... No
    2. Digital Signature Policy..... None

    3. Other Security Options (Trusted AIX and Standard)
      Security options vary based on choices.
      LAS, Sbd, BAS/CCEVAL

>>> 0 Continue to more software options.
```

Figure 2-10 Security Models submenu to configure the digital signature policy

2.4 Multifactor authentication

This section introduces solutions for *multifactor 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 they claim to be before granting access to a service or a resource.

Examples of authentication in daily life include:

- ▶ Opening a door by using a badge.
- ▶ Showing a driver license, an identity card, or a passport.
- ▶ Typing a user ID and a password to log in to a computer.

In all these cases, the person who must authenticate uses only one type of authentication of their identity.

In IT, providing a user name and password is perhaps the most commonly used method of authentication. However, this method has two disadvantages:

- ▶ User names can be easily disclosed or shared with other people.
- ▶ A 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 that makes them more difficult to crack but also more difficult to remember. This situation naturally led to development of other solutions to complement or extend the security that is provided by authentication uses only passwords. More complex solutions for authentication employ other authentication factors, such as cryptographic tokens or fingerprints.

2.4.1 Authentication factors

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

- | | |
|--------------------------------|---|
| Something that you know | This is information that you know, such as a password, passphrase, confirmation number, or Personal Identification Number (PIN). |
| Something that you have | This is an item that 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 that you are | This might be a fingerprint, retina scan, or other biometric data, such as data that is stored on chips that are on a biometric passport. |

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

MFA is employed in various activity domains where enhanced security is a requirement. There are practices and standards that govern access to government buildings or personal data that explicitly requires the usage of MFA to establish a person's identity.

One such example might be using a password and token-generated code to get access to personal financial data. Another example might 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 two types:

| | |
|-----------------------------------|---|
| In-band authentication | When using this method authentication, credentials are provided by using the same communication channel that is used to access the target service. |
| Out-of-band authentication | When using this method, authentication credentials are provided by using an alternative communication channel that is different from the one that is used to access the target service. |

2.4.3 In-band MFA

This section describes how MFA support in-band authentication by using various authentication factors. When using an in-band authentication method, the user who wants to log in must first get a token. The token can be generated by using IBM PowerSC MFA with various authentication factors. After it is generated, the token can be used directly to 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 the credentials that are provided are valid. This solution uses the following factors:

| | |
|--------------------------------|--|
| Something that you have | The RSA SecurID token, whether it is hardware or software. |
| Something that you know | The RSA SecurID PIN and something you know. |

Certificate-based authentication options

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

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

For in-band solutions, the PIV/CAC card readers can be directly connected to select IBM Power Systems servers (IBM Power System S812, IBM Power System S814, IBM Power System S822, IBM Power System S824, IBM Power System S914, IBM Power System S922, and IBM Power System S924) that use USB interfaces. The systems must run AIX 7.1 TL 5 Service Pack 1 or AIX 7.2 TL 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 might also work, but have not been tested at the time at writing. For more information about supported devices and PIV and CAC, see [IBM Knowledge Center](#).

IBM TouchToken

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

Something that you have A valid IBM TouchToken account and IBM TouchToken, IBM Verify, or other compatible app that is provisioned on a mobile phone.

Something that you are The personal fingerprint.

For more information about the IBM TouchToken solution, see [IBM Knowledge Center](#).

Radius-based solutions

In this case, MFA collaborates with a RADIUS server to decide whether the credentials that are provided are valid. The 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 solution employs a client/server architecture that 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 log in must log in first to an out-of-band specific web page by using one or more authentication factors to retrieve an authentication item that is named a Cache Token Credential (CTC). The CTC is valid for a specific period during which it can be used for directly logging in.

Starting with MFA V1.2, out-of-band authentication is supported for HMC V9R1.921.

2.4.5 Authentication on AIX systems by using RSA SecureID

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

Authentication manager

This is the core component of the solution that 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 that is provided and initiates an authentication session with the authentication manager. Depending on the answer that it gets from the authentication manager, it grants or denies the request.

Authenticator The authenticator handles the data that is used in the process of authentication. This authenticator can be a hardware or software RSA SecurID token.

Pluggable Authentication Module (PAM) framework

PAMs can incorporate various authentication mechanisms into an existing system by using individual pluggable modules. AIX implementations are composed of a library, pluggable modules, and a configuration file.

A 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 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 includes the following steps:

1. A user attempts to access a service on an AIX system where a 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 start the PAM application programming interface (API) directly for authentication rather than using the traditional AIX authentication routines. The following native AIX commands and applications are 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 that is requested, the corresponding access request is started. For example, **login** is used for local console access, and **telnet** or **ssh** can be used for remote logon.
4. The PAM module intercepts all logon requests, and by using PAM configuration files, directs the requests to the RSA module. If the user requesting access is configured not to use RSA SecurID, the RSA PAM module allows the request to proceed. If the user requesting access is configured to use RSA SecurID, the RSA PAM module proceeds with the authentication process. The RSA PAM agent on AIX provides RSA SecurID support for the following commands:
 - **login** (console)
 - **su**

- **ssh**
- **sudo**
- **rlogin**
- **telnet**
- **ftp** (limitations apply)

For more information about the RSA SecurID access authentication agent for AIX versions 7.1 and 7.2, see the following resources:

- [IBM AIX 7.1 - RSA SecurID Access Authentication Agent Implementation Guide](#)
- [RSA Announces RSA Authentication Agent for PAM 7.1.0.2 and Support for New Platforms](#)

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

2.5 Cryptographic libraries

This section introduces the cryptographic libraries that are supported on AIX.

2.5.1 OpenSSL

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

OpenSSL has no charge for commercial purposes, but there are terms and conditions that are included in the license that must be accepted before using it. Additionally, legislation regarding the use of cryptographic software varies from one country to another, and the usage of OpenSSL and any other cryptographic software is subject to laws and regulation that are specific to each country.

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

For more information about OpenSSL, see [OpenSSL](#).

2.5.2 CryptoLite for C library

The CliC library provides application programmers with the cryptographic primitives that are 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 OS relies on the CliC library for features such as Encrypted File System (EFS), Network File System (NFS) V4, and TE.

2.6 Address space layout randomization

One avenue of attack on vulnerable programs is to cause a *buffer overrun*, which modifies the return address for a function. Early attacks used the buffer overrun to put executable code on the process stack, but this technique is thwarted if the stack has the no-execute property. Therefore, current attacks rely on a *return-to-libc* technique. If a useful piece of code can be found in a loaded library, the return address can be overwritten so that control is transferred to the useful code (for example, the `execve()` system call) when a function returns, allowing an attacker to gain root access.

This technique is made more difficult if the useful code is hard to find. Randomizing text addresses achieves this goal. Because data sections can contain addresses of text, one way to determine the address of a useful piece of code is to find a data address containing a pointer to the useful code. Therefore, it is advisable to randomize data addresses and text addresses.

The current security guidelines of the CC GPOSPP Version 4.2.1 of the NIAP recommends mitigating the risk of return-to-libc attacks by loading memory mappings into unpredictable locations. This anti-exploitation technique is referred to as *address space layout randomization* (ASLR). The ASLR security requirement is described in “FPT_AS LR_EXT.1 Address Space Layout Randomization” in [Protection of the Target of Evaluation Security Functionality](#).

Beginning with AIX 7.2 TL 3 Service Pack 1, the OS randomizes process address space memory locations for the following entities:

- ▶ Main program text, main program data, and stacks for 32-bit and 64-bit processes
- ▶ Shared library text and data addresses and privately loaded libraries for 64-bit programs
- ▶ Shared memory attached segments (`shmat()`) and memory mapped files (`mmap()`) for 64-bit programs

Conversely, when ASLR is enabled, the AIX OS randomizes process address space memory locations except for the following entities:

- ▶ Shared memory segments (`shmat()`) and memory mapped files (`mmap()`) in 32-bit programs.
- ▶ Privately loaded libraries in 32-bit programs.
- ▶ Programs that are not marked to request or allow randomization.
- ▶ Programs that are marked to disable randomization.

To support the ASLR feature, the `ld` and `ldedit` commands are enhanced to mark programs with the relevant randomization attributes. To facilitate system-wide control over ASLR, the `vmo` command provides new tunables to govern the ASLR activation in general and the behavior for 32-bit and 64-bit programs specifically.

Randomized addresses can be provided per-process, per-shared-library-area (per-SLA), per-system, or per-program:

- ▶ Per-process randomization provides random addresses for each process individual execution.
- ▶ Per-SLA randomization provides random addresses for each user of a shared library area (SLA).

- ▶ Per-system randomization provides for random addresses when comparing different systems.
- ▶ Per-program randomization provides random addresses for concurrent users of a program.

Per-process randomization provides the best protection against attackers, but is the most costly, both in real memory usage and in the performance of the AIX Virtual Memory Manager (VMM), especially for randomizing shared library addresses.

In summary, with AIX 7.2 TL 3 Service Pack 1, the following enhancements in support of ASLR were implemented:

- ▶ The OS can randomize the address-space layout of shared libraries and marked programs.
- ▶ Some selected AIX programs are marked so that their address spaces may be randomized if ASLR is enabled by the system administrator.

System administrators may control ASLR as follows:

- ▶ Enable randomization of shared library addresses that are used by all programs.
- ▶ Enable ASLR for the programs that are marked to allow randomization.
- ▶ Allow randomization for new programs or extra existing programs by using the `-bas1r` option with the `ld` or `ldedit` command.

The remainder of this chapter provides more information regarding ASLR implementation and control.

2.6.1 Process address space randomized entities in AIX

Beginning with AIX 7.2 TL 3 Service Pack 1, randomization of different parts of the process address space is supported. The following sections cover the randomization techniques that are used in AIX ASLR for the following process address space entities:

- ▶ Main program text
- ▶ Main program data
- ▶ Main stack
- ▶ Shared library text
- ▶ Shared library data
- ▶ Privately loaded libraries

Main program text

The main program text contains position-independent code and can be relocated easily, except for programs that are compiled by the `roptr` (places constant pointers in read-only storage), `ro` (places string literals in read-only storage), and `roconst` (places constants in read-only storage) compiler options. These options allow many address constants to be moved from the data section to the text section, and require that programs be link-edited at their load-time addresses because they have text-section relocations.

A main program is made addressable by mapping the executable file segment into Effective Segment ID (ESID) 0x1 for a 32-bit program and ESID 0x10 for a 64-bit program. Multiple processes running the same program automatically share the main program's text because they use the same Segment ID (SID).

The loader supports programs that are larger than a segment. In a 64-bit process, the segments of the program are loaded into multiple ESIDs. In a 32-bit program, working-storage segments are created for the program, and the program's header, text section, and loader section are mapped into the segments. These working-storage segments are mapped into consecutive ESIDs of the process address space.

Main program text of 32-bit programs

AIX randomizes the text address of a main 32-bit program by creating a working-storage segment and reading or mapping the file at a random offset into the segment. If all processes running the same program concurrently share the working-storage segment, per-program randomization is achieved.

AIX ASLR implements per-program randomized page offsets for all 32-bit programs, including huge executable files and processes, by using an alternative page size. If the requested page size is pageable, AIX chooses a random offset that is a multiple of the page size. Otherwise, a random multiple of the text alignment is chosen. This technique is also used for programs larger than a segment (huge executable files).

Main program text of 64-bit programs

With ASLR enabled, AIX randomizes the text address of a main program on a per-process basis by choosing random ESIDs and mapping the file into those ESIDs. Using random ESIDs still allows the main program's text to be shared because the ESIDs are shared. The main 64-bit program is mapped to a randomized ESID 0x10 - 0x06FFFFFFF.

Main program data

The main program data of a 32-bit program starts at the beginning of segment 2 or segment 3, depending on the address space model. The main program data of a 64-bit program starts at the first ESID following the main program text. Two options are available to randomize data addresses on a per-process basis: Either a random ESID is used, or the data starts at a random offset within an ESID.

Main program data of 32-bit programs

Normally, the data for the main program begins at the beginning of the data segment, although in some situations privately loaded libraries precede the main program data. The data segment is ESID 2 for most programs, but ESID 3 is used for large or very large address-space model (maxdata) programs and LARGE_PAGE_DATA programs. To randomize data addresses for 32-bit programs, AIX ASLR provides a random offset for the main program data, accounting for the data resource limit and the address space model.

Main program data of 64-bit programs

Normally, data starts in the first page of the first ESID following the text segments of 64-bit programs. AIX ASLR randomizes the ESID that is used for data.

Main stack

The initial stack pointer for the main program is created by copying the CLI arguments and environment variables to memory just before `errno`, and then rounding down to a 16-byte boundary. The address of `errno` is fixed near the end of ESID 2 for 32-bit processes and at the end of ESID 0xFFFFFFFF for 64-bit processes.

AIX ASLR implements a per-process stack randomization by decrementing the initial stack pointer value by a multiple of 16 bytes for 32-bit and 64-bit processes.

Shared library areas and private libraries

SLAs use a fixed set of ESIDs into which shared library text and data are allocated. AIX ASLR randomizes SLAs of 64-bit processes by using random allocation of text and data within the ESIDs.

Virtual memory for shared library text and data and privately loaded libraries in 64-bit processes is allocated by using `xmalloc()`. But, the heaps that are used by the system loader are special because fragments are not supported. All allocations are rounded up to a multiple of 4 K, and the allocation is performed by an internal routine that is named `pmalloc()`. Therefore, heap addresses are randomized by an enhanced version of the `pmalloc()` routine in AIX 7.2 TL 3 SP 1 or later.

Shared memory segments and memory mapped files

Undirected `shmat()` system calls always return a segment-aligned address and consume a full segment of address space. (For `EXTSHM=ON`, the behavior is the same as with `mmap()`). The `mmap()` system call allocates a segment that does not conflict with an address that is returned by `shmat()` and allocates enough pages from the segment to map the object. Subsequent calls to `mmap()` use an existing `mmap()` segment if there is enough remaining space for the object. Otherwise, a new segment is allocated.

Randomizing `shmat()` for 32-bit processes is not effective because there are only 14 possible return values. For `mmap()`, it is possible to randomize allocation within a segment, but this situation is of limited use. Therefore, AIX ASLR does not randomize `shmat()` or `mmap()` addresses for 32-bit processes.

For 64-bit processes, the default region for `shmat()` and `mmap()` is any 64-bit address beginning with 0x0A. AIX ASLR randomizes the first ESID that is used for `shmat()` and `mmap()`, but return ESIDs in consecutive order for subsequent calls.

2.6.2 ASLR tuning and control

Randomizing addresses of existing programs can lead to failures, especially if programs make assumptions about their address space. Therefore, the AIX ASLR allows you to control whether randomization occurs or not. The implemented controls are both system-wide and per-program specific. System-wide controls are dynamic.

AIX 7.2 TL 3 and later releases provide one single global tunable that controls randomization, named `aslr`, which can be modified by the `vmo` command. Figure 2-11 shows the supported values for the `aslr` tunable. By default, ASLR is disabled or is controlled by undocumented restricted tunables.

```

root@roell:/ >vmo -h aslr
Help for tunable aslr:
Purpose: Specifies whether ASLR (address space layout randomization) is used.
Values:
    Default: 0
    Range: 0 - 2
    Type: Dynamic
    Unit: numeric
Tuning:
0 means that ASLR is disabled or is controlled by a restricted tunable.
1 means that randomization is used for shared library areas.
2 means that randomization is used for shared library areas and allowed for
marked programs.

```

Figure 2-11 Help message for the `aslr` `vmo` tunable

The `aslr` tunable can be changed dynamically. The value of the tunable is checked at allocation time for SLAs and at program run time.

AIX ASLR implementation defines three randomization attributes for all programs (main-program text, main-program data, and stacks) and two more randomization attributes for 64-bit programs (privately loaded libraries, and `shmat()` and `mmap()`).

To implement ASLR, the XCOFF format was extended in AIX 7.2 TL 3 to allow program settings to be recorded in the XCOFF header.

The AIX ASLR feature also adds the `-baslr` option to both the `ld` and `ldedit` commands to allow program settings to be specified. If the `-baslr` option has never been used, all program settings are unspecified. A marked program can be restored to its unspecified state by running the `ldedit -bnoaslr` command.

The `ld` command is used to link object files, and the new `-baslr` and `-bnoaslr` options are explained in Table 2-2.

Table 2-2 New `ld` command options support for ASLR

| Binder option | Description |
|--|--|
| <code>aslr</code> <code>aslr:[tdsmp]*</code> <code>aslr:-</code> | Using <code>aslr</code> by itself sets all attribute values to on. When <code>aslr</code> is followed by a colon, individual attributes can be turned on, where <code>t</code> , <code>d</code> , <code>s</code> , <code>m</code> , and <code>p</code> denote text, data, stack, <code>shmat()</code> and <code>mmap()</code> , and private-library attributes. ^a If some attribute values are not specified, they remain off. <code>-baslr:-</code> disables all of the randomization attributes. |
| <code>noaslr</code> | Cancels the effect of previous <code>aslr</code> options on the command line. |

a. Note: `m` and `p` cannot be specified for 32-bit programs.

Selected programs are compiled and linked so that some relocatable addresses are put in the text section, requiring relocation of the text section at run time. These programs generally fail if randomization is enabled for text and data. Therefore, when text-section relocation exists, text and data randomization are turned on only if program text (`t`) and program data (`d`) are explicitly specified with the `aslr` binder option.

The new `-baslr` and `-bnoaslr` options that modify an XCOFF executable file header by using the `ldedit` command are described in Table 2-3.

Table 2-3 New `ldedit` command options support for ASLR

| Binder option | Description |
|--|--|
| <code>aslr</code> <code>aslr: [[+-] [tdsmp]*]*</code> | Using <code>aslr</code> by itself sets all attribute values to on. When <code>aslr</code> is followed by a colon, attribute values can be specified, where + denotes on and - denotes off. Values can be followed by an optional list of attributes, where <code>t</code> , <code>d</code> , <code>s</code> , <code>m</code> , and <code>p</code> denote text, data, stack, <code>shmat()</code> and <code>mmap()</code> , and private-library attributes. ^a If no attributes are specified, the attribute value applies to all values. If values are not specified for some attribute, their values are unchanged. |
| <code>noaslr</code> | Cancels the effect of previous <code>aslr</code> options on the command line. |

a. Note: `m` and `p` cannot be specified for 32-bit programs.

For each program attribute, AIX offers corresponding *system tunables* for both 32- and 64-bit programs. The ASLR system-wide properties for 32-bit programs are specified by the `aslr32` tunable and for 64-bit programs by the `aslr64` tunable.

Figure 2-12 shows the supported values for the `aslr32` tunable. By default, ASLR may be used for marked 32-bit programs for the randomization attributes main-program text (`t`), main-program data (`d`), and stack (`s`). The command `vmo -aslr32=1td` limits the randomization enablement to the main-program text and data of all marked 32-bit programs, for example.

```

root@roell:/ >vmo -h aslr32
Help for tunable aslr32:
Purpose: Specifies ASLR properties for 32-bit programs
Values:
    Default: "1"
    Range: -
    Type: Dynamic
    Unit: string
Tuning:
0 means that ASLR is disabled for 32-bit programs or is controlled by a
restricted tunable.
1 means that ASLR is allowed for marked 32-bit programs. You can also specify
one or more of the characters t, d, and s to allow randomization for the main
program text, main program data, or stack of marked 32-bit programs.

```

Figure 2-12 Help message for the `aslr32` `vmo` tunable

Figure 2-13 shows the supported values for the `aslr64` tunable. By default, ASLR may be used for marked 64-bit programs for the randomization attributes main-program text (`t`), main-program data (`d`), stack (`s`), privately loaded library (`p`), and `shmat()` and `mmap()` (`m`) addresses. The command `vmo -aslr64=1tdm` limits the randomization enablement to the main-program text, main-program data, and `shmat()` and `mmap()` addresses of all marked 64-bit programs, for example.

```
root@roell:/ >vmo -h aslr64
Help for tunable aslr64:
Purpose: Specifies ASLR properties for 64-bit programs
Values:
    Default: "1"
    Range: -
    Type: Dynamic
    Unit: string
Tuning:
0 means that ASLR is disabled for 64-bit programs or is controlled by a
restricted tunable.
1 means that ASLR is allowed for marked 64-bit programs. You can also specify
one or more of the characters t, d, s, m, and p to allow randomization for the
main program text, main program data, stack, shmat() or mmap() addresses, or
privately loaded libraries of marked 64-bit programs.
```

Figure 2-13 Help message for `aslr64 vmo` tunable

2.7 Trusted shared library area support

AIX 7.2 TL 3 supports trusted SLAs for both 32- and 64-bit trusted programs.

A *trusted SLA* is an unnamed SLA that can be used by selective programs, presumably trusted (or privileged) ones. A *trusted program* is a program that is either `setuid-root`, `setgid-system`, or `setgid-security`. If trusted programs use a trusted SLA, they are no longer susceptible to non-trusted programs adding malicious library code to the SLA. In addition, addresses that are used by the trusted SLAs are different from the addresses that are used by the non-trusted SLAs, making it harder for attackers to exploit trusted programs. Thus, using a separate trusted SLA provides more address space isolation for trusted programs.

AIX 7.2 TL 3 provides a single global tunable to control the usage of trusted SLAs. This tunable is named `useTrustedSLAs`, and can be modified dynamically by using the `vmo` command.

Figure 2-14 shows the supported values for the `useTrustedSLAs` tunable. By default, trusted SLAs are not used for newly run processes. A value of 1 (`vm0 -o useTrustedSLAs=1`) means that the usage of trusted SLAs is in effect. If the tunable is changed dynamically, future run times of trusted programs use the new value, but existing programs are unaffected. This setup ensures, for example, that daemons that are started early in the boot process are also covered by the protection mechanism.

