IBM

AIX 5L Differences Guide Version 5.3 Edition

- AIX - The industrial strength UNIX operating system
- An expert's guide to the new release
- AIX 5L Version 5.3 enhancements explained

Adrian Demeter
Shiv Dutta
Armin Röll
SeongLul Son

Redbooks

ibm.com/redbooks

International Technical Support Organization

AIX 5L Differences Guide Version 5.3 Edition

December 2004

SG24-7463-00

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

First Edition (December 2004)

This edition applies to AIX 5L for POWER Version 5.3, program number 5765-G03.

Contents

Figures

Tables

Examples

Notices

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

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

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

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

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

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

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

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

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

COPYRIGHT LICENSE:
This information contains sample application programs in source language, which illustrates 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. 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 IBM's application programming interfaces.

Trademarks

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

@server®	Enterprise Storage Server®	POWER5™
@server®	Hypervisor™	PTX®
Redbooks (logo) ™	HACMP™	pSeries®
eServer™	i5/OS™	QMF™
ibm.com®	IBM®	Redbooks™
AFS®	Micro Channel®	RDN™
AIX 5L™	Micro-Partitioning™	RS/6000®
AIX®	PowerPC®	SecureWay®
DB2®	POWER™	Versatile Storage Server™
developerWorks®	POWER2™	ViaVoice®
DFS™	POWER4™	Virtualization Engine™

The following terms are trademarks of other companies:

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

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

Preface

This IBM® Redbook focuses on the differences introduced in AIX® 5L™ Version 5.3 when compared to AIX 5L Version 5.2. It is intended to help system administrators, developers, and users understand these enhancements and evaluate potential benefits in their own environments.

AIX 5L Version 5.3 introduces many new features, including NFS Version 4 and Advanced Accounting, and exploits the advanced capabilities of POWER5™ equipped severs, such as Virtual SCSI, Virtual Ethernet, SMT, Micro-Partitioning™ technology, and others. There are many other enhancements available with AIX 5L Version 5.3, and you can explore them in this redbook.

For customers who are not familiar with the enhancements of AIX 5L through Version 5.2, a companion publication, *AIX 5L Differences Guide Version 5.2 Edition*, SG24-5765 is available.

The team that wrote this redbook

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

Adrian Demeter is a senior IT consultant and technical manager at IBM Business Partner, GC System a.s. in Prague, Czech Republic. He holds the Ing. academic degree in Electrical engineering and technical cybernetics from Czech Technical University (CVUT), Prague. He has 20 years of experience in computers and electronics, of which 12 years are in UNIX® and AIX. Adrian is responsible for project design, implementation, and support of high-end computer systems in Europe. He also teaches advanced IBM courses on AIX, HACMP™, Cluster 1600, and others. He has written extensively on hardware architectures, UNIX, communications, security, high availability, clustering, and parallel computing.

Shiv Dutta works for IBM US in Austin, providing technical assistance to ISVs in the areas of application benchmarking, porting, performance tuning and sizing guides for pSeries®. He also provides the pSeries and AIX roadmaps for the ISV community. Shiv has had extensive experience in application development, database design, and system administration for RS/6000®, SP, and AIX. He has been on the faculty of several colleges, universities, and training institutions where he taught various C and UNIX courses. Shiv has authored a number of

articles on IT topics for the IBM developerWorks® Web site. He has a Ph.D. in Physics from Ohio University and Project Management Professional (PMP) Certification from the Project Management Institute. He has a few published patent disclosures to his credit.

Armin Röll works as a pSeries system engineer in Germany. He has nine years of experience in pSeries and AIX pre-sales technical support and, as a teamleader, he fosters the AIX skills community. He holds a degree in experimental physics from the University of Hamburg, Germany. He co-authored the AIX Version 4.3.3 and the AIX 5L Version 5.0 Differences Guide Redbooks™.

SeongLul Son is an advisory education specialist working for IBM Korea. He has eight years of experience in networking, e-learning, and operating systems in various environments. His areas of expertise include AIX, TCP/IP networking, and HACMP. He is an IBM certified Advanced Technical Expert - pSeries and AIX 5L and Cisco Certified Network Professionl (CCNP). He has written extensively on system management, file system management, communcations, and security.

The team that authored this publication was managed by:

Scott Vetter
IBM Austin

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

Mathew Accapadi, David K Altobelli, Dwip Banerjee, Kavitha Baratakke, Anne Bohy, Joyce Bolton, Paul S. Bostrom, Matthew T. Brandyberry, Larry Brenner, Luke Browning, Guoyou Chen, Diane Chung, Mathew Cronk, Zhi-wei Dai, Medha Date, Christopher DeRobertis, Saurabh Desai, Saravanan Devendran, Janet Ellsworth, John Emmons, Jianhua Feng, Lilian Fernandes, Paul Finley, Greg Flaig, Arnold Flores, Eric P Fried, David Geise, Denise Genty, Jeffrey George, Mark A. Grubbs, Julianne Haugh, David A. Hepkin, Mary Hoetzel, Vinit Jain, Patrick Laffey, Joseph Lampit, Zachary Loafman, Yantian Lu, Sheena Madan, Michael Mall, Richard Mankowski, Christopher McDonald, Laura L. McWilliams, Alex Medvedev, Steven Molis, James Moody, Frank Nichols, Ketan Pancholi, Shiva Persaud, Marcos Quezada, Dino Quintero, Meenatchi Ramalingam, Eduardo Reyes, Mark Rogers, Adam Ruprecht, Scot Sakolish, Carolyn Scherrer, Michael Schmidt, Ravi Shankar, Johnny Shieh, Luc Smolders, Andrew N Solomon, Michael Spornhauer, Marc Stephenson, Kent Stuiber, Jean-philippe Sugarbroad, Robert Thompson, Duyen Tong, Marvin Toungate, Kedron J. Touvell, Basu Vaidyanathan, Vasu Vallabhaneni, Marcos A Villarreal, Drew Walters, Wayne Wheeler, Ann Wigginton, Andy Wong, Jason Wu, Wu Zheng

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience

with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

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

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

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

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

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

  ibm.com/redbooks

- Send your comments in an Internet note to:

  redbook@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. JN9B, Building 905 Internal Zip 9053D003
  11501 Burnet Road
  Austin, Texas 78758-3493

1

Virtualization

This chapter discusses the virtualization engine technologies that are now integrated into the p5 servers, AIX 5L Version 5.3. It is also found in Chapter 3 of *Advanced POWER Virtualization on IBM @server p5 Servers: Introduction and Basic Configuration*, SG24-7940.

These technologies include:

- POWER™ Hypervisor™
- Micro-Partitioning
- Virtual Ethernet
- Shared Ethernet Adapter
- Virtual SCSI
- PLM

1.1 Advanced POWER Virtualization feature

This section describes the packaging and ordering details for the Advanced POWER Virtualization feature available on p5 systems.

The Advanced POWER Virtualization feature is a combination of hardware enablement that includes the following components that are available together as a single priced feature:

- Firmware enablement for Micro-Partitioning
- Installation image for the Virtual I/O Server software, which supports:
  - Shared Ethernet Adapter
  - Virtual SCSI server
- Partition Load Manager

Note that virtual Ethernet is available without this feature when the server is attached to an HMC, as well as LPAR and dynamic LPAR. SMT is available on the base hardware with no additional features required.

Figure 1-1 shows an overview of the different parts of the hardware order that enables firmware and includes the software orders.

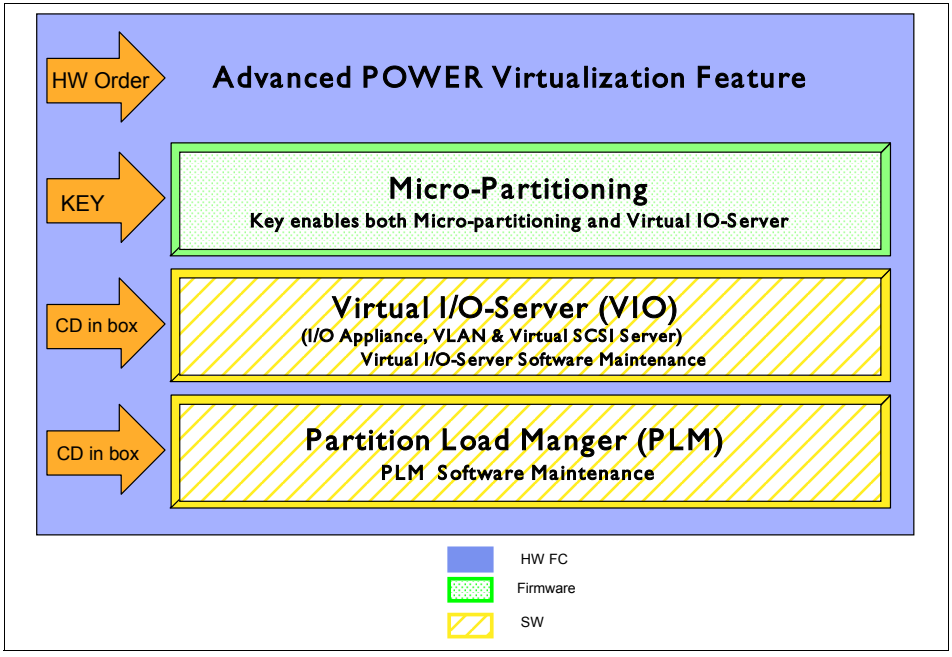

Figure 1-1 Advanced POWER Virtualization feature

When the hardware feature is specified with the initial system order, the firmware is shipped activated to support Micro-Partitioning and the Virtual I/O Server. For upgrade orders, IBM will ship a key to enable the firmware similar to the CUoD key.

Figure 1-2 shows the HMC panel where you enable the Virtualization Engine™ Technologies.