```
root@roe11:/ >vm0 -h useTrustedSLAs
Help for tunable useTrustedSLAs:
Purpose: Specifies whether trusted shared library areas are used.
Values:
    Default: 0
    Range: 0, 1, 32, 64
    Type: Dynamic
    Unit: numeric
Tuning:
0 means that trusted shared library areas are not used.
1 means that trusted shared library areas are used for trusted programs, that
is, programs that are setuid-root, setgid-system, or setgid-security. You can
also specify 32 or 64 to enable trusted shared library areas for 32-bit trusted
programs only or 64-bit trusted programs only.
```

Figure 2-14 Help message for the `useTrustedSLAs` `vm0` tunable



Networking enhancements

This chapter describes networking enhancements that are available for AIX.

This chapter describes the following topics:

- ▶ Redundant link aggregation network interface backup
- ▶ Shared memory communication over Remote Direct Memory Access

3.1 Redundant link aggregation network interface backup

Beginning with AIX 7.2 Technology Level (TL) 2, the Etherchannel and IEEE 802.3ad link aggregation implementation is enhanced to include multiple adapters as part of its redundancy failover strategy. The backup interface in a network interface backup (NIB) configuration can be defined as an Etherchannel or IEEE 802.3ad aggregated link. System administrators can specify the primary interface as an Etherchannel / IEEE 802.3ad device and the backup interface as an Etherchannel / IEEE 802.3ad device. This architecture retains the performance benefits of a primary interface aggregated ports and links configuration if a failover event occurs. If multiple backup adapters are configured and available, they operate as Etherchannel or IEEE 802.3ad devices in the same mode or hash mode that is defined for the primary channel. Table 3-1 provides an overview of the available mode and hash mode combinations for Etherchannel and IEEE 802.3ad devices.

Table 3-1 Mode and hash mode combinations and the outgoing traffic distributions each one produces

| Mode | Hash mode | Outgoing traffic distribution |
|---------------------------------|--------------|--|
| Etherchannel standard or 8023ad | default | The adapter selection algorithm uses the last byte of the destination IP address (for TCP/IP traffic) or MAC address (for address resolution protocol (ARP) and other non-IP traffic). |
| Etherchannel standard or 8023ad | src_port | The adapter selection algorithm uses the source TCP or UDP port value. |
| Etherchannel standard or 8023ad | dst_port | The outgoing adapter path is selected by the algorithm by using the destination system port value. |
| Etherchannel standard or 8023ad | src_dst_port | The outgoing adapter path is selected by an algorithm by using the combined source and destination TCP or UDP port values. |
| Etherchannel round-robin | default | The outgoing traffic is spread evenly across all the adapter ports in the Etherchannel. This mode is not supported for IEEE 802.3ad LA configurations. |

Up to eight primary Ethernet adapters and up to eight backup Ethernet adapters are supported per Etherchannel or IEEE 802.3ad link aggregation device.

Note: Multiple primary single root input/output virtualization (SR-IOV) logical ports are allowed in a Link Aggregation Control Protocol (LACP) IEEE 802.3ad device. This statement also applies to an IEEE 802.3ad link aggregated device if it is used as backup interface in a NIB configuration.

An SR-IOV logical port cannot be included as a primary adapter in an Etherchannel configuration with more than one primary adapter because only one primary SR-IOV logical port is allowed in an Etherchannel device. This limitation also applies to an Etherchannel device if it is used as backup interface in a NIB configuration.

For more information about SR-IOV supported configurations, see *IBM Power Systems SR-IOV: Technical Overview and Introduction*, REDP-5065.

3.2 Shared memory communication over Remote Direct Memory Access

Before AIX 7.2 TL 2, existing user-space applications or kernel extensions that were going to use transports that support Remote Direct Memory Access (RDMA), such as InfiniBand, iWARP, and RDMA over Converge Ethernet (RoCE), had to be refactored to use the user Direct Access Programming Library (uDAPL) services for the user-space domain or the kernel Direct Access Programming Library (kDAPL) services for the kernel-space domain. kDAPL operates on top of OpenFabrics Enterprise Distribution (OFED) or AIX InfiniBand. Kernel extensions can directly access these underlying services and bypass kDAPL, but that requires extensive modifications of networking application programming interfaces (APIs).

The implementation of the shared memory communication over RDMA (SMC-R) protocol in AIX 7.2 TL 2 removes this limitation and enables applications with TCP socket-based communication architecture to transparently use RDMA without needing to adapt and recompile any code.

The SMC-R protocol is based on the Internet Engineering Task Force (IETF) Request for Comments (RFC) 7609 and uses the existing socket API for IPv4 or IPv6 to allow access to RDMA transports and to enable direct, high-speed, and low-latency communication. The protocol allows for dynamic discovery of its TCP peers' RDMA capabilities, which enables an automatic setup of the RDMA connections. High availability (HA) and load-balancing capabilities are native to the SMC-R protocol.

SMC-R-based communication provides significant performance gains and CPU resource reduction for multitier server workloads with request/response patterns or for data center server communication with streaming data patterns. SMC-R is suited to support effective high-speed file transfer solutions based on the FTP, SFTP, and Network File System (NFS) protocols; IBM Sterling Connect:Direct®; and IBM WebSphere® MQ File Transfer Edition. SMC-R communication should be considered for solution architectures that rely on database load and unload utilities. In this context, AIX SMC-R is fully interoperable with the IBM z/OS® and the Linux on Z SMC-R implementations.

SMC-R communication in AIX adheres to the following characteristics:

- ▶ Hosts participating in SMC-R communication must provide RDMA-capable Network Interface Cards (NICs). All RoCE adapters that are supported on IBM Power Systems servers can facilitate SMC-R protocol-based communication. RoCE adapters must be dedicated to the logical partition (LPAR) communication endpoints.
- ▶ Loopback transport is not supported. The AIX fast path loopback option is suited to meet ambitious performance requirements for connections that are initiated from and to the localhost.
- ▶ The IP addresses of the communication endpoints must be in the same IP subnet (or prefix if you are using IPv6).
- ▶ SMC-R can be enabled only for TCP-based workloads.
- ▶ No Etherchannel / IEEE 802.3ad link aggregation is supported because the SMC-R protocol inherently facilitates failover and load-balancing capabilities if the required adapter resources are available.
- ▶ Up to two links per link group are supported.
- ▶ Kernel space TCP sockets are not supported.
- ▶ Virtual LAN (VLAN) support for SMC-R is available with AIX 7.2 TL 3 or later releases.

For more information about the SMC-R implementation, use cases, and performance, see the following resources:

- ▶ [Shared Memory Communications over RDMA \(SMC-R\)](#)
- ▶ [SMC-R on AIX for SAP - Experience report](#)
- ▶ [IBM Knowledge Center](#)



Virtualization and cloud capabilities

This chapter describes the virtualization capabilities for IBM AIX and options that are available for automating your AIX deployments and management.

This chapter describes the following topics:

- ▶ AIX on public cloud
- ▶ IBM Power Virtualization Center
- ▶ IBM Cloud Automation Manager
- ▶ Ansible automation and AIX
- ▶ Chef Infra client on AIX
- ▶ Puppet Enterprise on AIX

4.1 AIX on public cloud

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

- ▶ Providing a rapidly deployable development or test environment
- ▶ Providing an offsite disaster recovery (DR) environment
- ▶ Scaling workloads

AIX is available on several cloud providers. At the time of writing, here are the resources to learn more about them:

- ▶ [IBM Power Systems Virtual Servers on IBM Cloud](#)
- ▶ [Google Cloud](#)
- ▶ [Skytap](#)

4.2 IBM Power Virtualization Center

This section provides a general presentation of IBM Power Virtualization Center (IBM PowerVC) and introduces the latest IBM PowerVC capabilities.

4.2.1 Introduction

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

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

- ▶ PowerVM with a traditional Virtual I/O Server (VIOS) that uses either a Hardware Management Console (HMC) or NovaLink
- ▶ PowerVM with a software-defined I/O that uses NovaLink
- ▶ KVM on POWER with software-defined I/O

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

- ▶ Create VMs.
- ▶ Resize cores and memory that are allocated to VMs.
- ▶ Attach storage logical unit numbers (LUNs) to VMs.
- ▶ Define templates for VMs and deploy new VMs that are based on templates.
- ▶ Define storage providers and templates for VMs and deploy new LUNs that are 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 Power Systems server to another by using Live Partition Mobility (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 VM placement across multiple Power Systems servers.
- ▶ Inject Secure Shell (SSH) keys into newly deployed VMs.
- ▶ Import and export VMs among clouds.

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

- ▶ One-click deployment templates
- ▶ Project-based resource isolation (multi-tenancy)
- ▶ Role-based access control (RBAC)
- ▶ Approvals and expirations
- ▶ Definition of policies
- ▶ Metering data and email notifications

Because IBM PowerVC is based on OpenStack, Power Systems servers 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 a native IBM PowerVC GUI or CLI.
- ▶ Using scripts or applications that use IBM PowerVC Representational State Transfer (REST) application programming interfaces (APIs).
- ▶ Using any other application than can launch IBM PowerVC by using a standard OpenStack API.

PowerVM NovaLink

PowerVM NovaLink is a highly scalable software virtualization management interface for IBM POWER8 and POWER9 processor-based servers that use PowerVM virtualization. Although it represents an alternative to the HMC, it does *not* offer hardware or service management.

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

PowerVM NovaLink is deployed in a Linux VM by using either Ubuntu or Red Hat Enterprise Linux (RHEL). PowerVM NovaLink also features an installer that simplifies the deployment by automatically creating the VMs that are used for VIOs and NovaLink. The installer also deploys VIOS and NovaLink software, and optionally Open vSwitch (OVS) when using software-defined networking (SDN), which reduces deployment time for complex environments.

PowerVM NovaLink integrates with IBM PowerVC or other OpenStack solutions to manage Power Systems. It enables SDN to allow IBM PowerVC or OpenStack software to virtualize networks by using industry standard OVS and software-defined storage (SDS) to allow IBM PowerVC or OpenStack software to virtualize storage by using IBM Spectrum Scale (formerly known as IBM GPFS).

4.2.2 IBM PowerVC in the virtualization and cloud software stack

IBM PowerVC is part of a comprehensive Power Systems server-based virtualization and cloud software stack. IBM PowerVC offerings are positioned within the available solutions for the 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

IBM PowerVC integration into the architecture of HMC-managed Power Systems environments is depicted in Figure 4-1.

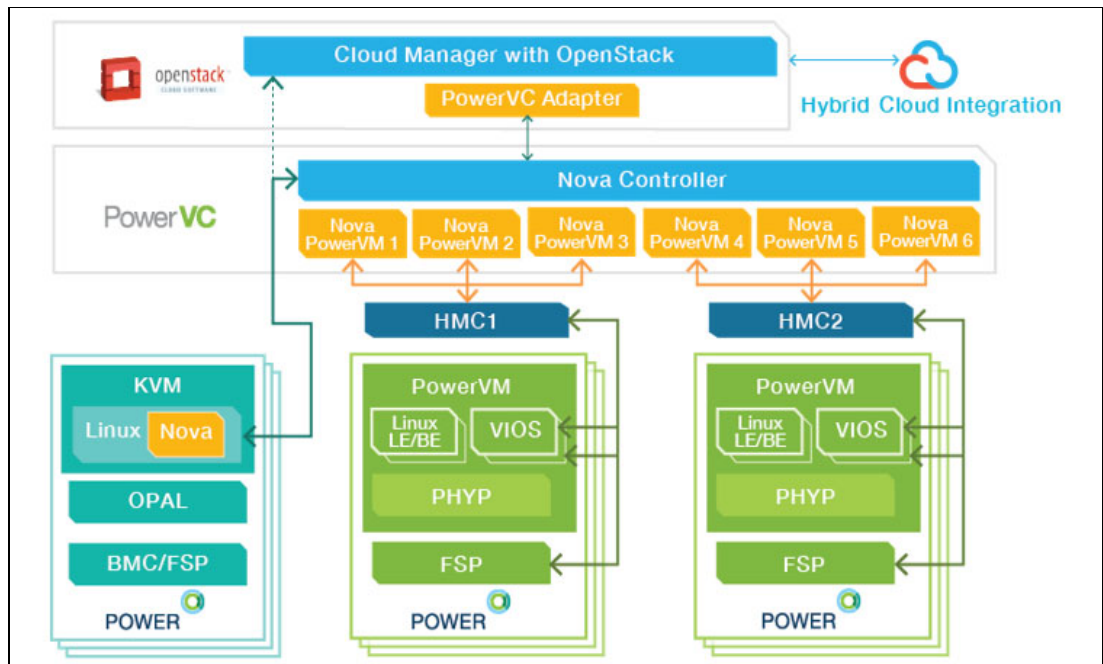


Figure 4-1 IBM PowerVC integration into HMC-managed Power Systems environments

IBM PowerVC integration into the architecture of NovaLink -managed Power Systems environments is depicted in Figure 4-2.

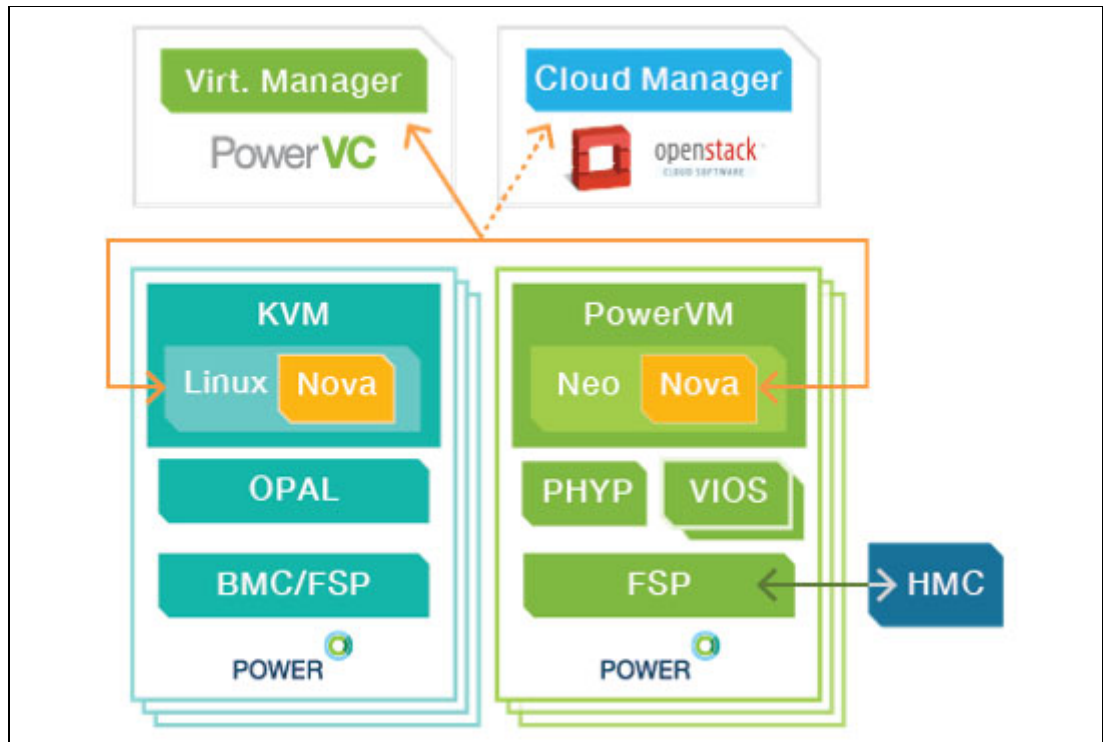


Figure 4-2 IBM PowerVC integration into NovaLink -managed Power Systems environments

IBM PowerVC APIs are built on OpenStack and provide Power Systems customers an integration point for a multi-cloud infrastructure.

At the time of writing, the POWER9 processor-based virtualization and cloud software stack includes the following elements:

- ▶ Hypervisor: PowerVM V3.1.1
- ▶ Virtualized I/O: VIOS V3.1.1 based on AIX 7.2 TL4
- ▶ Virtualization Management: NovaLink V1.0.0.16
- ▶ Hardware Management: HMC/vHMC V9R1M940
- ▶ Private Cloud Management: IBM PowerVC V1.4.4
- ▶ Multicloud Management: vRealize Operations V7.5

4.2.3 IBM PowerVC features

At the time of writing, IBM PowerVC V1.4.4 is the latest release, which introduces the following features:

- ▶ IBM i license key injection.
- ▶ Initiator storage tagging.
- ▶ Enables *pinning* a VM to a specific host.
- ▶ Inactive (*cold*) migration.
- ▶ Image sharing between IBM PowerVC projects.
- ▶ LPM VMs fallback to the original system after evacuation.
- ▶ Dynamically add IBM PowerVC created VMs to the HMC user resource role.
- ▶ Restricted admin role (new Admin Assistant role with no deletion access).

- ▶ FlexVolume Driver support for Red Hat OpenShift.
- ▶ NovaLink V1.0.0.16: Support for multi-volume attachment.

4.3 IBM Cloud Automation Manager

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

The key features of Cloud Automation Manager are:

- ▶ The cloud infrastructure templates use Terraform.
- ▶ You can use template parameters to describe how each Terraform variable is displayed and used, for example, if a parameter is a password or not.
- ▶ Cloud Automation Manager includes an orchestration engine that provides a language to describe the orchestration flow that is needed to deploy and manage services.
- ▶ Cloud connections that describe the necessary connection parameters to connect to supported clouds.
- ▶ The *content run time*, which is a collection of Chef Infra, software repository, and the pattern manager, which runs on a series of Docker containers. The content run time may be deployed from a Terraform template.
- ▶ After it is deployed, Cloud Automation Manager provides an interface to manage day 2 operations by using the Terraform Plan/Apply feature.

In this section, we describe how to use Cloud Automation Manager to deploy AIX partitions by using IBM PowerVC that uses Terraform templates, which provide you with the building blocks to support the lifecycle 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 that is shown in Example 4-2 on page 115 can be used to deploy an AIX partition to IBM PowerVC by using Terraform directly from the command-line interface (CLI) after configuring an OpenStack provider in your Terraform templates. To configure the OpenStack provider, you must use the same details that are described in Table 4-2 on page 108.

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

As a best practice, use a more secure password for your production systems.

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

```
provider "openstack" {
  user_name = "camsvc"
  tenant_name = "ibm-default"
  password = "abcd1234"
  auth_url = "https://9.47.76.108:5000/v3"
  domain_name = "Default"
  region = "RegionOne"
```



```
    insecure = true
  }
```

To deploy AIX by using the Terraform engine directly, you must obtain and install the Terraform CLI on a system with connectivity to your IBM PowerVC environment.

You can download and learn more about Terraform at [Terraform by HashiCorp](#).

4.3.2 Configuring a cloud connection

To enable Cloud Automation Manager to communicate with IBM PowerVC, you first must create a cloud connection to IBM PowerVC. Cloud connections specify the credentials and configuration to deploy workloads to a cloud provider. Creating the cloud connection is a one-time step per IBM PowerVC environment to which you deploy.

Table 4-1 shows some sample information for the 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 pre-created a user that is named camsvc to use as our IBM PowerVC user for the cloud connection rather than a user with full administrative privileges. For more information about the configuration and setup of IBM PowerVC and useful OpenStack commands, see *IBM PowerVC Version 1.3.2 Introduction and Configuration*, SG24-8199.

Table 4-1 Sample IBM PowerVC details

| Configuration item | Value |
|--------------------|-------------|
| IP address | 9.47.76.108 |
| Domain | Default |
| Region | RegionOne |
| Project name | ibm-default |
| User ID | camsvc |
| Password | abcd1234 |

To create a cloud connection to IBM PowerVC, complete the following steps:

1. Log in to Cloud Automation Manager and select **Manage** → **Cloud Connections**, as shown in Figure 4-3. Then, select **Create Connection**.

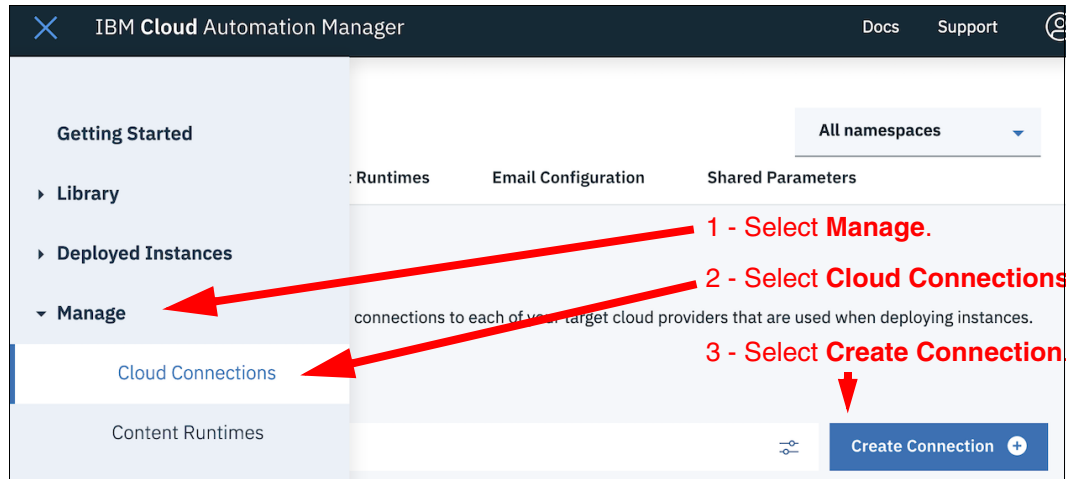


Figure 4-3 Creating a cloud connection

In the next window, you are prompted for your cloud connection details to allow Terraform to communicate with the instance of IBM PowerVC, as shown in Figure 4-4 on page 109.

Based on your IBM PowerVC installation details, you must complete the configuration window to suit your environment. Update the authentication URL's IP address to your IBM PowerVC instance's IP address, including the port and v3 because they are where the API for keystone identity API is.

Table 4-2 shows the required values that you enter 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 4-2 Cloud Automation Manager cloud connection for IBM PowerVC connectivity

| Field | Connection configuration input |
|------------------------|--|
| Cloud Provider | OpenStack |
| Connection Name | <Name of IBM PowerVC connection> |
| Connection Description | <Description of connection> |
| Authentication URL | https://<IP address of IBM PowerVC instance>:5000/v3 |
| Username | <User in IBM PowerVC to log in as> ^a |
| Password | <Password of IBM PowerVC user> |
| Domain Name | Default |
| Region | RegionOne |
| Project Name | ibm-default |

a. This user ID must have the appropriate privileges within IBM PowerVC to perform the deployment.

2. Select the **Cloud Provider** and then enter a connection name description, as shown in Figure 4-4.

IBM Cloud Automation Manager Docs Support

← Cloud Connections

Create Connection

Create cloud connections to deploy templates and services to target cloud providers. * indicates a required field

1. Select a Cloud Provider
 - * Cloud Provider
 - OpenStack
2. Select a Namespace
 - Assign Access
 - Make Connection Globally Accessible ⓘ
 - Make Connection part of a namespace
 - * Select a namespace
3. Enter Connection Name
 - * Connection Name ⓘ
 - PowerVC_Test
4. Connection Description
 - * Connection Description
 - PowerVC Test Environment

1 - Select OpenStack.

2 - Enter a name for your connection.

3 - Enter a description for your connection.

Figure 4-4 Selecting your cloud provider for a cloud connection

3. A window opens and shows the remainder of the fields that you must complete to finish configuring your connection based on your environments configuration. An OpenStack connection configuration is shown in Figure 4-5.

5. Configure Connection

How to configure an OpenStack cloud 1 - Enter the IBM PowerVC authentication URL.

* Authentication URL ⓘ
https://9.47.76.108:5000/v3

* User Name ⓘ 2 - Enter the IBM PowerVC user ID.
camsvc

* Password ⓘ 3 - Enter the IBM PowerVC user password.
.....

Domain Name ⓘ 4 - Enter the OpenStack domain.
Default

Region ⓘ 5 - Enter the OpenStack region.
RegionOne

Project Name ⓘ 6 - Enter the OpenStack project name.
ibm-default

CA Certificate ⓘ
Enter CA Certificate

Client Certificate ⓘ 7 - Click **Create**.
Enter Client Certificate

Client Private Key ⓘ
Enter Client Private Key

Create

Figure 4-5 Entering the configuration details for the OpenStack connection

4. After clicking **Create**, you see a final confirmation window to save the connection, as shown in Figure 4-6.

Success!

Your cloud connection details are valid.

Click "Save."

Edit Save

Figure 4-6 Saving your cloud connection

4.3.3 Creating a template for AIX deployment

The fundamental component for the deployment of an image within Cloud Automation Manager 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 (LPARs) deployed through IBM PowerVC by providing information, such as:

- ▶ A storage template to use
- ▶ A compute template to use
- ▶ A host group to which to deploy
- ▶ Network adapters to attach
- ▶ Extra disks to allocate

Terraform templates can also be used to trigger provisioners such as the **remote-exec** provisioner to perform more 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 that is named the Cloud Automation Manager Template Designer. For more information about how to use this tool, see [IBM Knowledge Center](#).

In this section, we 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 an URL internal to the organization

To create the template after logging in to Cloud Automation Manager, complete the following steps:

1. Open the menu bar from the upper left of the interface and select **Library** → **Templates** and then select **Import Template**, as shown in Figure 4-7.

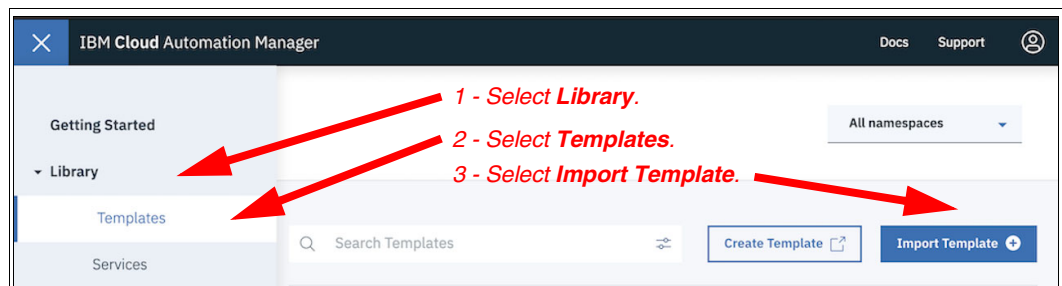


Figure 4-7 Creating a template from scratch in Cloud Automation Manager

2. After you click **Import Template**, the Import Template dialog box opens, where you select the import source for the template, name, description, and type of cloud provider this template is used for. Because we want to use this template to create an AIX partition that uses IBM PowerVC and want to start with an empty template, we must ensure that we select the template source as **From Scratch** and select the Cloud provider as **OpenStack**, as shown in Figure 4-8.

The screenshot shows the 'Import Template' dialog box with the following fields and annotations:

- 1 - Set source to From Scratch.** Points to the 'Import template source' dropdown menu.
- 2 - Select a name for your template.** Points to the 'Name' input field containing 'BasicAIX'.
- 3 - Enter a description for your template.** Points to the 'Description' text area containing 'Basic AIX Template'.
- 4 - Select OpenStack.** Points to the 'Cloud Provider' dropdown menu.
- 5 - Click Import.** Points to the 'Import' button.

Figure 4-8 Entering template metadata

3. After you click **Import**, your new template is loaded, and a new template window opens. From this window, you can select **Manage Template**, which you can use to add to the view the current versions of templates, modify parameters, and access an inline editor to perform updates through the web interface. In this window, we can edit our template and add initial content. To do the initial edit, you must select the submenu for the v1.0.0 template and select **Edit**, as shown in Figure 4-9.

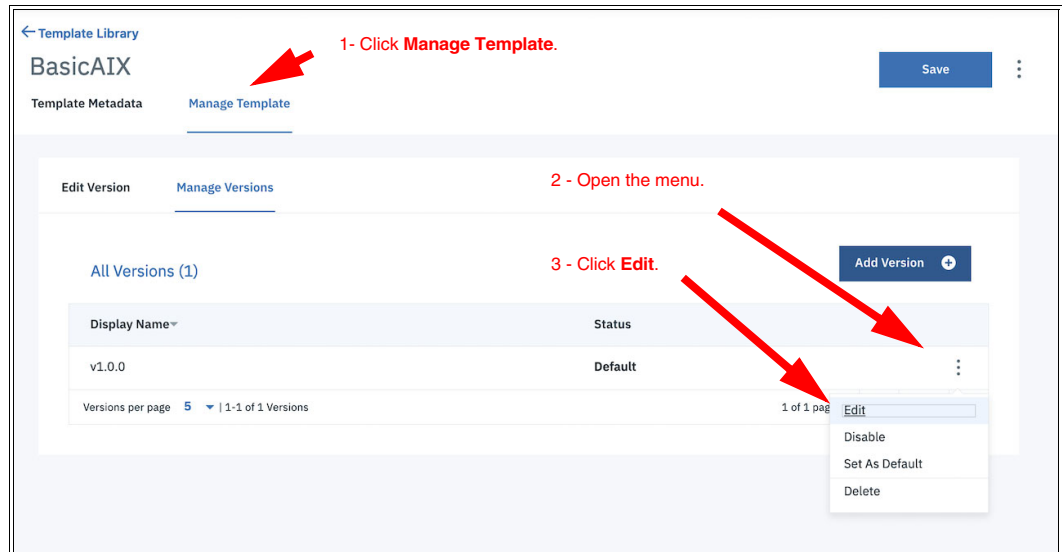


Figure 4-9 Opening the inline editor page

4. In the **Edit Version** window, you see three fields where you can update the name of the template version, change where to import the data from, and enter your Terraform template. To create a basic template, we enter our source directly into the form on this page, as shown in Figure 4-10. You can also import the template from a file, upload the file from your browser, or provide a URL from which Cloud Automation Manager pulls the file.

Edit Version

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.

```
1 {}
```

Figure 4-10 Adding your Terraform template

A basic Terraform template to deploy an LPAR uses the `openstack_compute_instance_v2` resource. A sample Terraform “all in one” template is shown in Example 4-2 on page 115, which creates a single LPAR with a single user-defined network and a data volume. A provisioner to run a local-exec to sleep for 600 seconds is included in this template to provide the new partition to establish an Resource Monitoring Control (RMC) connection before allocating its data volumes. By using variables in Terraform, we can allow for user input within Cloud Automation Manager.

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

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

Example 4-2 Terraform template to create an IBM PowerVC partition

```
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 115, the Terraform template is represented as a single file because of the inline editor that we are using to enter the information into Cloud Automation Manager. This template may be split into separate files if it is imported from another source. For more information about the standard Cloud Automation Manager structure, see [IBM Knowledge Center](#).

5. After inserting the template that is shown in Example 4-2 on page 115 into the field that is shown in Figure 4-10 on page 114 and clicking **Update**, you enable the parameter inputs within Cloud Automation Manager for this template. To do this task, go to the **Manage Template** window, click **Edit Version**, and click **Update Parameters**, as shown in Figure 4-11.

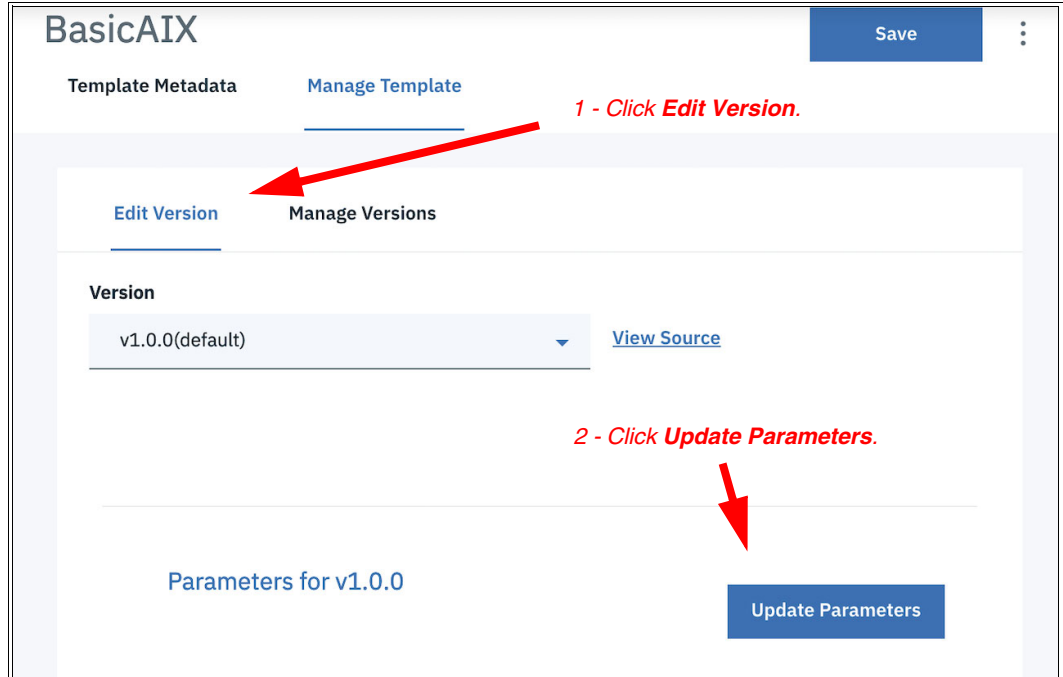


Figure 4-11 Parameter window

6. On the **Update Parameters** window, select **From Template Source** as the import source. This action automatically generates your parameter listing based on the variables in your Terraform template. Click **Update** before leaving this window, as shown in Figure 4-12.

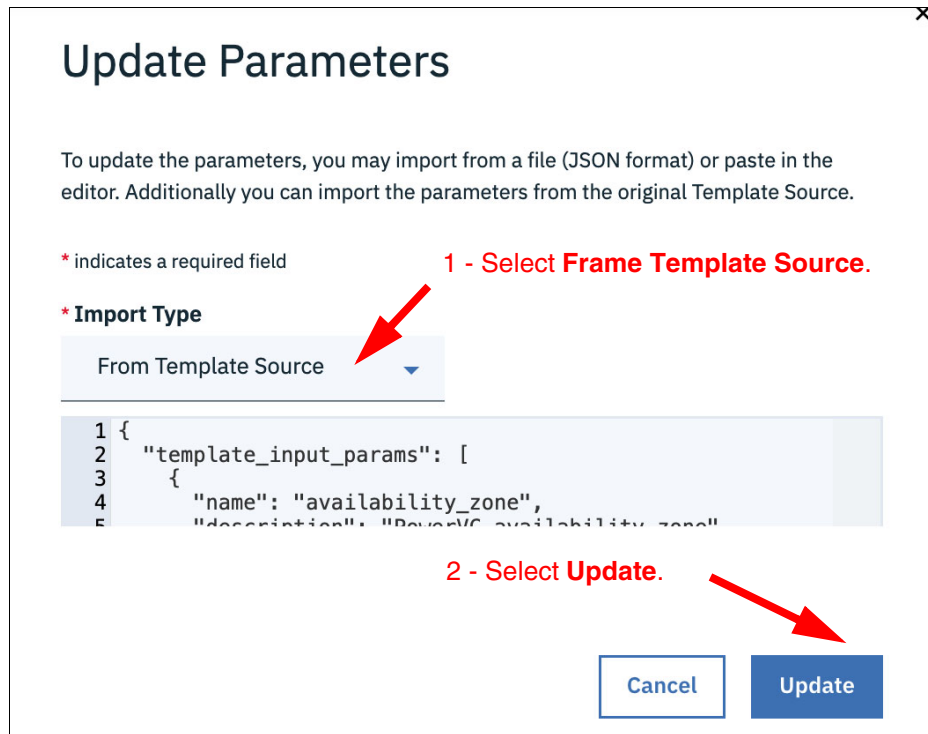


Figure 4-12 Generating parameters for your template

After you generate the parameters, you 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 is created, it can be used in services or deployed as is. Services can be used to add extra business logic that is required for deployments.

4.3.4 Deploying an AIX partition by using a template

1. To deploy an AIX partition by using a template, go to your newly created template by selecting **Library** → **Templates** and selecting the template that you want to use, as shown in Figure 4-13.

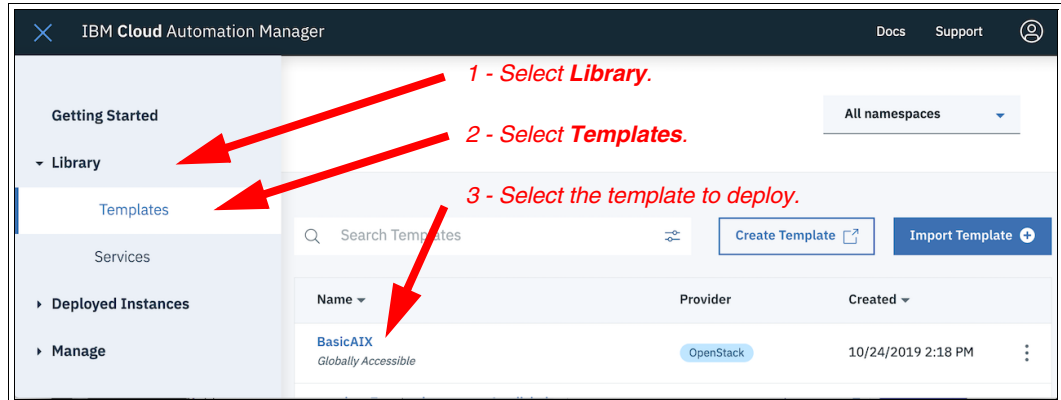


Figure 4-13 Selecting a template to deploy

2. After selecting the template to deploy, click **Deploy** at the lower right of the window. Enter the details of the deployment, which includes the **Namespace** that you want to deploy to in Cloud Automation Manager and an **Instance Name** for the specific deployment of the template. In this example, we use the **services** namespace and select **AIXbasic** as the instance name, as shown in Figure 4-14. For the **Cloud Connection** field, select the connection that matches the name you that created when adding the IBM PowerVC OpenStack provider.

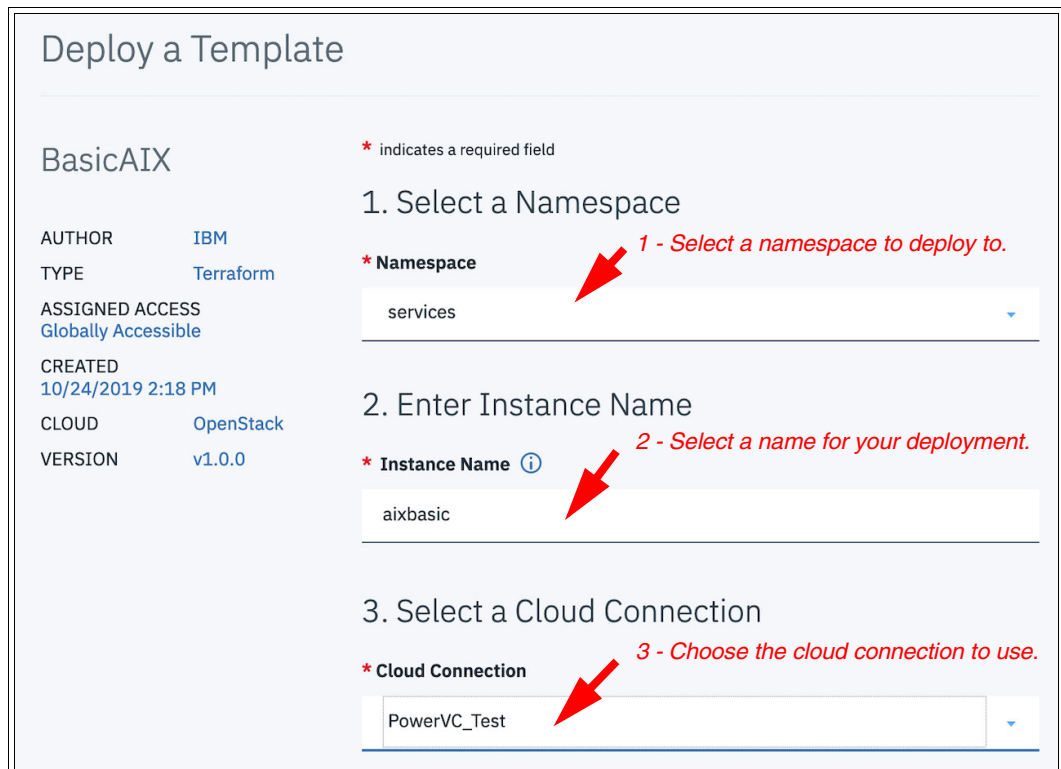


Figure 4-14 Entering the basic details for instance deployment

3. After you finish, you can see which cloud connections are available to the namespace that you selected. You see the list of extra parameters that are required by the template. For the template in Example 4-2 on page 115, complete the following fields:

| | |
|-------------------|--|
| availability_zone | References the host group within IBM PowerVC to which to deploy. |
| datavg_size | Size of the additional disk to allocate in gigabytes. |
| flavor_name | Name of the Compute Template to deploy. |
| hostname | Host name of the partition to deploy. |
| image_name | The image name from IBM PowerVC for the image you want to deploy. |
| network_name | The name of the network in IBM PowerVC that you want to which to deploy. |
| volume_type | The Storage Template in IBM PowerVC that you want to use for your data volume. |

Note: The OpenStack provider in Terraform does not allow you to specify a storage connectivity group to which to deploy. 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, log in to your IBM PowerVC server and set the `powervm:storage_connectivity_group` key by running the following command:

```
nova flavor-key medium-scg1 set powervm:storage_connectivity_group="<ID of storage connectivity group>"
```

4. After you enter all the details, click **Deploy**, as shown in Figure 4-15.

The screenshot shows a form titled "4. Additional Options" with the following fields and values:

- * availability_zone**: Default Group
- * datavg_size**: 10
- * flavor_name**: tiny
- * hostname**: aixlpar
- * image_name**: AIX_7.2
- * network_name**: VLAN2230
- * volume_type**: storwize-v7000 base template

At the bottom right, there are two buttons: "Cancel" and "Deploy".

Red arrows point from a central point to each of the seven input fields. A red arrow also points from the text "2 - Click Deploy." to the "Deploy" button. Another red arrow points from the text "1 - Complete all the additional options." to the "Deploy" button.

Figure 4-15 Entering more details for instance deployment

After you initiate the template deployment, the build process starts. If the build encounters any errors, they are shown in the Log File window. On successful deployment, you see a window similar to Figure 4-16. This window shows 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, the template version that was used for the deployment, and the IP address of the newly created partition.

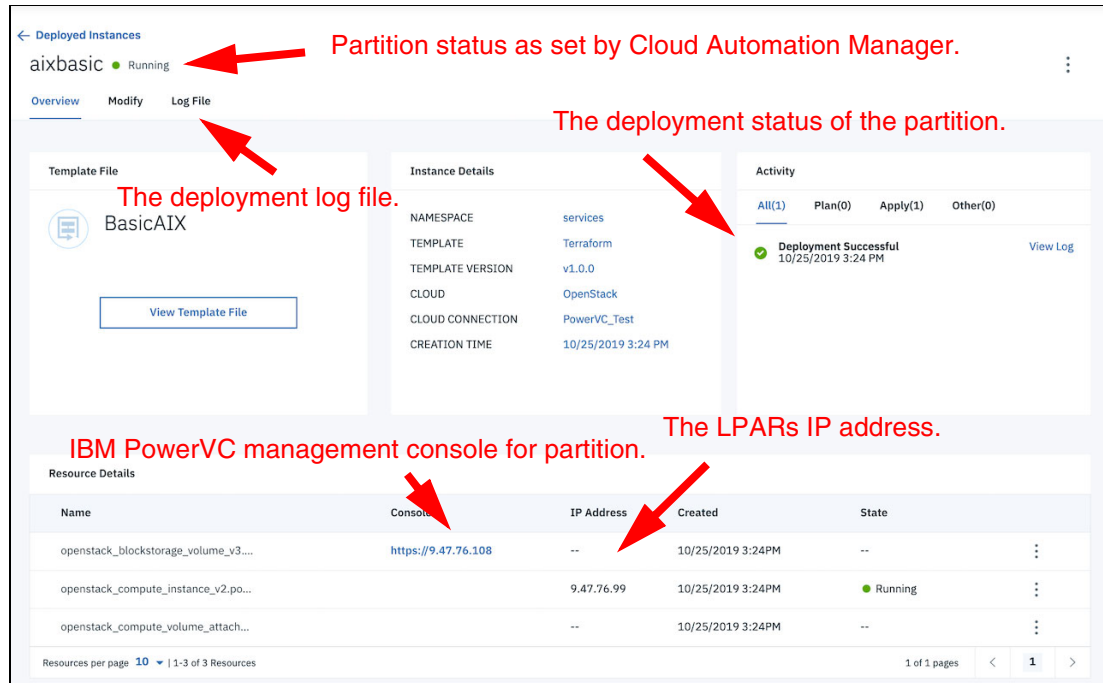


Figure 4-16 Successful instance deployment

After deploying the workload, you can delete the workload when it is no longer required, and power on and off the workload.

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

A list of Terraform supported provisioners can be viewed on the [Terraform website](#).

Cloud Automation Manager also comes with a selection of content providers. For more information, see [IBM Knowledge Center](#).

4.4 Ansible automation and AIX

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

Ansible communicates with your AIX based system securely by using the OpenSSH protocol. Ansible can use either password-based authentication or SSH key-based authentication.

With Ansible, you define the state of a system and allow Ansible to make changes to match the needed state, for example, ensuring that a certain file set is installed or an attribute of a particular AIX device is set to the needed value.

To learn more about Ansible, see the [Ansible website](#).

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

4.4.1 Installing Ansible on an AIX control node

The Ansible engine can be installed on various platforms, including AIX.

For AIX based systems, the Ansible engine is available as part of the [AIX Toolbox for Linux Applications](#).

If you configured a YUM package to manage AIX, you can simply run `yum install ansible`, as shown in Example 4-3.

Example 4-3 Installing Ansible by using YUM on AIX

```
# 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:
  gmp.ppc 0:6.1.2-1                libgcc.ppc 0:8.1.0-2                libstdc++.ppc
0:8.1.0-2                python-babel.noarch 0:0.9.6-1                python-httplib2.noarch
0:0.9.2-1                python-jinja2.noarch 0:2.10.1-1                python-keyczar.noarch 0:0.716-1
```

```
python-markupsafe.ppc 0:1.0-1 python-paramiko.ppc 0:1.7.6-1
python-pyasnl.noarch 0:0.2.3-1 python-pycrypto.ppc 0:2.6.1-1 sshpass.ppc
0:1.06-2
```

```
Complete!
# ansible --version
ansible 2.7.0
  config file = /etc/ansible/ansible.cfg
  configured module search path = [u'/.ansible/plugins/modules',
u'/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 more information about how to install and configure YUM on AIX, see [IBM Software](#).

4.4.2 AIX specific Ansible modules

As of Ansible V2.8, it has the modules that are shown in Table 4-3 that are available for use specifically for AIX. These modules are not the complete subset of modules that are available in Ansible, but only a small portion that is AIX specific.

Table 4-3 AIX specific Ansible modules that are available since Ansible V2.8

| Module | Minimum Ansible version | Description |
|----------------|-------------------------|---|
| aix_devices | 2.8 | Used to rescan a device, remove a device, or modify a device's attributes or state. |
| aix_filesystem | 2.8 | Used for creating, removing, or extending AIX file systems. |
| aix_inittab | 2.3 | Used for creating, removing, or changing AIX inittab entries. |
| aix_lvg | 2.8 | Used for creating, removing, or extending AIX volume groups (VGs). |
| aix_lvol | 2.4 | Used for creating, removing, or resizing AIX logical volumes. |
| installp | 2.8 | Used for installing and removing AIX packages. |
| mksysb | 2.5 | Creates an mksysb image of an AIX system. |

Additionally, third-party custom modules that are created by the open source community for AIX are available at the [AIXOSS GitHub project](#).

These custom Ansible modules can be used to perform more AIX and VIOS maintenance tasks, such as:

- ▶ Performing automatic downloads of fix packs by using Service Update Management Assistant (SUMA).
- ▶ Performing updates, restarting, and checking the status of Network Installation Manager (NIM) clients.

- ▶ Check for vulnerabilities on AIX systems by 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 that operates on a server/client model. Chef Infra client agents are available for AIX 7.1 and AIX 7.2. By using Chef Infra, you can turn your infrastructure into code and deploy environments in a testable and repeatable manner. Chef Infra can also be used for ongoing maintenance of your environment by 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 by using the HTTPS protocol. Using a pull model requires 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 many built-in resources that support AIX, Chef also has many community-contributed modules in the Chef supermarket. Regarding AIX, the open source community has contributed many custom resources that can be used during Chef cookbook development to simplify your code and manage your AIX environment. These custom resources are available on the Chef Supermarket and can be used as part of your development efforts. Custom resources exist for most aspects of management for AIX.

For more information about the AIX cookbook with custom resources, see the [Chef Supermarket](#).

For more information about Chef Infra, see [An Overview of Chef Infra](#).

4.6 Puppet Enterprise on AIX

Puppet Enterprise 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 Enterprise is written in a declarative language in which you specify the needed state, and Puppet Enterprise performs the steps that are required to meet that state.

Puppet Enterprise operates on a client/server model, but you can also operate a client in stand-alone mode. Puppet Enterprise supports running the Puppet Enterprise agent on AIX 6.1, 7.1, and 7.2.

For more information, see [Puppet Enterprise](#).



IBM AIX and IBM PowerVM features

This chapter provides a short overview of the wide range of features of IBM Power Systems servers, IBM PowerVM Hypervisor, and Virtual I/O Server (VIOS), and it explains briefly how they are supported and integrated with AIX.

This chapter describes the following topics:

- ▶ Storage access
- ▶ Network access
- ▶ Dynamic LPAR support
- ▶ Virtual processors
- ▶ Simultaneous multi-threading and logical processors
- ▶ Active System Optimizer and Dynamic System Optimizer
- ▶ Active System Optimizer and Dynamic System Optimizer
- ▶ Shared storage pools
- ▶ PowerVM NovaLink
- ▶ Power Enterprise Pools
- ▶ Linux on Power
- ▶ Virtual I/O Server enhancements

5.1 Storage access

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

AIX has *native* access to the following physical storage devices:

Direct Attached Storage Devices (DASDs)

These are physical storage devices that are directly attached to Power Systems servers by using a wide range of disk adapters, controllers, enclosures, and I/O drawers. They can be SCSI, SAS, SATA, solid-state drive (SSD), or flash devices. Some of these devices exceed 1 TB, 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 that are in external storage systems that are manufactured by IBM or other storage manufacturers. Access is possible by using various protocols, such as Fibre Channel (FC), Fibre Channel over Ethernet (FCoE) or iSCSI. FC adapters that are used to access SAN storage systems can have four ports and a speed of 32 Gbps.

Tape devices

These are tape libraries that are manufactured by IBM or other third-party manufacturers. Access is possible by using various protocols, such as FC or SAS.

Optical devices

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

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

Virtual SCSI (vSCSI)

This method provides standard SCSI-compliant access by using vSCSI interfaces. You can use this method to access various types of disk storage systems, tape libraries, and optical devices.

Virtual Fibre Channel

This method provides standard FC-compliant access by using virtual FC interfaces. You can use this method to access various types of disk storage systems and tape libraries.

iSCSI

This method provides standard SCSI-compliant access over Ethernet interfaces. You use this method to access various types of IP-based disk storage systems.

Both vSCSI and virtual FC 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 IBM POWER Hypervisor (PHYP) work together to ensure that each logical partition (LPAR) has access only to its own data. VIOS controls the data flow and performs direct memory access (DMA) operations to transfer data between logical memory ranges that are assigned to different partitions; the PHYP 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 that are available to get network access.

5.2.1 Dedicated adapters

AIX has *native* support for physical Ethernet network adapters that have four ports and speeds of 1, 10, 25, 40, and 100 Gb.

5.2.2 Virtual Ethernet adapters

With this type of virtual adapter, AIX LPARs can send and receive network traffic without having a physical Ethernet adapter. The PHYP provides a software implementation of an Ethernet switch that is compliant with standard IEEE 802.1Q. This switch allows operating systems (OSs) running in LPARs to communicate by using standard networking protocols.

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

5.2.3 VIOS Shared Ethernet Adapter

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

5.2.4 Virtual Network Interface Cards

This type of virtual adapter allows AIX LPARs to send and receive network traffic by using virtual functions that are incorporated in single root input/output virtualization (SR-IOV) capable physical network adapters. This advanced type of virtual adapter supports quality of service (QoS), and you can define a maximum of six backing physical adapters.

AIX can have *redundant* network access by using the methods that are described in the following sections.

Network Interface Backup

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

Etherchannel 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 has the network bandwidth of all aggregated adapters and continues to be operational if individual adapters fail.

5.3 Dynamic LPAR support

With dynamic LPAR (DLPAR), Power Systems servers can dynamically add and delete selected system resources from AIX LPARs while they are running. These resources include processor cores, memory, physical I/O adapters, and virtual adapters, such as vSCSI adapters or virtual FC adapters. Adding and removing resources from AIX LPARs can be performed *without* restarting the LPARs or the AIX OSs running in LPARs.

The minimum amount of memory that can be added or removed depends on the Logical Memory Block size that is used by the Power Systems server hosting the LPAR.

Virtual I/O adapters that can be added or removed can be vSCSI, virtual FC, and Virtual Ethernet adapters.

5.4 Virtual processors

Virtual processors introduce an abstract layer between the OS and the physical cores. The OSs perceive only virtual processors and act like they have physical cores that are available. Processes that are run by the OS are assigned to virtual processors, which are dispatched by the PHYP on actual physical cores. The PHYP uses virtual processors to encapsulate all data that is relevant to the LPARs' state from a processor core perspective at a specific point.

Virtual processors represent a key component of IBM PowerVM that enables Power Systems servers to support processor core sharing and IBM Micro-Partitioning®. Both dedicated and micropartitions use virtual processors.

PHYP uses a paravirtualization strategy, which means that OSs are enhanced to become hypervisor-aware and use a well-defined set of interfaces. AIX OS is enhanced to benefit from all capabilities that are provided by the hypervisor.

5.5 Simultaneous multi-threading and logical processors

Power Systems servers 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 OS by using the `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, and lesser threads per core lead to faster thread execution, which leads to improved application response time.

For IBM POWER9 processor-based systems running AIX 7.2 Technology Level (TL) 3, the default SMT is SMT8, which allows a maximum of 1536 threads in a full-system LPAR.

5.6 Active System Optimizer and Dynamic System Optimizer

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

DPO determines an optimal resource placement strategy for Power Systems servers that is based on the partitions' configuration and hardware topology on the system. It then performs a sequence of memory and processor relocations to the LPAR affinity and overall system affinity. This process occurs dynamically while the partitions are running.

Starting with versions AIX 6.1 TL 8 and AIX 7.1 TL 2, AIX is a DPO-aware OS.

For more information about DPO, see *IBM PowerVM Virtualization Managing and Monitoring*, SG24-7590.

5.7 Shared storage pools

A *shared storage pool* (SSP) is a pool of SAN storage devices that can be used among VIOSs. It is based on a cluster of VIOSs and a distributed data object repository with a global namespace. Each VIOS that is part of a cluster represents a cluster node.

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

SSPs provide the following benefits:

- ▶ Improve the usage of available storage.
- ▶ Simplify administration tasks.
- ▶ Simplify the aggregation of many disks among the VIOSs.

On VIOS Version 2.2.0.11 Fix Pack 24 Service Pack 1 and later, you can create SSPs. SSPs provide distributed storage access to all VIOS LPARs in a cluster.

Note the following considerations for using SSP for VIOS:

- ▶ 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 IPv6 address that is configured on the VIOS LPAR.

Thus, a cluster consists of up to 16 VIOS LPARs with an SSP that provides distributed storage access to the VIOS LPARs in the cluster. Each cluster requires a separate repository disk and SSP disks. The SSP can be accessed by all the VIOS LPARs in the cluster. All the VIOS LPARs within a cluster must have access to all the physical volumes (PVs) in an SSP.

SSP for VIOS simplifies cloud management and efficiency of storage usage. Starting with PowerVM V2.2.6, it includes a VIOS SSP performance enhancement that is known as *flash acceleration*. This feature can transparently increase a client's workload performance by using SSP flash storage caching on the VIOS.

In VIOS V3.1, 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 cannot access the file system of the SSP cluster pool while the PostgreSQL database process is performing an I/O operation, the PostgreSQL database stops all operations and generates a 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.

5.7.1 SSP consideration and procedures

The following section describes all the considerations to account for and the procedures for an SSP setup.

Before you create SSPs, ensure that all LPARs are preconfigured by using the Hardware Management Console (HMC). Here 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 LPARs, you must consider the following items:

- ▶ There must be at least one CPU and one physical CPU.
- ▶ The VIOS LPARs must be configured as VIOS LPARs.
- ▶ The VIOS LPARs must consist of at least 4 GB of memory.
- ▶ The VIOS LPARs must consist of at least one physical FC adapter.
- ▶ The rootvg device for a VIOS LPAR cannot be included in storage pool provisioning.
- ▶ The associated rootvg device must be installed with VIOS V2.2.2.0 or later.
- ▶ The VIOS LPAR must be configured with enough virtual server SCSI adapter connections that are required for the client LPARs.
- ▶ The VIOS LPARs in the cluster require access to all the SAN-based PVs in the SSP of the cluster.
- ▶ The VIOS LPAR 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 SSPs, the SEA must be in threaded mode.

Client logical partitions considerations

The client LPARs should meet the following considerations:

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

Network addressing considerations

For the networking, you must ensure stable connectivity among the VIOS LPARs and the client LPARs. Consider the following items:

- ▶ Uninterrupted network connectivity is required for SSP operations. The network interface that is used for the SSP configuration must be on a highly reliable network that is not congested.
- ▶ Ensure that both the forward and reverse lookup for the host name that is used by the VIOS LPAR for clustering resolves to the same IP address, either by a native hosts file or by a reliable DNS.

- ▶ The host name of each VIOS LPAR that belongs to the same cluster must resolve to the same IP address family, which is either IPv4 or IPv6.
- ▶ With VIOS V2.2.2.0 or later, clusters support IPv6 addresses. Therefore, VIOS LPARs 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 LPAR.

To change the host name of one VIOS LPAR node in an SSP cluster, delete the cluster, change the host name, and then re-create the cluster.

To change the host name for two or more VIOS LPARs in the cluster, you can remove the VIOS LPAR from the cluster and change the host name. Then, read the VIOS LPAR to the cluster again with the new host name.

In VIOS V2.2.3.0 or later, by default, the SSP cluster is created in unicast address mode. In earlier VIOS versions, the cluster communication mode is created in multicast address mode. When the cluster versions are upgraded to VIOS V2.2.3.0, the communication mode changes from multicast address mode to unicast address mode as part of the rolling upgrade operation.

When a cluster is created, you must specify one PV for the repository PV and at least one PV for the storage pool PV. The storage pool PVs are used to provide storage to the actual data that is generated by the client partitions. The repository PV 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 PVs. The repository disk must have at least 1 GB of available storage space. The PVs in the storage pool must have at least 20 GB of available storage space in total.

Each of the VIOS LPARs assigns hdisk names to all PVs that are available through the FC ports. The VIOS LPAR might select different hdisk numbers for the same volumes to the other VIOS LPARs in the same cluster. For example, the `viosA1` VIOS LPAR can have `hdisk9` assigned to a specific SAN disk, and the `viosA2` VIOS LPAR can have the `hdisk3` name that is assigned to that same disk. For some tasks, the unique device ID (UDID) can be used to distinguish the volumes. To obtain the UDID for each disk, use the `chkdev` command.

Use any method that is available for the SAN vendor to create each PV with at least 20 GB of available storage space. Map the PV to the LPAR FC adapter for each VIOS in the cluster. The PVs must be mapped only to the VIOS LPARs that are connected to the SSP.

5.7.2 SSP setup procedure

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

In our scenario, `hdisk2`, `hdisk3`, and `hdisk4` are shared disks between two VIOS LPARs. The `hdisk2` is a repository disk, and `hdisk3` and `hdisk4` are for the shared pool.

Complete the following steps:

1. Example 5-1 shows how to create an SSP cluster by running the **cluster -create** command, specifying the repository disk, pool name, and the pool disks members.

Example 5-1 Creating an SSP cluster

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

2. Create the logical unit that you will use for the client LPAR. Run the **mkbdsp** command to set the logical unit size to 10G, as shown in Example 5-2.

Example 5-2 Creating a logical unit that will be used for the client logical partition

```
VIOS1$ mkbdsp -clustername ssp1 -sp demosp 10G -bd lparA_lu1
Lu Name:lparA_lu1
Lu Udid: ad67212f2cbb000833f5dbd2822d5ae7
```

3. After creating the logical unit, map the logical unit to a virtual host (vhost) that is named vhost0. Example 5-3 shows how to map the logical unit to vhost0 with a Virtual Target Device (VTD) name.

Example 5-3 Mapping the logical unit to a virtual host with a VTD name

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

Example 5-3 shows lparA_lu1 as the LUN and lparA_rootvg as the VTD name.

4. Because this SSP cluster has only one node (vios1), you must add at least one more node to use this SSP cluster. Run the **cluster -addnode** command, 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 perform any other SSP commands and tasks from the second node.

The logical units are represented as VSCSI disks in the client LPAR side.

The common scenario is that you want to increase the logical unit size where it reflects in the LPAR VSCSi disk side.

5. In VIOS V2.2.40 and later, you can optionally increase the size of an existing SSP logical unit. Example 5-4 shows how to increase the logical unit number (LUN) from 10G to 20G.

The logical unit name is lparA_lu1, which is a rootvg LUN for a specific client LPAR.

Example 5-5 Increasing dynamically the logical unit 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 LPAR, you must run `chvg -g` command against the rootvg that has the disk that was resized. The `chvg -g` command examines all the disks in the volume group (VG) to see whether they grew in size. If any disks grew in size, the command attempts to add more PPs to the physical volume and to the VG.

For more information about the SSP setup, see [Creating Simple SSP among two VIO servers](#).

5.8 PowerVM NovaLink

PowerVM NovaLink is a Linux based software that provides on-host management capabilities for Power Systems servers. PowerVM NovaLink is available on POWER8 or higher processor-based systems. The NovaLink application can run on either a Red Hat Enterprise Linux (RHEL) 7.3 or Ubuntu Linux based IBM PowerPC® 64 Little Endian (PPC64LE) based LPAR. An appliance-based installation is also available for running an Ubuntu Linux based installation of PowerVM NovaLink. The PowerVM NovaLink 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 describe the following topics:

- ▶ The components of PowerVM NovaLink
- ▶ Software-defined infrastructure (SDI) capabilities
- ▶ Resource Monitoring Control (RMC) communication
- ▶ NovaLink and HMCs

For more information about PowerVM NovaLink, see [IBM Knowledge Center](#).

5.8.1 Components of PowerVM NovaLink

NovaLink is composed of five components that fall into two categories:

- ▶ PowerVM NovaLink Core Services, which provide communication between the NovaLink system and the PowerVM Hypervisor (PHYP) by using two features:
 - A command-line interface (CLI) for shell interaction with PowerVM and management of your PowerVM environment.
 - A Representational State Transfer (REST) application programming interface (API) for management of the system.
- ▶ OpenStack drivers that can be use 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 LPARs by using SEAs.
 - Ceilometer performance monitoring agents for providing metrics for compute, storage, and network.

The core components of PowerVM NovaLink can be updated quickly by using the package manager for the OS it is on, such as dpkg on Ubuntu Linux based installations and RPM on RHEL-based installations. Because the components of PowerVM NovaLink are packages on the OSs they are on, you can install and manage your PowerVM NovaLink systems by using monitoring and management tools of your choice. It also allows for the PowerVM NovaLink systems to be secured according to your own security requirements.

5.8.2 Software-defined infrastructure capabilities

With PowerVM NovaLink, you can abstract your network and storage capabilities by using two different operating modes. Creating a software-defined environment (SDE) is covered in detail in *Building a SAN-less Private Cloud with IBM PowerVM and IBM PowerVC*, REDP-5455. NovaLink supports two modes of SDI.

Software-defined network

Software-defined networking (SDN) provides an abstraction of network hardware by using software-based solutions. PowerVM NovaLink provides this abstraction by using industry standard Open vSwitch (OVS) technology. In this mode, PowerVM NovaLink can host some or all networks on a Power Systems server.

Software-defined environment

PowerVM NovaLink can also operate in the PowerVM I/O SDE mode. In this mode, PowerVM NovaLink takes the place of VIOSs and provides network and storage to the LPARs that are hosted on the system.

Note: At the time of writing, PowerVM NovaLink must be installed on Ubuntu Linux to use SDE.

5.8.3 Resource Monitoring Control communication

Using PowerVM NovaLink simplifies the configuration of RMC and DLPAR operations. NovaLink uses an internal virtual switch that is named *MGMTSWITCH* that runs on virtual LAN (VLAN) ID 4094 to communicate to LPARs and the VIOSs on the managed system by using IPv6 adapters on each partition with self-assigned addresses.

Inter-partition communication over the MGMTSWITCH is prevented when 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 cannot communicate to one another because traffic can be sent and received only to the vNIC that is 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.

Communicating over the internal virtual switch eliminates the requirement of having firewall to open firewall rules between the management console and the LPARs on a system.

Communication over the IPv6 MGMTSWITCH network for RMC provides virtual network interface card NovaLink with connectivity to the partitions; traditional RMC connectivity requirements still apply if you want to perform dynamic operations from your HMC through a physical network.

5.8.4 PowerVM NovaLink and Hardware Management Consoles

Although PowerVM NovaLink provides functions that are similar to an HMC in terms of LPAR management and managing your PowerVM environment, they complement each other where an environment that has them co-manage it provides the benefits of both. Unlike an HMC, PowerVM NovaLink servers are not appliances.

Table 5-1 shows how the capabilities of NovaLink and HMCs differ, but it also shows how they complement each other. By using both in co-management mode, you can modernize your AIX infrastructure with the capabilities that are offered by PowerVM NovaLink while maintaining the hardware management abilities that are offered by HMCs. To run in co-management mode, you need HMC V8.4.0 Service Pack 1 or later.

Table 5-1 PowerVM NovaLink and Hardware Management Console capabilities

| Task | PowerVM NovaLink | Hardware Management Console |
|---|------------------|-----------------------------|
| Runs on the system it manages. | Yes | No |
| Concurrent firmware update. | No | Yes |
| Disruptive firmware update. | Yes | Yes |
| Hardware monitoring. | No | Yes |
| Service agent call home. | No | Yes |
| Performs DLPAR updates. | Yes | Yes |
| Enables OpenStack connectivity. | Yes | No |
| SDN capabilities. | Yes | No |
| SDE capabilities. | Yes | No |
| Simplified RMC by using an internal virtual switch. | Yes | No |
| Redundant configuration. | No | Yes |
| Can manage more than one system. | No | Yes |

5.9 Power Enterprise Pools

This section introduces the concepts, features, and benefits of Power Enterprise Pools (PEP), a technology that shares processor cores and memory among a pool of Power Systems servers. PEP is the natural response to an increased demand for flexibility and responsiveness.

PEPs are available in two editions:

- ▶ The first edition uses the concept of *mobile capacity*,
- ▶ The second edition uses the concept of *utility capacity*.

This section briefly introduces the components and benefits for both editions.

5.9.1 Power Enterprise Pools first edition

With PEP first edition, you can benefit from resources that are on different Power Systems servers. It is centered on the concept of *resource activation*.

Power Systems servers have hardware resources, such processor cores and memory, and have processor books and memory cards. Depending on the machine type and model, a Power Systems server can have a certain number and type of cores and memory modules physically *installed*. As a result of the purchasing process, some or all of these resources can be *activated* before the system is shipped. For example, an Power Systems server might have 32 cores and 8 TB of memory that is 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 by using an *activation code*.

Activation codes are unique to each Power Systems server and are based on system machine type, model, and serial number. You can find them at [IBM Capacity on Demand \(CoD\)](#).

To activate installed and inactive physical resources, the activation code must be entered into the system by using either HMC or Asynchronous Systems Management Interface (ASMI) interface.

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

- | | |
|---------------|--|
| Static | The resources that are associated to this type of activation are activated and bound to the Power Systems server where they are activated. They cannot be moved from one system to another. |
| Mobile | The resources that are associated to this type of activation are mobile and can be activated on any Power Systems server that is part of the pool and has inactive resources. Mobile resources can float from one Power Systems server to another one and can be active on only one Power Systems server concurrently. At any point, the amount of usable resources on any individual system is the sum of static and mobile resources that are activated on the system. |
| Dark | The resources that are associated to this type of activation are physically installed in Power Systems servers, but are not activated. They are activated when mobile, static, or CoD activation codes are used. |

Each Power Systems server maintains its own history of activations. The replacement of physical parts does not affect activations. Static activations can be converted to mobile activations. There are rules that 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, so at any point an individual LPAR can use any type of resources or a mix of them.

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

The first edition of PEP distinguishes between two types of pools:

- | | |
|-----------------|--|
| Midrange | This type of pool can include IBM Power System 770+ (9117-MMD), IBM Power System E870 (9119-MME), IBM Power System E870C (9080-MME), and IBM Power System E880C (9080-MHE) systems. Systems with different clock rates and POWER processor-based technologies can be part of the same pool. |
| High-end | This type of pool can include IBM Power System 780+ (9179-MHD), IBM Power System 795 (9119-FHB), IBM Power System E880 (9119-MHE), Power E870C (9080-MME), and Power E880C (9080-MHE) systems. Systems with different clock rates and POWER processor-based technologies can be part of the same pool. |

Resource assignment and movement to different systems that are part of the same pool are controlled by an HMC that acts as a *pool master*. The master HMC manages the activation keys and knows at any point which Power Systems servers 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 servers is non-disruptive.

Power Systems servers that are included in a pool must be HMC-managed and can belong to only one pool at any point. 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 periods.

Requirements for PEP first edition

PEP first edition has the following requirements:

- ▶ Select only IBM POWER7 or IBM POWER8 processor-based systems can be included in midrange or high-end pools.
- ▶ There is a minimum number of static processor activations on each system in the pool:
 - Minimum of four static processor activations for Power 770 systems.
 - Minimum of four static processor activations for Power 780 systems .
 - Minimum of eight static processor activations for Power E870 systems.
 - Minimum of eight static processor activations for Power E880 systems.
 - Power 795 systems require a minimum of 24 or 25% of the installed processors (whichever is larger) to be statically activated.
- ▶ There are minimum amounts for memory that is installed and activated:
 - Fifty percent of installed memory must be active.
 - Twenty-five percent of the active memory must be static.
- ▶ Memory activation is done in increments of 100 GB.

For more information about this version of PEP and its requirements, see *Power Enterprise Pools on IBM Power Systems*, REDP-5101.

Enabling PEP first edition

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

1. Power Systems servers that are eligible for PEP are ordered with static activations and mobile activations. Mobile activations will *initially ship* as static.
2. The customer signs a PEP contract to allow the definition of a pool.

3. The customer signs an addendum to the contract to allow adding or removing Power Systems servers from an existing pool.
4. The customer orders a miscellaneous equipment specification (MES) to add the PEP 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 were converted into mobile.
7. The customer enters the deactivation codes on each Power Systems server.

5.9.2 Power Enterprise Pools second edition

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

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

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

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

Resource usage in terms of processor and memory is monitored at the system level every minute and then aggregated at the pool level. Resource usage at the pool level is metered and any usage that exceeds pool-based capacity is charged. There are five types of charges, four of which are core-related and one of which is memory-related.

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 OSs that are installed in the LPARs, there might 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 OSs that 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, but a valid Linux license entitlement must be acquired separately to ensure that all cores or sockets that are 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 the *actual* core usage of LPARs in the pool.

Memory usage is metered by tracking memory *assignment* to active LPARs, not by memory that is used by the OS running in the LPARs.

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

Customers must purchase in advance *capacity credits* that are 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 website. 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 the pool level based on 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

For more information about CMC Enterprise Pool 2.0, see [Power Enterprise Pools 2.0 with Utility Capacity](#).

For more information about CMC, see [IBM CMC for Power Systems](#).

Requirements for PEP second edition

PEP second edition has the following requirements:

- ▶ Only Power E980 (9080-M9S) servers are supported.
- ▶ There can be a maximum of 16 Power E980 servers in the pool.
- ▶ There can be a maximum of 500 LPARs in the pool.
- ▶ All LPARs must be shared. Dedicated and dedicated-donating LPARs are not supported.
- ▶ A minimum of 25% of installed processors must be activated by using the base processor activation.
- ▶ A minimum of 50% of installed memory that is 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 CMC.
- ▶ Capacity credits must be purchased in advance.
- ▶ Cloud Edition software is required for each base processor activation.
- ▶ The requirements for software running in LPARs are available at [IBM Power Enterprise Pools 2.0 delivers enhanced Utility Capacity for processors and memory across a collection of IBM Power E980 servers](#).

Enabling PEP second edition

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

1. Power E980 systems are ordered with base processor and memory activations and all corresponding software license entitlements.
2. The customer purchases an initial number of capacity credits.
3. The customer starts the PEP on the [IBM Elastic Storage Servers \(IBM ESS\) website](#) by using the serial number of a Power E980 system and receives a pool ID.
4. The customer connects to IBM CMC and defines a PEP by using the pool ID.
5. All installed processor and memory resources are activated on all Power E980 systems that are included in the pool, and CMC starts monitoring the pool.
6. Resources that are used in excess of the pool base are debited against the capacity credits by CMC in real time. The IBM ESS website is also updated daily.

This version of PEP cannot coexist with other CoD capabilities.

5.9.3 Comparison between PEP first and second edition

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

Table 5-2 Comparison between PEP editions

| Criterion | PEP first edition | PEP second edition |
|--|---|-----------------------|
| Power Systems servers that are supported. | Power 795 (9119-FHB) Power 770+ (9117-MMD) Power 780+ (9179-MHD) Power E870 (9119-MME) Power E880 (9119-MHE) Power E870C (9080-MME) Power E880C (9080-MHE) Power E980 (9080-M9S) | Power 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 |
| Activation types supported. | Static, Mobile | Base |
| Granularity for memory activation | 100 GB | 256 GB |
| Requires activation of a minimum number of cores? | Yes | Yes |
| Requires activation of a minimum amount of memory? | Yes | Yes |
| Management interface. | HMC | CMC |

| Criterion | PEP first edition | PEP second edition |
|--|-------------------|--------------------|
| Coexistence with CoD options? | Yes | No |
| Requires non-technical steps for enablement? | Yes | Yes |
| Payment for used resources. | Prepaid | Postpaid |
| Multiple countries? | No | Yes |
| Relocate resources from one system to another? | Yes | No |

5.10 Linux on Power

If your system is a POWER8 or later processor-based system that uses PowerVM, you can deploy AIX, Linux, and IBM i partitions. When deploying Linux partitions, you can configure either PowerPC 64 Little Endian (PPC64LE) partitions, or PowerPC 64 Big Endian (PPC64BE) partitions. AIX and IBM i support only Big Endian, but with Linux you can select based on the distribution and your application requirements. PowerVM supports mixing both Big Endian and Little Endian partitions on the same physical system so that you can run your Linux workloads with your AIX partitions.

When deploying Linux on Power, you can use the same automation tools that are mentioned in Chapter 4, “Virtualization and cloud capabilities” on page 101 to automate the deployment of your Linux environments. You can create a truly heterogeneous environment and use the benefits of PowerVM for your Linux workloads.

For more information about supported Linux distributions for POWER8 and POWER9 processor-based servers, see [IBM Knowledge Center](#).

5.11 Virtual I/O Server enhancements

VIOs are software appliances that you can use to share physical hardware resources on Power Systems servers with LPARs. This section introduces enhancements to the VIOS software with the release of VIOS V3.1.

5.11.1 Key features

With the release of VIOS V3.1, the underlying base operating system (BOS) was upgraded to AIX 7.2 TL 3. This upgrade provided the following benefits:

- ▶ VIOS V3.1 is streamlined by removing unnecessary file sets, which results in quicker updates and a smaller installation size.
- ▶ iSCSI over vSCSI support, including iSCSI multipath I/O support (MPIO).
- ▶ Multipathing enhancements.
- ▶ Simplified migrations by using the **viosupgrade** command.

5.11.2 N_Port ID Virtualization

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

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

For more information about how to use and configure virtual FC, see *IBM PowerVM Getting Started Guide*, REDP-4815.

5.11.3 iSCSI support

iSCSI provides block-level access to storage devices by carrying SCSI commands over an IP network. iSCSI is used to facilitate data transfers over the internet by using TCP, which is a reliable transport mechanism that uses either the IPV6 or IPV4 protocols.

As of VIOS V3.1, VIOS can now operate as iSCSI initiators and access iSCSI target devices on both POWER8 and POWER9 processor-based servers. iSCSI support allows iSCSI disks to be presented to LPARs as virtual disks. If you are using a POWER8 processor-based server, your systems firmware level must be at FW860.20 or later. All firmware levels for POWER9 processor-based servers provide iSCSI support. VIOS V3.1 also enables MPIO support for the iSCSI initiator.

As of VIOS V3.1.0, using iSCSI devices on VIOS has the following limitations:

- ▶ VIOS may not use iSCSI devices as boot devices.
- ▶ File-based discovery policy for iSCSI is not supported. The discovery mode must be set to odm.
- ▶ Using iSCSI volumes to create logical volume-backed devices is not supported.
- ▶ iSCSI volumes may not participate in SSPs as either repository disks or SSP disks.
- ▶ iSCSI volumes may not be used for paging devices for AMS or remote restart capabilities.
- ▶ When using an iSCSI device, virtual LPARs cannot use the `client_reserve` and `mirrored` attributes.

iSCSI volumes can be presented to client virtual servers through vSCSI. When an iSCSI target can be accessed over multiple network paths, you can use MPIO to configure multiple paths to the storage provider by presenting the disk to one or multiple VIOSs.

Figure 5-1 illustrates data flow from a iSCSI storage provider over two network paths. In this case, if iSCSI volumes were mapped to the VIOS to be presented to the virtual partition over SCSI, the client LPAR sees a single virtual disk per presented LUN with two paths. This setup provides redundant connectivity at both the VIOS level and the client LPAR.

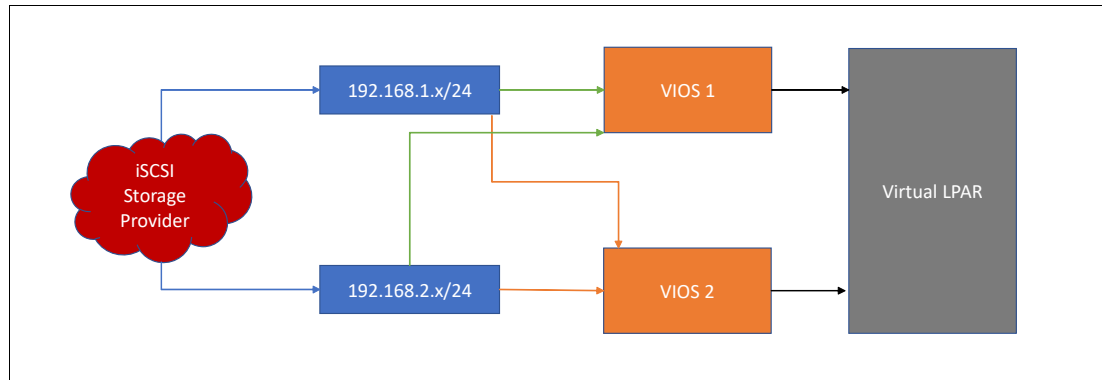


Figure 5-1 iSCSI data flow

The commands `lsiscsi`, `mkiscsi`, `chiscsi`, and `rmiscsi` can be used to manage your iSCSI devices. The iSCSI commands that are available on VIOS are the same as on AIX 7.2.

5.11.4 Upgrading Virtual I/O Server to Version 3.1

With the introduction of VIOS V3.1, a simplified upgrade procedure is available by using the `viosupgrade` command on 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, but the process is simplified by using the `viosupgrade` command. The `viosupgrade` process, when initiated on either a VIOS directly or from a NIM server, requires an unused disk to be available on the VIOS because the upgrade process initiates a `altdisk` restore of the VIOS V3.1 `mksysb` image onto the disk.

The upgrade to VIOS V3.1 is a fresh installation of VIOS V3.1 with the virtual and logical mapping configuration restored. It does not restore customized configurations for files, such as `ntp.conf` and `netsvc.conf`, or any VIOS security settings that you applied in the past. Third-party applications also are not restored, so if you are planning to upgrade, you must take steps to ensure that you back up any relevant third-party application data that you need to restore after the upgrade. In addition, if the `rootvg` of the VIOS has any logical volumes that are mapped to the LPARs, they must be migrated off the `rootvg` because the upgrade results in a clean installation of VIOS V3.1 with virtual mappings restored.

As a best practice, update your VIOS to the latest level of Version 2.2 before performing a migration to Version 3.1. In addition, you should ensure that you have the latest available images for Version 3.1 media downloaded for the upgrade. As always, it is highly recommended that you take appropriate backups of the VIOS configuration by running the `viosbr` command before performing the upgrade.

When upgrading the VIOS by using NIM, you must ensure that you create the appropriate spot (for **bosinst** upgrades only) and **mksysb** resource. To do this task, obtain the **mksysb** from the VIOS V3.1 Base Install Flash media by loop mounting the ISO and copying the file out, as demonstrated in Example 5-6.

Example 5-6 Extracting mksysb from VIOS V3.1 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
-r--r--r-- 1 root system 3561932800 Apr 27 18:16 mksysb_image
# cp /mnt/usr/sys/inst.images/mksysb_image
/export/mksysb/VI031/VI0_3.1_Flash.mksysb
```

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

Alternatively, you can create the **mksysb** image by using the two-part ISO installation media for VIOS V3.1, as shown in Example 5-7.

Example 5-7 Extracting mksysb from the VIOS V3.1 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/VI031/VIOS_3.1.0.20.mksysb
```

To generate the **mksysb** from the ISO files on a VIOS itself, run the **viosupgrade** command as follows, which generates an **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/p
admin/Virtual_IO_Server_Base_Install_3.1.0.20_DVD_2_of_2_052019.iso -w
/home/padmin/mksysb
```


To run `viosupgrade -l -i <mksysb image file> -a <install disk>` by using a VIOS, the following command shows the migration to a pair of alternative disks:

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

In cases where you are upgrading a VIOS uses SSPs, you must include the `-c` flag. You can also have particular files that you might need copied across to the upgraded cloned images by including a file list by using the `-g` parameter. For more information, see the **viosupgrade** documentation in [IBM Knowledge Center](#).

To upgrade a VIOS from NIM, you must ensure that you create an **mksysb** resource by using the VIOS V3.1 **mksysb** image. For NIM-based upgrades, there are two options when using **viosupgrade**: a **bosinst** upgrade or an alternative disk upgrade. For **bosinst** upgrades, you also must ensure that you create the appropriate SPOT by using the VIOS V3.1 **mksysb** resource. To perform the upgrade by using the alternative disk method after you defined the resources, run the `viosupgrade -t altdisk -n <VIOS> -m <mksysb resource> -a <disk list>` command. An example of the command follows, where the VIOS name is `vio1`, `hdisk2` and `hdisk3` are the target disks that you are cloning onto for the upgrade, and the **mksysb** resource name is `vios31_mksysb`:

```
# viosupgrade -t altdisk -n vio1 -m vios31_mksysb -a hdisk2:hdisk3
```

To perform a **bosinst** type upgrade, change the type to **bosint** by running the following command:

```
viosupgrade -t bosinst-n <VIOS> -m <mksysb resource> -p <VIOS 3.1 spot name>
```




Disaster recovery and high availability

This chapter presents disaster recovery (DR) and high availability (HA) solutions in IBM Power Systems environments. It introduces PowerHA and IBM VM Recovery Manager (VMR) for Power Systems, which are the typical solutions that are used to provide HA and DR capabilities for logical partitions (LPARs) running AIX.

This chapter describes the following topics:

- ▶ IBM VM Recovery Manager for Power Systems
- ▶ PowerHA

6.1 IBM VM Recovery Manager for Power Systems

IBM VM Recovery Manager for Power Systems is an enterprise-grade availability solution that provides automated recovery for virtual machines (VMs) running on Power Systems servers. VMs on Power Systems are referred to as LPARs.

Unlike operating system (OS) or application-level clustering solutions, IBM VM Recovery Manager is a VM-level technology. On Power Systems servers, VMs are treated like containers that can host various OSs. Therefore, IBM VM Recovery Manager is a OS-neutral solution. Because IBM VM Recovery Manager does not have any OS or middleware dependencies, you can use it to deploy uniform HA/DR solutions in heterogeneous environments that might include AIX, IBM i, and all Linux distributions that are supported by Power Systems servers.

There are two IBM VM Recovery Manager versions:

- ▶ VM Recovery Manager HA (VMR HA)
- ▶ VM Recovery Manager DR (VMR DR), formerly known as Geographically Dispersed Resiliency (GDR)

VMR HA

With VMR HA, you can move VMs between different Power Systems servers by using Live Partition Mobility (LPM) for planned outages or VM restart for unexpected outages. VMR HA solutions can be deployed in environments where Power Systems servers are in the same site and have access to the same network and storage devices.

During a failover, VMs on an Power Systems server are moved by LPM or restarted on a different Power Systems server. VMR HA is an availability solution that can be used in Power Systems environments where PowerHA is not used.

VMR DR

Unlike VMR HA, VMR DR is a DR solution that provides automated recovery for VMs running on Power Systems servers that are at different sites and have access to different storage devices. The distance between primary and DR sites can be several to thousands of kilometers. VMR DR relies on an out-of-band monitoring and management component and consistently uses storage replication technologies.

During the failover from the primary site to the secondary site, VMR DR orchestrates the shutdown of the VMs on the primary site, manages storage-level replication between the two sites to preserve data consistency, and starts VMs at the secondary site.

VMR HA and VMR DR can be deployed in Power Systems environments as virtualized solutions by using PowerVM, Virtual I/O Servers (VIOs) and managed by Hardware Management Consoles (HMCs). At the time of writing, VMR solutions are supported only for Power Systems environments that use HMCs. When both NovaLink and HMCs are used for management, the HMCs must be in master mode.

6.1.1 IBM VM Recovery Manager versions and lifecycle

The IBM VM Recovery Manager versions that are available at the time of writing are 1.3 and 1.4.

For more information, see the [IBM Announcement letter for VMR HA for Power Systems V1.4 and VMR DR for Power Systems V1.4](#).

6.1.2 IBM VM Recovery Manager HA components

This section provides a brief overview of VMR HA and introduces the solution's main components and capabilities.

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, and communicates and interact with HMCs, Power Systems, and VIOSs. KSYS runs on an AIX VM that is on a Power Systems server that is *not* part of the IBM VM Recovery Manager scope to ensure that it remains operational if the entire managed infrastructure fails.

VM agent

This is an agent that communicates with KSYS, and monitors the health status of the VMs and (optionally) applications running in the VMs.

GUI server

This is the GUI to the KSYS that represents a convenient alternative to the KSYS CLI. It can be used to manage both HA and DR deployments.

Hardware Management Consoles

These are the HMCs that manage the Power Systems servers. All Power Systems servers that are included in a VMR HA solution must be managed by HMCs. All HMCs are registered with KSYS and act as an intermediary that provides KSYS access to Power Systems servers, VIOSs, and that are VMs included in the scope of VMR HA solution.

Virtual I/O Server

VIOSs are used to deploy VMs on Power Systems servers and provide KSYS with all the required configuration details for all VMs. VIOSs also run a special process that is named Host Monitor (HM) that monitors the heartbeats that are sent by VMs, and intermediates the communications with KSYS.

VMR HA has the following key features:

- ▶ VM moves by using LPM for planned outages.
- ▶ Restart-based VM moves for unplanned outages.
- ▶ Host grouping for Power Systems servers with similar network and storage connectivity.
- ▶ Definition of colocation and anti-colocation policies.
- ▶ Definition of VM relocation policies.
- ▶ Host, VM, and application-level monitoring.
- ▶ Cross-VM application dependency.
- ▶ Enhanced VIOS monitoring.
- ▶ GUI and CLI-based management.

For more information and implementation guidelines for VMR HA, see *Implementing IBM VM Recovery Manager for IBM Power Systems*, SG24-8426.

VMR HA requirements

Table 6-1 includes the requirements for various VMR HA components.

Table 6-1 Requirements for VMR HA components

| VMR HA component | Requirements |
|------------------|---|
| KSYS controller | IBM AIX 7.2 with Technology Level (TL) 2 Service Pack 1 (7200-02-01) or later. |
| HMC | HMC Version 9 Release 9.1.0 or later. |
| VIOS | VIOS V3.1.0.1 or later. |
| VM OS | <ul style="list-style-type: none"> ▶ AIX 6.1 or later. ▶ Red Hat Enterprise Linux (RHEL) (Little Endian) Version 7.4 or later (kernel version:3.10.0-693). ▶ SUSE Linux Enterprise Server (Little Endian) Version 12.3 or later (kernel version -4.4.126-94.22). ▶ Ubuntu Linux distribution Version 16.04. ▶ IBM i Version 7.2 or later. |
| VM agent | <p>At the time of writing, the VM agents that are used to monitor VMs and applications can be installed only on the following OSs:</p> <ul style="list-style-type: none"> ▶ AIX 6.1 or later. ▶ RHEL (Little Endian) Version 7.4 or later (kernel version:3.10.0-693). ▶ SUSE Linux Enterprise Server (Little Endian) Version 12.3 or later (kernel version -4.4.126-94.22). |

6.1.3 IBM VM Recovery Manager DR components

This section provides a brief overview of VMR DR, and introduces the main solution components and capabilities.

Controller system (KSYS)

This is the core component of the solution that acts as a single point of control (SPOC) for the entire environment that is managed by the VMR DR solution. KSYS handles discovery, verification, monitoring, notification, and recovery operations to support DR. KSYS communicates with the HMC to collect the configuration information of the Power Systems managed systems. It also interacts with the VIOSs 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 deployed in the DR site so that it is not affected by any issues or failure at the primary site. An LPAR that hosts KSYS must run IBM AIX 7.2 TL 1 Service Pack 1 or later.

Sites

These are the sites where the Power Systems servers are located. Active VMs are in the primary site and are failed over to the DR site during the recovery process. The distance between the sites is determined by the storage replication technology.

Hosts

These are the Power Systems servers at either of the two sites. All Power Systems servers must be managed by HMCs.

Virtual machines or logical partitions

These are LPARs that are deployed on the Power Systems servers. They are failed over from the primary site to the DR site during the recovery process.

Storage agents

These agents interact with storage controllers from various storage systems at both sites to manage storage-specific operations, such as starting, stopping, suspending, reversing, resynchronizing, pausing, and resuming storage replication.

Hardware Management Consoles

These are the HMCs that manage the Power Systems servers. All Power Systems that are included in a VMR DR must be managed by HMCs. HMCs provide KSYS data about hosts, VIOSs, and VMs. Data that is collected includes information such as number of processor cores, amount of memory, and worldwide port numbers (WWPNs) of the physical Fibre Channel (FC) adapter. The HMC also checks the VIOSs capability for DR operations.

Virtual I/O Server

VIOSs are used to deploy VMs on Power Systems servers and provide KSYS all the configuration details for all VMs, especially the storage configuration details. VIOSs 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 failover rehearsal.
 - It allows for administrator initiated failover.
 - It reduces human intervention.
 - It is less prone to errors, which is especially useful in complex environments.
 - It allows for automatic discovery of changes in the entire environment, such as adding or deleting LPARs, or adding new logical unit numbers (LUNs).
 - Host grouping for Power Systems.
 - It allows for daily cross-site checks and validation.
 - It allows for automatic notifications.
- ▶ Better capacity management:
 - It allows for increases and decreases in memory and CPU entitlement for VMs that are moved to the DR site.
 - It allows for advanced cross-site pairings.
- ▶ Single administrative interface:
 - It allows for GUI and CLI-based management.
 - It allows for coexistence with VMR HA.
 - It allows for 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 HMC, VIOS, LPM, and Simplified Remote Restart (SRR) capabilities.
 - VMR DR solutions for Power Systems environments can be complemented with IBM PowerVC and PowerHA.
 - Has support for Power Enterprise Pool (PEP) first edition.
 - Has support for Elastic (On/Off) Capacity-on-Demand (CoD).

VMR DR requirements

Table 6-2 includes the requirements for VMR HA components:

Table 6-2 Requirements for the VMR DR components

| VMR DR component | Requirements |
|------------------|--|
| KSYS controller | <ul style="list-style-type: none"> ▶ IBM AIX 7.2 TL 1 Service Pack 1 (7200-01-01) or later ▶ OpenSSL for AIX V1.0.1.516 or later |
| HMC | <ul style="list-style-type: none"> ▶ HMC Version 8 Release 8.7.1 or later ▶ HMC Version 9 Release 9.1.0 or later |
| VIOS | <ul style="list-style-type: none"> ▶ VIOS Version 2.2.6.30 or later ▶ VIOS Version 3.1.0.21 or later |
| VM OS | <ul style="list-style-type: none"> ▶ AIX Version 6.1 or later ▶ RHEL (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, VMR DR supports the storage systems that are listed in the following subsections.

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 SAN Volume Controller and Storwize storage systems

VMR DR supports IBM SAN Volume Controller Version 6.1.0 and later and 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 IBM DS8700 or later and IBM 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 FlashSystem A9000

VMR DR supports the IBM XIV® Storage System and IBM FlashSystem A9000. Both Metro Mirror (synchronous) and Global Mirror (asynchronous) modes replication are supported.

6.2 PowerHA

IBM PowerHA SystemMirror® for AIX (formerly known as HACMP and now referred to as PowerHA) is the Power Systems data HA solution for applications running on AIX LPARs.

It monitors, detects, and reacts to an extensive list of events that might affect application availability. PowerHA relies on services that are provided by Reliable Scalable Cluster Technology (RSCT) and Cluster Aware AIX (CAA).

RSCT is a set of low-level OS 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 can see the same storage.

Enterprise Edition provides both local and remote replication functions. In this case, cluster nodes are in different data centers that are separated by significant distances and integrate with storage level replication services such as Copy Services, IP Replication, or IBM HyperSwap®.

6.2.2 PowerHA versions and lifecycle

The PowerHA versions and lifecycle data that is valid at the time of writing are as follows:

- ▶ PowerHA 7.2 TL1 is estimated to go out of support on 30 April 2020.
- ▶ PowerHA 7.2 TL2 is estimated to go out of support on 30 April 2021.
- ▶ PowerHA 7.2 TL3 is estimated to go out of support on 30 April 2022.

Release notes for all TLs of IBM PowerHA SystemMirror V7.2 for AIX, both Standard and Enterprise editions, are available at [IBM Knowledge Center](#).

For more information about the PowerHA SystemMirror support lifecycle, see [PowerHA SystemMirror support lifecycle information](#).

PowerHA SystemMirror V7.2 supports the latest AIX and PowerVM enhancements, such as:

- ▶ CAA
- ▶ AIX Live Update
- ▶ PEP
- ▶ LPM
- ▶ Logical Volume Manager (LVM) rootvg failure monitoring

For more information about PowerHA SystemMirror 7.2, see *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 the AIX level that is required for each PowerHA 7.2 TL.

PowerHA SystemMirror V7.2.0 for AIX is supported on the following versions of AIX:

- ▶ IBM AIX 6.1 with TL 9 with Service Pack 5 or later
- ▶ IBM AIX 7.1 with TL 3 with Service Pack 5 or later
- ▶ IBM AIX 7.1 with TL 4 with Service Pack 1 or later
- ▶ IBM AIX 7.2 with TL 0 with Service Pack 1 or later

PowerHA SystemMirror V7.2.1 for AIX is supported on the following versions of AIX:

- ▶ IBM AIX 7.1 with TL 3 with Service Pack 6 or later
- ▶ IBM AIX 7.1 with TL 4 with Service Pack 1 or later
- ▶ IBM AIX 7.2 with TL 0 with Service Pack 1 or later

PowerHA SystemMirror V7.2.2 for AIX is supported on the following versions of AIX:

- ▶ IBM AIX 7.1 with TL 4 or later
- ▶ IBM AIX 7.1 with TL 5 or later
- ▶ IBM AIX 7.2 with TL 0 or later
- ▶ IBM AIX 7.2 with TL 1 or later
- ▶ IBM AIX 7.2 with TL 2 or later

PowerHA SystemMirror V7.2.3 for AIX is supported on the following versions of AIX:

- ▶ IBM AIX 7.1 with TL 3 with Service Pack 9 or later
- ▶ IBM AIX 7.1 with TL 4 with Service Pack 4 or later
- ▶ IBM AIX 7.1 with TL 5 or later
- ▶ IBM AIX Version 7.2 with Service Pack 4 or later
- ▶ IBM AIX 7.2 with TL 1 with Service Pack 2 or later
- ▶ IBM AIX 7.2 with TL 2 or later
- ▶ IBM AIX 7.2 with TL 3 or later

For a full and detailed PowerHA compatibility matrix, see [PowerHA for AIX Version Compatibility Matrix](#).

For a full and detailed list of PowerHA known fixes, see [IBM Knowledge Center](#).

6.2.4 PowerHA licensing

This section introduces several concepts that are used by PowerHA licensing.

PowerHA licensing is core-based, which means that PowerHA must be licensed for each core that used by the cluster nodes. The licensing is enforced by proper entitlement of the LPARs.

Because they are core-based licenses for both PowerHA Standard and Enterprise Editions, the licenses depend on the Power Systems servers on which the cluster nodes run. The Power Systems servers can be divided into the following categories:

| | |
|--------------------|---|
| Small tier | IBM Power Systems S914 and S814, S922 and S822, S924 and S824, and S950 and S850. |
| Medium tier | IBM Power Systems E870, E880, and E980. |

The length of the 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. The license is packaged with an entitlement for support in the form of software maintenance (SWMA) for a specific period that must be at least 1 year. When the SWMA period expires, it must be renewed. If SWMA is not renewed, the client is still entitled to use the product but not to the benefits of SWMA.

Termed licenses This is a time-based license that may be used in cloud environments. This type of licensing bundles the license itself and SWMA for a specific period. The terms of validity may 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. If the license is not renewed, the client is no longer entitled to use the product.



IBM AIX fundamentals

This chapter explains IBM AIX fundamentals.

This chapter describes the following topics:

- ▶ Logical Volume Manager
- ▶ AIX Enhanced Journaled File System
- ▶ Role-based access control
- ▶ Encrypted File System
- ▶ AIX Security Expert and IBM PowerSC integration
- ▶ The AIX Auditing subsystem and Autonomic Health Advisor File System
- ▶ MultiBOS

7.1 Logical Volume Manager

IBM AIX partitioning is based on Logical Volume Manager (LVM). The role of LVM is to present a simple logical view of underlying physical storage space. LVM manages individual physical disks and the individual partitions on them.

7.1.1 Introduction to the LVM

This section describes the following topics:

- ▶ LVM integration with file systems
- ▶ AIX operating system (OS) calls with LVM
- ▶ LVM I/O with a storage area network (SAN)

A logical volume may correspond to a physical partition (PP). One volume may be composed of several partitions on multiple physical disks. The volumes can be extended while the OS is running and the file system is being accessed.

You might be wondering how traditional file systems like FAT or High Performance File System (HPFS) can be extended at run time. The answer is that they cannot. To take full advantage of LVM, it is necessary to use a file system that is designed for it. LVM and Journaled File System (JFS) can exist separately, but only when working in concert can both of them reach their full potential.

The AIX OS LVM calls the disks as physical volumes (PVs) that are divided into fixed areas of space named *physical partitions*. These PPs are uniform in size and span from the outer edge of the disk, moving inward toward the center of the spindle. PPs that are gathered are known as *volume groups* (VGs). 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 when the VG is created, and extra PVs that are 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 grow, the minimum size by which they expand is one PP.

Several mathematical and physical-based properties affect disk I/O, latency, and access across disk edge regions. Because of the 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 is 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 drive (HDD) heads are in the center area of the disk, where the head is most likely to pass.

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 OS. 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 *mirroring* and *striping*. With LVM, you can mirror LVs with up to three copies of the data. So, one LP can point to one or more PPs. If a hardware failure occurs, the data is preserved on other physical devices. In smaller systems that use internal storage, it is crucial that all data is mirrored to prevent unplanned outages.

Striping places data across multiple HDDs so that multiple read and write operations can occur simultaneously across many drives. You enable striping by changing the inter-PV 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 can affect the LV. By setting the policy to the maximum, LVM stripes the data across the maximum number of disks within the VG, which maximizes the number of I/O operations that can occur simultaneously.

It is never a matter of *if* a disk failure occurs, but rather a question of *when* it occurs. No disk works 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 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 must have good performance so that applications do not get held up by any I/O wait. The disk must 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 that form the LVM structure:

- ▶ Volume group
- ▶ Physical volume
- ▶ Logical volume
- ▶ File systems

These components are described in the following sections.

Volume group

A VG can be created as one of different three types:

- ▶ Small or normal volume group: A collection of 1 - 32 PVs of varying sizes and types.
- ▶ Big volume group: A collection of 1 - 128 PVs.
- ▶ Scalable volume group: A scalable VG can have up to 1024 PVs.

A PV can belong to only one VG per system; there can be up to 255 active VGs.

When a PV is assigned to a VG, the physical blocks of storage media on it are organized into PPs of a size that you specify when you create the VG.

When you install the system, one VG (rootvg) is automatically created that contains the base set of logical volumes that are required to start the system, and any other logical volumes you specify in 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 VGs. For example, the rootvg cannot be imported or exported. When performing a command or procedure on rootvg, you must be familiar with its unique characteristics:

- ▶ You create a VG by running the **mkvg** command.
- ▶ You add a PV to a VG with the **extendvg** command.
- ▶ You change the size of a PV by running the **chvg** command.
- ▶ You remove a PV from a VG by running the **reducevg** command.

Some of the other commands that you use on VGs include:

- ▶ List (**lsvg**)
- ▶ Remove (**exportvg**)
- ▶ Install (**importvg**)
- ▶ Reorganize (**reorgvg**)
- ▶ Synchronize (**syncvg**)
- ▶ Make available for use (**varyonvg**)
- ▶ Make unavailable for use (**varyoffvg**)

Small systems might require only one VG to contain all the PVs that are attached to the system. However, you might want to create separate VGs for security reasons because each VG can have its own security permissions. Separate VGs 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 PVs that is necessary for system operation.

You can move data from one PV to other PVs in the same VG by using the **migratepv** command. With this command, you can free a PV so it can be removed from the VG. For example, you can move data from a PV that is going to be replaced.

A VG that is created with smaller physical and logical volume limits can be converted to a format that can hold more PVs and more logical volumes. This operation requires that there be enough free partitions on every PV in the VG for the volume group descriptor area (VGDA) expansion. The number of free partitions that are required depends on the size of the current VGDA and the PP size. Because the VGDA is on the edge of the disk and 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 PPs are renumbered to reflect the loss of the partitions for VGDA usage. This renumbering changes the mappings of the logical volumes to PPs in all the PVs of this VG.

If you 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 VG is taken with the map option and you plan to restore by using those maps, the restore operation might fail because the partition number might no longer exist (due to reduction). As a best practice, take the backup before the conversion and after the conversion if the map option is used. Because the VGDA space increased substantially, every VGDA update operation (creating a logical volume, changing a logical volume, adding a PV, and so on) might take considerably longer to run.

Physical volume

A disk must be designated as a PV and put into an available state before it can be assigned to a VG.

A PV has certain configuration and identification information that is written on it. This information includes a PV identifier that is unique to the system.

The LVM can use the additional space that a Redundant Array of Independent Disks (RAID) can add to a logical unit number (LUN) by adding PPs to the PV that is associated with the LUN.

Logical volume

After you create a VG, you can create logical volumes within that VG.

Although a logical volume can be on noncontiguous PPs or even on more than one PV, it appears to users and applications as a single, contiguous, and extensible disk volume. You can create more logical volumes by running the `mk1v` command. With this command, you can specify the name of the logical volume and define its characteristics, including the number and location of logical partitions (LPARs) to allocate for it.

After you create a logical volume, you can change its name and characteristics by running the `ch1v` command, and you can increase the number of LPARs that are allocated to it by running the `extend1v` command. The default maximum size for a logical volume at creation is 512 LPARs, unless you specify a larger size. The `ch1v` command can override this limitation.

Logical partition

When you create a logical volume, you specify the number of LPARs for the logical volume.

An LPAR is one, two, or three PPs, depending on the number of instances of your data that you want maintained. Specifying one instance means that there is only one copy of the logical volume (the default). In this case, there is a direct mapping of one LPAR to one PP. Each instance, including the first, is termed a copy. Where PPs are located (that is, how near each other physically) is determined by options that you specify when you create the logical volume.

File system

The logical volume defines the allocation of disk space down to the PP level. Finer levels of data management are accomplished by higher-level software components such as the AIX Virtual Memory Manager (VMM) 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, run the `crfs` command.

7.1.3 Principles to optimize LVM disks

Combining the concepts of how AIX physically structures HDDs with how LVM can impose logical structures onto disks, you can use several principles to optimize LVM disks, which are described in the following sections.

Making LVs contiguous

When file systems are located across a disk, the HDD head takes longer to find data. If the LV is in one continuous area, the seek time is minimized and the files can be found more quickly.

Placing 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 benefit from the higher rotational velocity at the outer edge.

Placing LVs with high activity toward the center

If you have a file system that has many reads and writes and needs quick responsiveness, because of averages the HDD head most likely is near the center at any point. By placing those file systems in this area, there is a higher likelihood that the head will be in that general vicinity. This configuration reduces seek time and maintains good I/O.

Placing 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, that is, near the spindle at the inner edge. For example, logical volumes that are seldom used fit here.

Using 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 because the disk head must go to several areas of the platters instead of just one.

Keeping paging space sizes uniform

Suppose that an AIX server has two 80 GB disks, and one has a 1 GB paging space LV and the other one has a 4 GB paging space LV. One of these two LVs experiences more wear. Depending on how the paging space was added to the server or shrunk in size, one of the LVs might become full and adversely affect the system. Keeping sizes the same makes the `lspv` command output cleaner and more accurate.

Keeping some free space on the disks

This is not just having one or two large gaps in the inner or outer middle regions. You should have space among the LVs in case anything needs to be grown, and have a little free space in the file systems themselves.

Tracking newly created file systems on a mirrored environment

One common mistake administrators make is to create a file system on a mirrored VG and then forget to create a copy onto the other disk. A mirrored VG does not mean that any later added LVs that support the file systems also are mirrored. Always make sure that new LVs have a copy on a separate PV.

Distributing an 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, 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 HDDs

The most common form of storage in AIX, internal HDDs are typically used for root VG disks and servers with smaller foot prints. When using internal HDDs, have at least two disks per VG, and mirror the rootvg HDDs.

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, you should manage precisely the configuration of the environment because there might be some single points of failure. The storage should be optimized with a proper RAID configuration, such as a RAID 5 setup with hot spare disks.

Large SAN storage

In larger SAN storage environments where multiple servers access many storage devices, such as IBM System Storage™ DS devices, through SAN switches, there are 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 (FC) cards to communicate with different fabrics and improve throughput.

7.1.5 LVM configuration

With LVM, you can mirror VGs, 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 VG.

The following instructions show you how to mirror a root VG by using the System Management Interface Tool (SMIT):

1. Select a VG in the Volumes container, and then choose Mirror from the Selected menu. Experienced administrators can use the `mirrorvg` command.
2. With root authority, add a disk to the VG by using the SMIT fast path that is shown in Example 7-1.

Example 7-1 Extending the volume group by adding a disk for mirroring

```
# smitty extendvg
```

```
Add a Physical Volume to a Volume Group
```

```
Type or select values in entry fields.
```

```
Press Enter AFTER making all desired changes.
```

```
FORCE the creation of volume group?      no          [Entry Fields]
+
* VOLUME GROUP name                       []
+
* PHYSICAL VOLUME names                   []
+
```

3. Mirror the VG on to the new disk by using the SMIT fast path that is shown in Example 7-2.

Example 7-2 Using the mirrorvg SMIT menu

```
# smit mirrorvg
```

```
Mirror a Volume Group
```

```
Type or select a value for the entry field.
```

```
Press Enter AFTER making all desired changes.
```

```
* VOLUME GROUP name                       []          [Entry Fields]
+
```

4. In the first panel, select a VG for mirroring. In the second panel, define the mirroring options or accept the 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 PPs, which are copies of the original PP, run the **syncvg** command. The command can be used with logical volumes, PVs, or VGs, by using the **Name** parameter, which represents the logical volume name, PV name, or VG name. The synchronization process can be time-consuming, depending on the hardware characteristics and the amount of data.

When you use the **-f** flag, a good physical copy is chosen and propagated to all other copies of the LPAR, 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, depending on the flag that is passed to the command. You can query the sync operation, which reports a verbose list of sync operation process identifiers (PIDs) with the sync rate of the operation.

A new flag that is known as **-T SyncRate** throttles 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 MBps, 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 VG. 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 VG.

Direct specific reads

There is a new LVM mirror policy to direct all reads to a specific mirror, which allows an LVM mirror to be used as fast copy for improved read performance. This policy off loads the primary storage resources, which improves the overall performance of the storage environment.

To use this policy, use the **-R** flag with the **mklv** or **chlv** command. The **-R** option sets the read preference to the copy of the logical volume. If the **-R** flag is specified and the preferred copy is available, the read operation occurs from the preferred copy. If the preferred copy is not available, the read operation follows the scheduling policy of the logical volume.

The **PreferredRead** variable can be set to a value 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 while creating of the LV or afterward.

Example 7-3 Setting the preferred read copy by using mklv or changing it by using 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 OS or a file system.

Bypassing the file system can yield better performance from the controlling application, especially from database applications. However, the amount of improvement depends on factors such as the size of a database or the application's driver, so you must provide the application with the character or block special device file for the new raw logical volume. The application links to this device file when it attempts opens, reads, writes, and so on.

Each logical volume has a logical volume control block (LVCB) in the first block. The size of LVCB is the block size of the PVs within the VG. Data begins in the second block of the PV. In a raw logical volume, the LVCB is not protected. If an application overwrites the LVCB, commands that normally update the LVCB 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 (CLI) to define a raw logical volume. The information in this how-to scenario was tested by using specific versions of AIX. The results that you obtain might vary depending on your version and level of AIX.

1. With root authority, find the free PPs on which you can create the raw logical volume by using the SMIT fast path that is shown in Example 7-4.

Example 7-4 Checking the free PPs

```
# smitty lspv
```

2. Select the disk, accept the default in the second dialog status, and click OK.
3. Multiply the value in the FREE PPs field by the value in the PP SIZE field to get the total number of megabytes that are 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 the SMIT menu.
4. Run the `mklv` command to create the raw logical volume.

Example 7-5 creates a raw logical volume that is named `lvdb2003` in the `db2vg` VG by using thirty-eight 4 MB PPs.

Example 7-5 Using the `mklv` command to create raw LVs

```
mklv -y lvdb2003 db2vg 38
```

Now, the raw logical volume is created. If you list the contents of your VG, 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 that a file system is mounted for your raw logical volume.

To learn how to open `/dev/rawLVname` and use this raw space, see your application's instructions.

Removing a disk from a live running system

The following sections describe how to remove a disk by using the hot-removability feature, which lets you remove the disk without turning off the system. Hot removability is useful when you want to do the following tasks:

- ▶ Remove a disk that contains data in a separate non-rootvg VG for security or maintenance purposes.
- ▶ Permanently remove a disk from a VG.
- ▶ Correct a disk failure.

Removing a disk without data

To remove a disk that contains either no data or no data that you want to keep, complete the following steps:

1. Unmount the file systems by using either of the following methods:
 - Run the **umount** command, as shown in

Example 7-6 Unmounting a file system by using umount

```
# umount /filesystem
```

- Use SMIT menu, as shown in Example 7-7.

Example 7-7 Unmounting a file system by 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)
  -OR-
Unmount all REMOTELY mounted file systems?  no
+
  -OR-
NAME of file system to unmount              []
+
```

2. Deactivate the volume group by running the **varyoffvg** command, as shown in Example 7-8.

Example 7-8 Deactivating the volume group by using the varyoffvg command

```
# varyoffvg VGNAME
```

3. Delete the volume group information that is referenced as metadata by running the **exportvg** command, as shown in Example 7-9.

Example 7-9 Deleting the volume group by running the exportvg command

```
# exportvg VGNAME
```

4. Delete the disk definition from the AIX server. Deleting the disk definition removes all information that is related to this disk from the AIX Object Data Management (ODM), as shown in Example 7-10.

Example 7-10 Deleting the disk by running the rmdev command

```
# rmdev -Rdl hdiskX
```

5. Physically remove the disk, whether it is an internal disk or mapped as a LUN from a SAN mapping.

Removing a disk with data

The disk that you are removing must be in a separate non-rootvg VG. Use this procedure when you want to move a disk to another system. It removes a disk that contains data without turning off the system.

1. List the volume group that is associated with the disk that you want to remove by running the **lspv** command, as shown in Example 7-11.

Example 7-11 Listing the hdisk2 information by using lspv hdisk2

```
# lspv hdisk2
PHYSICAL VOLUME:   hdisk2                VOLUME GROUP:   imagesvg
PV IDENTIFIER:    00083772caa7896e VG IDENTIFIER
0004234500004c00000000e9b5cac262
PV STATE:         active
STALE PARTITIONS: 0                ALLOCATABLE:    yes
PP SIZE:          16 megabyte(s)          LOGICAL VOLUMES: 5
TOTAL PPs:        542 (8672 megabytes)    VG DESCRIPTORS: 2
FREE PPs:         19 (304 megabytes)      HOT SPARE:      no
USED PPs:         523 (8368 megabytes)
FREE DISTRIBUTION: 00..00..00..00..19
USED DISTRIBUTION: 109..108..108..108..90
```

The name of the VG is listed in the VOLUME GROUP field. In this example, the VG is imagesvg.

2. Make sure that the disk is in a separate non-rootvg VG by running the **lsvg** command, as shown in Example 7-12.

Example 7-12 Listing the volume group information

```
# lsvg imagesvg
VOLUME GROUP:   imagesvg                VG IDENTIFIER:
0004234500004c00000000e9b5cac262
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                       STALE PPs:      0
ACTIVE PVs:    1                       AUTO ON:        yes
MAX PPs per PV: 1016                   MAX PVs:        32
LTG size:       128 kilobyte(s)        AUTO SYNC:      no
HOT SPARE:     no
```

In the example, the TOTAL PVs field indicates that there is only one PV that is associated with imagesvg. Because all data in this VG is contained on hdisk2, hdisk2 can be removed by using this procedure.

3. Unmount the file system, deactivate the volume group, and export it, as shown in Example 7-13.

Example 7-13 Exporting the volume group

```
# umount /filesystem
# varyoffvg imagesvg
# exportvg imagesvg
```

These commands unmount the file system, and then **varyoff** the VG so that you can export it. This process removes the file system's information from ODM.

4. Remove the disk from the OS by running the `rmdev` command, as shown in Example 7-14.

Example 7-14 Removing the disk by using the `rmdev` command

```
# rmdev -Rdl hdisk2
```

Resizing a RAID volume group

On systems that use a RAID, `chvg` and `chpv` command options can add a disk to the RAID group and grow the size of the PV that LVM uses without interruptions to the use or availability of the system.

The size of all disks in a VG is automatically examined when the VG 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 VG is activated in classic or in enhanced concurrent mode.
 - ▶ The rootvg VG cannot be resized by using the following procedure.
 - ▶ A VG with an active paging space cannot be resized by using the following procedure.
1. Check for disk growth and resize if needed by running the `chvg -g` command, as shown in Example 7-15.

Example 7-15 Checking disk growth

```
# chvg -g VGname
```

VGname is the name of your VG. This command examines all disks in the VG. If any have grown in size, the command attempts to add PPs to the PV. If necessary, it determines the appropriate 1016 limit multiplier and converts the VG to a big VG.

2. Turn off the LVM bad block relocation policy for the VG by running the `chvg -b` command, as shown in Example 7-16.

Example 7-16 Turning off the bad block relocation policy

```
# chvg -b ny VGname
```

VGname is the name of your VG.

When to create separate volume groups

There are several reasons why you might want to organize PVs into VGs separate from rootvg:

- ▶ For safer and easier maintenance.
OS updates, reinstallations, and crash recoveries are safer because you can separate user file systems from the OS so that user files are not jeopardized during these operations.
Maintenance is easier because you can update or reinstall the OS without having to restore user data. For example, before updating, you can remove a user-defined VG from the system by unmounting its file systems. Deactivate it by using the **varyoffvg** command, and then export the group by using the **exportvg** command. After updating the system software, you can reintroduce the user-defined VG by using the **importvg** command, and then remount its file systems.
- ▶ For different physical-partition sizes.
All PVs within the same VG must have the same PP size. To have PVs with different PP sizes, place each size in a separate VG.
- ▶ Different quorum characteristics are required.
If you have a file system for which you want to create a nonquorum VG, maintain a separate VG for that data; all the other file systems should remain in VGs operating under a quorum.
- ▶ To switch physical volumes between systems.
If you create a separate VG for each system on an adapter that is accessible from more than one system, you can switch the PVs between the systems that are accessible on that adapter without interrupting the normal operation of either.

7.2 AIX Enhanced Journaled File System

The 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 is 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, which is called a *snapshot*.

7.2.1 Enhanced Journaled File System functions

Enhanced Journaled File System (JFS2) is a file system that stores much larger files than the existing 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, either create a logical volume or let AIX create the logical volume and the file system on it. To access the file system, mount it on a directory mount point. When multiple file systems are mounted, a directory structure is created that presents the image of a single file system.

JFS2 is a hierarchical structure with a single root. This structure includes the base file systems and any file systems that you create. You can access both local and remote file systems by using the `mount` command.

JFS can be increased online but cannot be shrunk, but JFS2 can copy and back up the old data to a resized file system.

There are several improvements and differences with JFS2. For more information about these items, see [IBM Knowledge Center](#).

7.2.2 JFS2 features

JFS2 has several features that make it more reliable and secure.

JFS2 performance

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 write operations are performed after the logging of changes to the metadata are complete.

Support for data sets is integrated into JFS2 as part of the AIX OS. 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 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 a mounted file system, that is, a mounted file system instance might contain multiple data sets. Data set support is enabled in JFS2 by running the `mkfs -o dm=on` command. By default, data set support is not enabled.

JFS2 cannot disable metadata logging. However, the implementation of journaling on JFS2 makes it more suitable to handle metadata-intensive applications. Thus, the performance penalty is not as high under JFS2 as it is under JFS.

The main performance advantage of using JFS2 over JFS is scaling, JFS2 can store larger files than the existing JFS. The maximum size of a file under JFS is 64 GB. Under JFS2, AIX supports files up to 16 TB, although the file system architecture is set up to eventually handle file sizes of up to 4 PB.

JFS2 security

Encrypted File System (EFS) is supported only on JFS2. EFS can 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 keystore. 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.

For more information about EFS, see 7.4, “Encrypted File System” on page 182.

7.3 Role-based access control

Traditionally, only the *root* user 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, AIX introduced powerful security features like role-based authentication control (RBAC), Multilevel Security (MLS), Trusted Execution (TE), EFS, and others to complement 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. Although a traditional UNIX system administration cannot achieve these goals, you can use RBAC to do so.

RBAC creates 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 that are typically reserved for the root user can be assigned to regular system users.

RBAC accomplishes this task by defining job functions (*roles*) within an organization and assigning those roles to specific users. RBAC is essentially a framework that you use to do system administration by using 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, and another role might be to enable the creation of user accounts.

RBAC administration has the following advantages compared to traditional UNIX administration:

- ▶ System administration can be performed by multiple users without sharing account access.
- ▶ Security isolation is set by granular administration because each administrator does not need to be granted more power than is required.
- ▶ Allows for enforcing a least-privilege security model. Users and applications are granted necessary privileges only when required, which reduces the impact a system attacker can have.
- ▶ Allows for implementing and enforcing company-wide security policies consistently regarding system management and access control.
- ▶ 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 three core concepts, which are described in the following sections.

Authorizations

An *authorization* is a text string that is 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 that is governed by an authorization is run, access is granted only if the launching user has the required authorization. An authorization can be thought of as a key that can 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: The `aix.ras.error.errrpt` and `aix.ras.trace.trcrptch0` access authorizations were introduced in AIX 7.2 to manage system error reports, and to administer system trace reports.

Roles

Roles allow a set of management functions in the system to be grouped. 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 subrole. A subrole is another role from which a role inherits the authorizations.

A role by itself does not grant the user any additional privileges, but 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 is 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. After a role is assigned to a user, the user can use the authorizations that are 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 be assigned only to members of the administrative staff. Just as the powers of root have historically been given only to trusted users, roles should be assigned only to trusted users. Grant roles only to users with legitimate needs and only during 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. Privileges are the restriction mechanism that is used in the kernel to determine whether 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 restarting 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 in the AIX kernel that provide an extra layer of security. Table 7-1 shows the tables.

Table 7-1 RBAC Kernel Security Tables

| Kernel Security Table Name | Definition |
|----------------------------|-------------------------|
| auth | For authorizations |
| role | For roles |
| cmd | For privileged commands |
| dev | For privileged devices |
| dom | For domains |
| domobj | For domain objects |

As for the RBAC definition files, they are listed under the `/etc/security` path. Table 7-2 shows RBAC definition files.

Table 7-2 RBAC definition files

| RBAC definition file | File function |
|--------------------------------------|---|
| <code>/etc/security/privcmds</code> | To run AIX commands |
| <code>/etc/security/privfiles</code> | To edit AIX files with a special edit command |
| <code>/etc/security/privdevs</code> | To read/write devices |
| <code>/etc/security/domains</code> | To domain names |
| <code>/etc/security/domobjs</code> | To objects that are protected by domains |

RBAC can be customized to serve administration needs for who can do exactly what commands with certain options.

RBAC has default roles to directly assign to users for certain jobs like backup, shutdown, and file systems management.

7.3.3 Customizing an RBAC role for certain tasks

This section describes a real example of a customized role by RBAC.

Customizing a role for a command that is associated to an authorization

The section describes the customization of a role for any specific command where this command has an associated authorization included in the `/etc/security/privcmds` file.

Complete the following steps:

1. For example, if you want to create a custom role for the `mklv` command, check its full path and whether it exists in the privileged command, as shown in Example 7-17.

Example 7-17 Checking the `mklv` command full path and whether 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. Check whether there are any roles that are associated with the command authorization as `accessauthx`, as shown in Example 7-18.

Example 7-18 Checking whether any role is associated to the access authorization

```
# lsrole ALL | grep aix.lvm.manage.create
```

The output shows that no current role exists with that specific authorization.

3. Create the custom role that will have the authorization by running the `mkrole` command, as shown in Example 7-19.

Example 7-19 Creating a role by running the `mkrole` command

```
# mkrole dfltmsg="LVs Creation" authorizations="aix.lvm.manage.create"
makelvrole
```

4. After the role is created, you must update the kernel security table by running the `setkst` command, as shown in Example 7-20.

Example 7-20 Running the `setkst` command to update the 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 `make_lvrrole` to a user, run the `chuser` command to assign the role either in the roles database or in the `default_roles` database.

If you assign the role to the roles database, the user must switch to it by running the `swrole` command. If you assigned the role to `default_roles`, it is applied automatically when the user logs in. Example 7-21 shows an example of assigning the new role to user bob.

Example 7-21 Running the chuser command to add the new role

```
# chuser roles=make_lvrrole default_roles=make_lvrrole bob
```

6. If user bob logs in, they see that their `make_lvrrole` is automatically active, as shown in Example 7-22.

Example 7-22 Listing roles that are 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 that is not associated to any authorization

This section describes customizing a new role with any command privileges in AIX, including any other non-AIX command, like HTTP and `apachectl` application commands.

Complete the following steps:

1. Check whether the `enhanced_RBAC` kernel attribute is enabled in the system, as shown in Example 7-23.

Example 7-23 Checking whether enhanced_RBAC is enabled

```
# lsattr -El sys0 -a enhanced_RBAC
enhanced_RBAC true Enhanced RBAC Mode True
```

Example 7-23 shows that `enhanced_RBAC` is enabled (`true`). If `enhanced_RBAC` is not enabled, enable it and restart the AIX server, as shown in Example 7-24.

Example 7-24 Enabling enhanced_RBAC and restarting

```
# chdev -l sys0 -a enhanced_RBAC=true
sys0 changed
# shutdown -Fr
```

2. Create a user-defined authorization for the new role by running the `mkauth` command, as shown in Example 7-25.

Example 7-25 Running the mkauth command to create an authorization

```
# mkauth df1tmsg='My Custom Auth App' my_app_auth
```

This authorization is added to the `/etc/security/authorizations` file. To view the authorization, run the `lsauth` command, as shown in Example 7-26.

Example 7-26 Viewing the authorization

```
# lsauth my_app_auth
my_app_auth id=10020 dfltmsg=My Custom Auth App
```

- Trace the privileges for the command or binary file that you want for the run, and grab the security attributes that you want to associate with the authorization.

If a command like `/usr/local/bin/myapp` is traced, it should be written with its full path and arguments or you do not receive the required privileges. If the command is provided with no arguments but needs them, the trace does not return an output.

Example 7-27 shows how to trace a certain command.

Example 7-27 Running tracepriv for a certain command

```
# 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 Example 7-27 shows five security attributes that are used while running the `/usr/local/bin/myapp` command. Make a note of them.

- Set the security attributes for the binary object.

You must bind the security attributes from Example 7-27 with the authorization. Run the `setsecattr` command, as shown in Example 7-28.

Example 7-28 Running the setsecattr command to bind security attributes to the authorization

```
# 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
```

The authorization, and the security attributes that are attached to it, were added to the `/etc/security/privcmds` file and can be viewed by running the `lssecattr` command, as shown in Example 7-29.

Example 7-29 Viewing the complete authorization that is associated with the command

```
# lssecattr -Fc /usr/local/bin/myapp
```

- Create a role and associate it with the authorization that you created by running the `mkrole` command, as shown in Example 7-30.

Example 7-30 Running the mkrole command

```
# mkrole authorizations=my_app_auth dfltmsg="My_Role" exec_app
```

- Add the role to the user roles or `default_roles` by running the `chuser` command, as shown in Example 7-31.

Example 7-31 Adding the new role to the user roles

```
# chuser roles=exec_app testuser
```

7. Update the Kernel Security Table by running the `setkst` command, as shown in Example 7-32.

Example 7-32 Updating the kernel security table by running the `setkst` command

```
# 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 (domains), and the specific access rules determine access to resources by the users.

Definition and access rules

Here are the key terms for Domain RBAC that 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 superset 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 that are known as 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 describes domain RBAC implementation by using a scenario where a non-root user is granted a role that can mount file systems without being able to mount *specific* file systems.

Complete the following steps:

1. Make sure that **enhanced_RBAC** is enabled, as shown in Example 7-33.

Example 7-33 Checking the enhanced_RBAC status

```
# lsattr -El sys0 -a enhanced_RBAC
enhanced_RBAC true Enhanced RBAC Mode True
```

2. Create a user for the job of mounting file systems, as shown in 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 the security authorizations of the specific **mount** command, as shown in Example 7-35.

Example 7-35 Checking the authorizations that are associated with the mount command

```
# lssecattr -c -a accessauths /usr/sbin/mount
/usr/sbin/mount accessauths=aix.fs.manage.mount
```

Example 7-36 shows that the command has the `aix.fs.manage.mount` authorization.

4. Check the security authorizations of the specific **umount** command, as shown in Example 7-36.

Example 7-36 Checking the authorizations that are associated with the umount command

```
# lssecattr -c -a accessauths /usr/sbin/umount
/usr/sbin/umount accessauths=aix.fs.manage.umount
```

Example 7-36 shows that the command has the `aix.fs.manage.umount` authorization.

5. Create a role and assign both authorizations to it, and then list the role to see its authorizations, as shown in Example 7-37.

Example 7-37 Creating a role with both authorizations and listing the role

```
# mkrole authorizations=aix.fs.manage.mount,aix.fs.manage.umount
fs_manage

# lsrole fs_manage
fs_manage
authorizations=aix.fs.manage.mount,aix.fs.manage.umount
rolelist= groups= visibility=1 screens=* dfltmmsg= msgcat=
auth_mode=INVOKER id=18
```

6. Assign the newly created role to the designated user, as shown in Example 7-38.

Example 7-38 Assigning the newly created role to the designated user

```
# chuser roles=fs_manage test
# lsuser -a roles test
test roles=fs_manage
```

7. Update the Kernel Security Table, as shown in Example 7-39.

Example 7-39 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.
```

8. Test the role of the user, as shown in Example 7-40.

Example 7-40 Testing the role of the user

```
# su - test
$ swrole ALL
test's Password:
$ mount /testfs1
$ mount /testfs2
$ df -g | grep testfs
/dev/fs1v11 1.00 1.00 1% 4 1% /testfs1
/dev/fs1v12 1.00 1.00 1% 4 1% /testfs2
$ umount /testfs1
$ umount /testfs2
```

The user can mount or unmount any file system. Now, you must use domain RBAC objects to restrict the user from being able to mount specific file systems.

9. Create two RBAC domains and list them, as shown in Example 7-41.

Example 7-41 Creating RBAC domains

```
# mldom dom1
# mldom dom2
# lsdom ALL
dom1 id=1
dom1 id=1
```

10. Check the file systems to which you apply the domains, as shown in Example 7-42.

Example 7-42 Checking the file system

```
# lsfs | grep testfs
/dev/fs1v11 -- /testfs1 jfs2 2097152 -- yes no
/dev/fs1v12 -- /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.

Example 7-43 Assigning a domain to a file system

```
# setsecattr -o domains=dom1 objtype=device secflags=FSF_DOM_ANY
/dev/fs1v11

# setsecattr -o domains=dom2 objtype=device secflags=FSF_DOM_ANY
/dev/fs1v12
```

```
# 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* the dom1 domain to the test user, as shown in Example 7-44.

Example 7-44 Assigning only the dom1 domain to the test 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 that is associated to the dom1 domain is mounted, as shown in Example 7-46.

Example 7-46 Attempting to mount the file systems

```
# su - test
$ swrole ALL
test's Password:

$ mount /testfs1
$ mount /testfs2
mount: 0506-324 Cannot mount /dev/fslv12 on /testfs2: Operation
not permitted.
```

Because the user can mount only a file system with the security attribute of the domain that is assigned to them, they cannot mount the file systems in a different domain.

7.4 Encrypted File System

EFS is a JFS2 file system-level encryption system that uses individual keystores. It enables file encryption that protects confidential data from attackers with physical access to the computer.

User authentication and access control lists (ACLs) can protect files from unauthorized access while the OS is running, but it is easy to circumvent the ACLs 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 keystore. On successful login, the user's keys are loaded into the kernel and associated with the process credentials. When the process must open an EFS-protected file, the system tests the credentials. If the system finds a key matching the file protection, the process can decrypt the file key and file content. The cryptographic information is kept in the extended attributes for each file.

EFS uses extended attributes Version 2, and each file is encrypted before being written on to the disk. The files are decrypted when they are read from the disk into memory so that the file data that is 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.

To create and use the EFS-enabled file system on a system, the following prerequisites must be met:

- ▶ Install the CryptoLite for C (CLiC) cryptographic library.
- ▶ Enable the RBAC.
- ▶ Enable the system to use the EFS file system.

7.4.1 EFS commands

This section discusses EFS essential commands.

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.

The 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 when this command runs.

efskeymgr

The **efskeymgr** command is dedicated to all key management operations that are 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 that are 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 be created only on the EFS-enabled JFS2 file systems. Inheritance is set on the file system or the directory where the file is being created by using this command. When inheritance is set on a directory, all new files that are created in this directory are encrypted by default. The cipher that is 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 that are created in this subdirectory are not 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: sales, marketing, and finance. These three departments share an AIX machine to store their confidential content. If EFS is not enabled, the potential of having the data exposed among the three departments is high.

Enabling EFS

To enable EFS, complete the following steps:

1. To enable the EFS, run the `efsenable` command, as shown in Example 7-47.

Example 7-47 Enabling EFS by running `efsenable`

```
# efsenable -a
Enter password to protect your initial keystore:
Enter the same password again:
```

2. Check the EFS directories that were created under `/var/efs` to facilitate the EFS operation, as shown in Example 7-48.

Example 7-48 Checking the EFS directories

```
# cd /var/efs
# ls
efs_admin efsenabled groups users
```

Creating the EFS

All the EFS capabilities should now be enabled. Now, create a separate file system for all the three departments.

Creating an EFS is similar to creating a normal file system. The only difference is that you must enable the `efs = yes` attribute. Example 7-49 shows how to create EFSs.

Example 7-49 Creating EFSs

```
# 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 now successfully created three separate file systems for these three departments by enabling the `efs` option.

Creating the users and checking the keystores

To create the users, run the `mkuser` command and check the keystore, as shown in Example 7-50.

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
```

Creating EFS directories and setting the EFS properties

To create EFS directories, the EFS file system must be mounted. Example 7-51 shows how to create EFS directories and set the inheritance.

Example 7-51 Creating the EFS directory and setting the inheritance

```
# 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 that you can use with `efsmgr` that can 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 that the encryption is on and you try to create files in the directory `yearlyreport`, you should receive an error, as described in Example 7-52.

Example 7-52 Checking the EFS enablement

```
# cd yearlyreport
# ls
# touch apr_report
touch: 0652-046 Cannot create apr_report.
```

Loading the EFS keystore in to the shell

To use the previous touch command activity, you must have the EFS keystore loaded in to the shell. Example 7-53 shows how to load the EFS keystore.

Example 7-53 Loading the EFS keystore into the shell

```
# efskeymgr -o ksh
financeman's EFS password:
# touch enc_file
```

Now that you loaded the keystore, any information that is added to this file is encrypted at the file system level.

Checking the encryption of files and the ability to encrypt a single file

You can always check the encryption flag of the files that are created in EFSs and to encrypt individual files.

Checking the encryption

To check the encryption of files, run the **ls** command, as described in Example 7-54.

Example 7-54 Listing the encrypted files

```
# ls -U enc_file
-rw-r--r--e 1 financeman system 0 Oct 21 09:14 enc_file
```

The **-U** option for the **ls** command outputs an extra bit, which is the 11th character. It can be one of the following items:

- E** Indicates that a file has extended attributes information. The extended attributes of a file are displayed by running the **getea** command.
- Indicates that a file does not have extended attributes information.
- e** Indicates that a file is encrypted.

You can also list the encrypted file attributes by running the **efsmgr** command, as described in Example 7-55.

Example 7-55 Listing the 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
```

Encrypting an individual file

If you must encrypt an individual file, run the **efsmgr** command, as shown in Example 7-56.

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 Integrating an EFS keystore with OpenSSH key authentication

This section describes how to configure EFS keystore access while using OpenSSH public key authentication. It explains the procedure for automatic opening of the EFS keystore when Secure Shell (SSH) public key authentication is used to log on to a remote system.

This section sets up public key authentication in OpenSSH.

Enabling public authentication on both the client and server

Create a user on the client side and generate keys for this user. Then, generate public-private key pairs by using the `ssh-keygen` command. Enable public key authentication on both the client and servers by using the `/etc/ssh/ssh_config` file and set `PubKeyAuthentication` to yes, as shown in Example 7-57.

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
```

Configuring the OpenSSH client and server to use EFS login

To enable EFS login for both client and server, set `AllowPKCS12keystoreAutoOpen` to yes in `/etc/ssh/ssh_config`, as shown in Example 7-58.

Example 7-58 Enabling EFS login for the client and server

```
# hostname
client.ibm.com
# grep AllowPKCS12keystoreAuto Open /etc/ssh/ssh_config
AllowPKCS12keystoreAutoOpen yes
# hostname
server.ibm.com
# grep AllowPKCS12keystoreAuto Open /etc/ssh/ssh_config
AllowPKCS12keystoreAutoOpen yes
```

Restarting the ssh daemon in both the client and server

Restart `sshd` in both the client and server so that it can take effect, as shown in Example 7-59.

Example 7-59 Restarting sshd on both the 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

```

Creating a user in the client side and generating the key through the user

Generate the key by using a newly created user on the client machine. Example 7-60 shows how to create the user and generate the key.

Example 7-60 Creating a client user and generating 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
:07:5d:0f:20:95:d4:9c:15:8d:77:bd:93:ea:3c:ac:99 bob@client
The key's randomart image is
+--[ RSA 2048 ]--+
|   . . . . .0+=000+. |
| 0.....+0. = |
| . . . . .+. |
| ..... + |
| S . . . . . |
| . . . . . |
| 0= . . . . . |
+-----+

```

The **ssh-keygen** command prompts for a passphrase. This passphrase is used to encrypt the private-key file on the client side. The **ssh-keygen** command accepts an empty passphrase, in which case the private-key file is not encrypted.

Copy the public keys on to the server to the file `~/ .ssh/authorized_keys`, as shown in Example 7-61.

Example 7-61 Copying public keys on to the server into authorized_keys

```

# hostname
server.ibm.com
# cat id_rsa.pub > /home/laxman/.ssh/authorized_keys
# cat /home/laxman/.ssh/authorized_keys

```

```
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEAqYK16NpoJ1Nq1/ccb1Ftu2fGk0Qd2T4H74d1c6Qgs
kRHG07e0yTqt58yFJ05h7Zr8g1eQLo09H6CVA7hi7EKwfg7fPpGWUGdpL6
Aq8sgwRkhJ0YptczeRu
jSCi7hyvkT2DhLx7svZ0x47K1PfHFTNPRUjKZ1yPscTs2XWqAdDvPQPv0T14agRFqB81d/gXm2vFS
VUP+PJDoVVub/DMY928FRBd6fYEffgZybyMOR1
4kbuQoJFrnoGZACg4maiPi5fKLiXY0W1+
```

Afterward, a client user's public key can be copied multiple times into the `~/.ssh/authorized_keys` file on the server user account.

EFS must be enabled on the server side by running the **efsenable** command, which creates an admin keystore. The keystore is created whenever a new user is created, when a password is assigned to the user, or when the user logs in.

The path where the user keystore is created on the system is `/var/efs/users/<userlogin>/keystore`, and the format of the user keystore is Public-Key Cryptography Standards #12 (PKCS#12), which contains public and private objects.

Private objects are protected by user access key. This user access key is a hash of a user-defined password, which is either a login password or another password that is specific to EFS.

Inserting the public key into the server keystore

Create the public key cookie and insert it into the keystore on the server side. Run the **efskeymgr** command to insert the cookie. A public key cookie is **passwd** that is encrypted with the user's public key.

Example 7-62 shows how to create a keystore for a user and insert the public key cookie.

Example 7-62 Creating a keystore and insert the public key cookie

```
# 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
```

After the configuration setting is complete, run **ssh** to log in to the remote machine by using public key authentication.

Verifying the EFS login authentication

The OpenSSH client user bob is set for public key authentication to user laxman on the OpenSSH server with EFS login. Verify this configuration by using an ssh login from the client, as shown in Example 7-63.

Example 7-63 Verifying the 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
Type ..... Private key
Algorithm ..... RSA_1024
Validity ..... Key is valid
Fingerprint ..... a1a07c79:e0d57e83:8f148a2c:ac778fab:f813cf11

$hostname
server.ibm.com
```

You can use this setup with different applications, including database engines for which OpenSSH public key authentication can be used. A good example is IBM Db2®. If you are running a Db2 system on the AIX OS, you may set up an encrypted database by using AIX EFS. For more information about Db2 encryption by using EFS, see [IBM Knowledge Center](#).

7.5 AIX Security Expert and IBM PowerSC integration

AIX Security Expert (AIXpert) is a security hardening AIX tool that is part of the `bos.aixpert` file set. It uses a policy file to automatically lock down AIX. It has various settings, like low, medium, high, and SOX/Cobit.

The policy has over a hundred rules and tools, such as password aging and lockout on failed logins, SUID bit programs, disabling services, and blocking port scanning.

AIXpert can implement the appropriate level of security without requiring a great amount of knowledge and then needing to implement individually each security element. It also can be used to take a security configuration snapshot. You can use this snapshot to set up the same security configuration on other systems. This process saves time and ensures that all systems have the correct 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 that incorporates many functions into one system.

Before AIXpert, you needed to remember many different commands. AIXpert incorporates over 300 security configuration settings while still providing control over each element. 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

This level should be used in environments where security and controls are of paramount importance. Applications such as telnet; rlogin; and FTP, and other systems that transmit non-encrypted passwords do not work, so be careful when using this level. Understand that most ports are blocked in this scenario. Systems that can be connected directly to the internet that have sensitive data are good examples of a system that would use this level.

▶ Medium level

This level is appropriate for systems that are behind a firewall where users use services such as telnet but still want protection. This setting also provides port-scanning and password-setting protections.

▶ Low level

This level is used when the system is in an isolated and secured type of LAN. It also is used where system administrators must be careful not to interrupt any services to the environment.

▶ Advanced level

This level is for customization. With this level, you can use different rules from different levels. Some of the rules might be mutually exclusive. This level does not in itself provide a higher level of security.

More than any specific feature, it is the consolidation of security that really adds value to AIXpert. For example, AIXpert incorporates the File Permissions Manager in the form of the `fpm` command to also help manage set user ID (SUID) programs.

Another important feature is the ability to take snapshots of your system and reproduce these settings throughout the enterprise so that you can clone your security features across the organization. This feature includes the ability to undo settings, and it also checks the overall security health of the systems to report on settings that might have changed.

The ease with which you can undo security levels cannot be overstated. It is as simple as running the following command:

```
aixpert - u undo.xml
```

To set a specific AIXpert security level, run the AIXpert command that is shown in Example 7-64.

Example 7-64 Setting a specific AIXpert security level

```
# aixpert l <LEVEL>
```

The <LEVEL> parameter can be any of the following options that are shown in Table 7-3.

Table 7-3 Listing the AIXpert security levels

| AIXpert security level | Definition |
|------------------------|---|
| high | Specifies high-level security options. |
| medium | Specifies medium-level security options. |
| low | Specifies low-level security options. |
| default | Specifies AIX standard-level security options. |
| sox-cobit | Specifies SOX-COBIT best practices levels security options. |

7.5.2 Using AIXpert to generate a compliance report by running the IBM PowerSC pscxpert command

Because AIXpert can be implemented and used to generate compliance reports for the security level that you use, you can get a compliance report that is generated by the **pscxpert** command, which is a part of the IBM PowerSC file set.

In the following example, we apply low-level security by using AIXpert:

1. To implement low-level AIXpert security, run the **aixpert** command, as shown in Example 7-65.

Example 7-65 Implementing AIXpert low-level security in AIX

```
# aixpert -l low
```

The command takes time to process several rules, and eventually you see an output that shows the failing rules. If you need verbose output, use the **-p** option, as shown in Example 7-66.

Example 7-66 Applying low-level security with verbose mode

```
# 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/applieaixpert.xml
```

Example 7-66 shows one only failed rule, which means that one rule does not comply with the low-level security setting of AIXpert:

```
lls_minlen: Processing lls_minlen_97D38C75 : failed
```

The low-level security setting uses the instructions in an XML file to apply the failed rule. Example 7-66 on page 192 lists the XML input file at the end of the output:

```
/etc/security/aixpert/core/appliedaixpert.xml
```

2. There must be a discrepancy between the `minlen` rule from the XML file and what is implemented in the AIX system. Example 7-67 shows the `minlen` attribute in both the XML input file and in the `/etc/security/user` file.

Example 7-67 Checking the rule in the XML file and in /etc/security/user

```
# 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 that is shown in Example 7-67, we decide to comply with `minlen=8` and change that setting in `/etc/security/user`.

Then, we check the security settings against the previously applied set of rules by running the `aixpert` command with the `-p` option for verbosity, as shown in Example 7-68.

Example 7-68 Checking the security settings against the previously applied set of rules

```
# aixpert cp| grep lls_minlen
Processing lls_minlen_97D38C75 :done.
```

4. Because we are sure that we are now in compliance with the security rule level we set earlier, we can generate a compliance report by running `pscxpert`, which is a part of the `powerscStd.ice` file set.

The commands generate `.txt` and `.csv` files. Example 7-69 shows how to generate the report.

Example 7-69 Generating a compliance report by running pscxpert

```
# pscxpert -c -r

Processedrules=44 Passedrules=44 Failedrules=0 Level=LLS
Input file=/etc/security/aixpert/core/appliedaixpert.xml
```

5. Example 7-70 shows a sample of the `.txt` file that shows the compliance report.

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 as a `.csv` file that is named `/etc/security/aixpert/check_report.csv`, which can be loaded into a spreadsheet and filtered.

7.6 The AIX Auditing subsystem and Autonomic Health Advisor File System

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 Autonomic Health Advisor File System (AHAFS) monitoring.

7.6.1 The AIX Auditing subsystem

The AIX Auditing subsystem records security-related information and alerts system administrators about potential and actual violations of the system security policy. The information that is collected by auditing includes the name of the auditable event, the status (success or failure) of the event, and any additional event-specific information that is related to security auditing.

While dealing with the AIX Auditing subsystem, we must understand five items, which are described in the following sections.

Mode

BIN mode, STREAM mode, or both can be chosen while setting up the audit:

- ▶ BIN mode is useful when you plan to store records on a long-term basis.
- ▶ STREAM mode is useful when you want to do real-time information processing.

Events

Events are security-related activities that are defined by the system. Here are some 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.)

For more information about all available events, see [IBM Knowledge Center](#).

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 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, writes, and executes can be audited by using audit objects, and the file itself can be an executable command or a configuration file,

7.6.2 Implementing the AIX Auditing subsystem for exact events

You can define what classes you want to audit for a specific user, and you can audit one or more classes per user.

The following scenario describes an auditing setup for the PROC_Execute event, which is when a user runs any program.

Applying the mode

Edit the `/etc/security/audit/config` file and apply the needed mode.

Example 7-71 shows how to enable the stream mode.

Example 7-71 Changing the audit mode in the `/etc/security/audit/config` file

```
# vi /etc/security/audit/  
start:  
    binmode = off  
    streammode = on
```

Defining a class name

Define the class that includes the exact event that you need.

At the end of the `classes:` section in the `/etc/security/audit/config` file on the line before the `users:` section, enter a new line, as shown in Example 7-72.

Example 7-72 Defining a class that has the event name in the audit config file

```
classes:  
...  
    my_class = PROC_Execute
```

Adding the newly defined class under a specific user

Under the `users:` stanza in the `/etc/security/audit/config` file, you can either delete all the entries and add an entry for the class that you defined, or add it at the end of the `users:` section, as shown in Example 7-73.

Example 7-73 Adding the class to a specific user

```
users:  
...  
    my_user = my_class
```

Note: If you have any other users that you want to monitor for any PROC_Execute events, add an entry for that user name.

Adjusting the audit stream commands file to run auditstream

You can adjust the `/etc/security/audit/streamcmds` file to include the `/usr/sbin/auditstream` command, which is redirected to a certain output file, so that it runs 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.

Back up the `/etc/security/audit/streamcmds` file and edit its contents as shown in Example 7-74.

Example 7-74 Adding the `auditstream` command for redirection to an output file

```
# 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 &
```

`"/etc/security/audit/streamcmds"` 1 line, 72 characters

Setting up a monitoring action on a specific action

You can always monitor an exact object, such as a certain file or command, by monitoring its reads, writes, or executions.

Set up an audit on the execution of `date` command object, which has the full path of `/usr/bin/date`, by completing the following steps:

1. Define this object by name in `/etc/security/audit/object` and define what action you want to audit, such as `r` for read, `w` for write, and `x` for execution.

Example 7-75 shows 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"
```

You defined an event that is named `DATE_EXEC` that is reported every time the `date` command runs.

2. Format or define the format string of the `DATE_EXEC` event that is reported in the `/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"
```

The `%s` output is the event that will be formatted as a text string.

For more information about text formats, see [IBM Knowledge Center](#).

Recycling the audit

For the configuration to work and take effect, you must stop and start the audit subsystem, as shown in Example 7-77.

Example 7-77 Recycling the audit subsystem

```
# audit shutdown
# audit start
```

Wait until the issue that you set the audit for happens. Afterward, stop the audit and read the stream.out file, as shown in Example 7-78.

Example 7-78 Stopping the audit and monitoring the events

```
# audit shutdown
# cd /audit
# more stream.out
```

For more information, see [AIX Auditing Best Practices](#).

7.6.3 Autonomic Health Advisor File System

AHAFS provides an event monitoring framework in AIX. Users can monitor predefined system events and receive notifications about them.

Each event is represented as a file in a pseudo-file system that is named AHAFS. The standard file application programming interfaces (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 sections of code that can detect an event as it happens and notify the AIX event infrastructure of event occurrences.

Table 7-4 represents examples of available event producers.

Table 7-4 Available predefined event producers

| Predefined event producer | When it notifies the consumer |
|-----------------------------|---|
| utilFs | The utilization of a monitored file system crosses the user-specified threshold. |
| modFile | The contents of a monitored file are modified. |
| modFileAttr | Attributes like modebits, ACL, and ownership of a monitored file are modified. |
| modDir | A file or subdirectory is created, renamed, or deleted under a directory. |
| schedo | The value of a monitored scheduler tunable changed. |
| vmo | The value of a monitored VMMtunable changed. |
| waitTmCPU | The average wait time of all the threads waiting for CPU resources exceeds the user-specified threshold. |
| waitersFreePg | The number of waiters to get a free page frame in the last second exceeds the user-specified threshold. |
| waitTmPginOut | The average wait time of all the threads waiting for page in or page out operations to complete exceeds the user-specified threshold. |
| processMon or pidProcessMon | The monitored process exits. |

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 wants to monitor.

AHAFS setup sample

AHAFS has many useful sample programs and scripts under the directory `/usr/samples/ahafs/` that makes AHAFS 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.

Make a copy of the `aha-pl.inp` file, modify it by uncommenting lines in the file for things that you want to monitor or modify some key values if you want to use values other than the defaults, and then launch the `aha.pl` script by specifying your tailored `.inp` file. Optionally, specify the list of email IDs to which you want the monitoring reports to go.

By default, the AHAFS file set `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 the /aha file system

```
# mkdir /aha  
  
# mount -v ahafs /aha /aha
```

Mounting the AHAFS creates the following items.

Event produce list

The `/aha/evProds.list` file, which contains the list of predefined and user-defined event producers that are 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`, such as `mem/`, `cpu/`, and `fs/`.

Monitor factories

The subdirectories under the component subdirectories with the file type of `.monFactory`, each of which corresponds to an event producer for AHAFS.

Here is the procedure to do an example setup of AHAFS:

1. Copy the `aha-pl.inp` file to another directory and modify it to look like the monitoring line that is shown in Example 7-80.

Example 7-80 Editing the 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
```

Example 7-80 monitors the `/tmp` directory for any changes, sends an email alert after the second change, and then rearms the alert to start watching again in the next 5 minutes.

2. Mount the aha file system and start the monitoring Perl script, as shown in Example 7-81.

Example 7-81 Mounting the aha file system and start monitoring

```
# mkdir /aha
# mount -v ahafs /aha /aha
# /usr/samples/ahafs/bin/aha.pl -i /myfiles/inputfile -e bob
```

The bob user is the user who receives an email notification of the event.

After you complete the setup, this example AHAFS scenario does the following tasks:

1. Send a notification about the event match in the stdout output and in an email.
2. The notification matches any new files or subdirectories that are created, renamed, copied, or deleted under the directory /tmp.
3. The event match does not include changes in the files' content because those changes are a modDir event, not a modFile event.
4. The event match does not include any metadata modification (ownerships, permissions, and so on) because they are not modFileAttr events.

7.7 MultiBOS

With the **multibos** utility, you can as the root user create multiple instances of AIX on the same current root VG (rootvg). This process is different than **altinst_rootvg** cloning, which processes the clones onto different disks.

The two main differences are:

- ▶ The instances are inside the rootvg itself.
- ▶ Only two instances of the base operating system (BOS) are supported per rootvg.

The MultiBOS setup operation creates a standby BOS that boots from a distinct boot logical volume (BLV), which creates two bootable instances of a BOS on a rootvg.

You can boot from either instance of a BOS by specifying the appropriate BLV as an argument to the **bootlist** command or by using system firmware boot operations.

You can simultaneously maintain two bootable instances of a BOS. The instance of a BOS that is associated with the booted BLV is designated as the active BOS.

The instance of a BOS that is associated with the BLV that is not booted is designated as the standby BOS. Only two instances of BOS are supported per rootvg.

With MultiBOS, you can access, install, maintain, update, and customize the standby BOS either during the setup or any subsequent customization operations. Installing technology level (TL) updates to the standby BOS does not change system files on the active BOS, which means that you can do a concurrent update of the standby BOS while the active BOS remains in production.

Here are the general requirements for MultiBOS:

- ▶ The 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 VG limits.

7.7.1 Standby BOS setup

The following section describes the MultiBOS standby setup operation and procedures.

Standby BOS setup operation

The standby BOS setup creates the standby BOS instance. The following actions are performed after the standby BOS setup starts:

1. The MultiBOS methods are initialized.
2. 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 a 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.
3. The standby logical volumes are created based on `image.data` attributes. The active and standby logical volumes are marked with unique tags in the LVCB. The **multibos** utility uses these tags to identify copied logical volumes. If the active logical volume names are classic names, such as `hd2`, `hd4`, and `hd5`, then the `bos_ prefix` is added to create a standby name. If the active logical volume names have the `bos_ prefix`, the prefix is removed to create a new standby name.
4. 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 LVCB and `/etc/filesystems`. The **multibos** utility uses these tags to identify copied logic volumes. The `/bos_inst` prefix is added 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`.
5. The standby file systems are mounted.
6. A list of files that will be copied from the active BOS is generated. This list is composed of the current files in the copied active BOS file systems without any files that you excluded with the optional exclude list.
7. The list of files is copied to the standby BOS file systems by using the backup and restore utilities.
8. Any optional customization is performed, which may include the installation of file set updates or other software.
9. The standby boot image is created and written to the standby BLV by using the AIX **bosboot** command. You can block this step by setting the `-N` flag. Use the `-N` flag only if you are an experienced administrator and have a good understanding the AIX boot process.
10. 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 by using the `-t` flag.

Note: The LVM limits the maximum length of a logical volume name to 15 characters, which 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 by 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 use the standby BOS commands.

Complete the following steps:

1. Run the `multibos -s` command to create the MultiBOS instance. Run the command in preview mode beforehand by adding the `-p` flag. You can include the `-X` flag to allow for automatic file system expansion if space is needed during the instance setup operation.

After you see “Return Status = SUCCESS”, then you may run the command without the `-p` flag. The log file for every MultiBOS operation is `/etc/multibos/logs/op.aalog`, as shown in Example 7-82.

Example 7-82 Previewing the standby BOS instance creation

```
# multibos -sXp
Initializing multibos methods ...
Initializing log /etc/multibos/logs/op.aalog ...
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.aalog
Return Status = SUCCESS
```

2. Run the normal command without a preview. Example 7-83 shows the MultiBOS stages that create the instance.

Example 7-83 Creating the MultiBOS instance

```
# multibos -sX
Initializing multibos methods ...
Initializing log /etc/multibos/logs/op.aalog ...
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.aalog
```

3. Check the file systems and logical volumes that are created by the `lsvg` and `lsfs` commands, 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
```

4. Because both active and standby instances are on the same `hdisk0`, you can set the boot list to first check the standby instance, and then check the active instance.

The current active instance has the normal AIX logical volumes and mount points, and the standby instance has the logical volumes with the `bos_` prefix, as shown in Example 7-84.

Setting the boot list to put the standby instance first and then the current active instance requires specifying the BLV name, as shown in Example 7-85.

Example 7-85 Setting the boot list to point to the 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
```

5. Restart and see how the instance activates. Determine and confirm which instance was used to start, as shown in Example 7-86.

Example 7-86 Restarting the system and checking which instance was used to start

```
# 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

# 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      4       4       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

# 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 Example 7-86 on page 203, the system started by using the standby image, which now is the active image. It has the same mount points but different logical volume names, which is correct.

Important: The `bosboot -a` command always updates the boot image of the current BLV from which you started. In our case, the current BLV is `bos_hd5`. So, running the `bosboot -a` command updates the active boot image (`bos_hd5`) in the current situation. To update the other standby image (`hd5`), run the `multibos` command with the `-B` option.

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 images

```

# bootinfo -b
hdisk0
# bootinfo -v
bos_hd5
# bosboot -ad hdisk0
bosboot: Boot image is 57372 512 byte blocks.

# multibos -B
Initializing multibos methods ...
Initializing log /etc/multibos/logs/op.aalog ...
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.aalog
Return Status = SUCCESS

```

In MultiBOS, what makes the logical volumes and file systems of the instances unique is the tag number that is on top of each logical volume and attached to each file system information inside the `/etc/filesystems` file. Example 7-88 shows the MultiBOS unique tagging.

Example 7-88 Checking the 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 want to remove an instance (either the current active one or the other standby one), start by using the instance that you want to keep, and then run the **multibos** command with option **-R**. This command removes the other inactive instance, as shown in Example 7-89.

Example 7-89 Removing the MultiBOS instance

```

# multibos -RX
Initializing multibos methods ...
Initializing log /etc/multibos/logs/op.aalog ...
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.aalog
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
hd11admin        jfs2      1         1         1    open/syncd  /admin
lg_dump1v        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 |

Example 7-89 on page 205 shows the removal process of the inactive instance. Because the current instance is bos_LVNAME, it is kept with no issues with the normal regular mount points, as shown in Example 7-89 on page 205.

If you want to keep only the normal names, start by using the normal hd5 BLV to remove the other bos_LVNAME instance.

Related publications

The publications that are listed in this section are considered suitable for a more detailed description of the topics that are covered in this book.

IBM Redbooks

The following IBM Redbooks publications provide more information about the topics in this document. Some publications that are referenced in this list might be available in softcopy only.

- ▶ *IBM AIX Version 7.1 Differences Guide*, SG24-7910
- ▶ *IBM PowerHA SystemMirror V7.2.3 for IBM AIX and V7.22 for Linux*, SG24-8434
- ▶ *IBM PowerVC Version 1.3.2 Introduction and Configuration*, SG24-8199
- ▶ *IBM PowerVM Virtualization Introduction and Configuration*, SG24-7940
- ▶ *IBM PowerVM Virtualization Managing and Monitoring*, SG24-7590
- ▶ *Implementing IBM VM Recovery Manager for IBM Power Systems*, SG24-8426
- ▶ *iSCSI Implementation and Best Practices on IBM Storwize Storage Systems*, SG24-8327
- ▶ *NIM from A to Z in AIX 5L*, SG24-7296

You can search for, view, download, or order these documents and other Redbooks, Redpapers, web docs, drafts, and additional materials, at the following website:

ibm.com/redbooks

Online resources

These websites are also relevant as further information sources:

- ▶ AIX Auditing best practices
<https://www.ibm.com/support/pages/aix-auditing-best-practices>
- ▶ AIX Toolbox for Linux Applications Overview
<https://www.ibm.com/support/pages/aix-toolbox-linux-applications-overview>
- ▶ AIX Open Source Software community forum Kubernetes on AIX user guide
<https://www.ibm.com/developerworks/community/forums/html/topic?id=d05ec6c1-bce4-48c9-9867-63d80db815be&ps=25>
- ▶ AIX Web Download Pack
<https://www-01.ibm.com/marketing/iwm/iwm/web/pickUrxNew.do?source=aixbp>
- ▶ AIX WPARs - How to guide
<https://www.ibm.com/support/pages/aix-wpars-how>
- ▶ amepat command documentation
https://www.ibm.com/support/knowledgecenter/ssw_aix_72/a_commands/amepat.html

- ▶ Ansible home page
<https://www.ansible.com>
- ▶ Ansible playbooks developed by the AIX open source community
<https://github.com/aixoss/ansible-playbooks>
- ▶ Audit event listings
https://www.ibm.com/support/knowledgecenter/en/ssw_aix_72/security/audit_events.html
- ▶ Audit events file format
https://www.ibm.com/support/knowledgecenter/en/ssw_aix_72/filesreference/events.html
- ▶ Capacity on Demand (CoD) activation code lookup
<http://www-912.ibm.com/pod/pod>
- ▶ Chef Infra Overview
https://docs.chef.io/chef_overview.html
- ▶ Chef Supermarket
<https://supermarket.chef.io/cookbooks/aix>
- ▶ Cloud Automation Manager content providers
https://www.ibm.com/support/knowledgecenter/en/SS2L37_3.2.1.0/content/cam_content_terraform_provider.html
- ▶ Configuring YUM on AIX
<https://ftp.software.ibm.com/aix/freeSoftware/aixtoolbox/ezinstall/ppc/README-yum>
<https://developer.ibm.com/articles/configure-yum-on-aix/>
- ▶ Creating an application WPAR
https://www.ibm.com/support/knowledgecenter/en/ssw_aix_72/workloadpartitions/create-app.html
- ▶ Db2 encryption using Encrypted File System (EFS)
https://www.ibm.com/support/knowledgecenter/SSEPGG_10.1.0/com.ibm.db2.luw.admin.sec.doc/doc/c0055327.htm
- ▶ Google Cloud
<https://cloud.google.com/ibm>
- ▶ IBM AIX Workload Partition (WPAR) documentation
https://www.ibm.com/support/knowledgecenter/ssw_aix_72/workloadpartitions/wpar-kickoff.html
- ▶ IBM Cloud
<https://www.ibm.com/cloud/power-virtual-server>
- ▶ IBM Cloud Management Console (IBM CMC) Enterprise Pool 2.0 documentation
<https://ibmcmc.zendesk.com/hc/en-us/articles/360021928094-Enterprise-Pools-2-0>
- ▶ IBM CMC FAQ
<https://ibmcmc.zendesk.com/hc/en-us/sections/207305647-FAQ>

- ▶ IBM PowerHA compatibility matrix
<http://www-03.ibm.com/support/techdocs/atmsastr.nsf/WebIndex/TD101347>
- ▶ IBM PowerHA known fixes
https://aix.software.ibm.com/aix/ifixes/PHA_Migration/ha_install_mig_fixes.htm
- ▶ IBM PowerHA SystemMirror lifecycle
<http://www-01.ibm.com/support/docview.wss?uid=isg3T1023563>
- ▶ IBM PowerHA SystemMirror release notes
https://www.ibm.com/support/knowledgecenter/SSPHQG_7.2/navigation/releasenotes.htm
- ▶ IBM PowerSC Multifactory Authentication components and requirements
https://www.ibm.com/support/knowledgecenter/SS7FK2_1.2/com.ibm.powersc.mfa.install/pmfa_install_requirements.html
- ▶ IBM TouchToken concepts
https://www.ibm.com/support/knowledgecenter/SS7FK2_1.2/com.ibm.powersc.mfa.install/pmfa_totp_concepts.html
- ▶ IBM VM Recovery Manager high availability and disaster recovery (DR) V1.4 announcement letter
<https://www-01.ibm.com/common/ssi/cgi-bin/ssialias?infotype=AN&subtype=CA&htmlfid=897/ENUS219-428&appname=USN>
- ▶ Introduction to Active Memory Expansion (AME)
https://www.ibm.com/support/knowledgecenter/en/ssw_aix_72/performance/intro_ame_process.html
- ▶ Journaled File System (JFS) and Enhanced Journaled File Systems (JFS2) differences
https://www.ibm.com/support/knowledgecenter/en/ssw_aix_72/devicemanagement/fs_jfs2.html
- ▶ Linux on Power supported distributions
<https://www.ibm.com/support/knowledgecenter/en/linuxonibm/laam/laamdistros.htm>
- ▶ lvmstat manual page
https://www.ibm.com/support/knowledgecenter/en/ssw_aix_72/l_commands/lvmstat.html#lvmstat__lvmstat_r
- ▶ Managing Cloud Automation Manager templates from Template Designer
https://www.ibm.com/support/knowledgecenter/en/SS2L37_3.2.1.0/cam_bpd_template.html
- ▶ Managing multiple instances of altinst_rootvg and applying fixes to them
<https://www.ibm.com/support/pages/managing-multiple-instances-altinstrootvg-and-applying-fixes-them>
- ▶ Managing WPAR clients
https://www.ibm.com/support/knowledgecenter/en/ssw_aix_72/install/concepts_manage_wpar_clients.html
- ▶ Multiple alternative rootvg criteria documentation
<https://www.ibm.com/support/pages/multiple-alternate-rootvg-criteria>

- ▶ OpenSSL cryptographic library home documentation
<https://www.openssl.org/>
- ▶ PowerVM NovaLink on IBM Knowledge Center documentation
https://www.ibm.com/support/knowledgecenter/en/POWER9/p9eig/p9eig_kickoff.htm
- ▶ Puppet Enterprise home page
<https://puppet.com>
- ▶ RSA SecurID Access Authentication Agent Implementation Guide for AIX 7.1
<https://community.rsa.com/docs/DOC-62752>
- ▶ RSA SecurID Access Authentication Agent for Pluggable Authentication Module (PAM) 7.1.0.2 announcement
<https://community.rsa.com/docs/DOC-77540>
- ▶ Shared Storage Pool (SSP) setup
<https://www.ibm.com/support/pages/creating-simple-ssp-among-two-vio-servers>
- ▶ Skytap
<https://www.skytap.com/product/cloud-platform/power-systems-aix-linux-ibmi>
- ▶ Software requirements for Enterprise Pool 2.0 LPARs
https://www-01.ibm.com/common/ssi/ShowDoc.wss?docURL=/common/ssi/rep_ca/3/872/E NUSAG19-0003/index.html&lang=en&request_locale=en#hardx
- ▶ Structure of Cloud Automation Manager templates
https://www.ibm.com/support/knowledgecenter/en/SS2L37_3.2.0.0/cam_struct_template.html
- ▶ Subsystem Device Path Control Module (SDDPCM) to AIX Path Control Module (AIXPCM) migration guide
<https://www.ibm.com/support/pages/how-migrate-sddpcm-aixpcm>
- ▶ Terraform documentation
<https://www.terraform.io>
- ▶ Terraform supported provisioner listing
<https://www.terraform.io/docs/provisioners/index.html>
- ▶ Versioned WPAR creation guide
<https://www.ibm.com/support/pages/versioned-wpar-creation>

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Redbooks

IBM AIX Enhancements and Modernization

(0.2"spine)
0.17"->0.473"
90->249 pages



SG24-8453-00

ISBN 0738458287

Printed in U.S.A.

Get connected

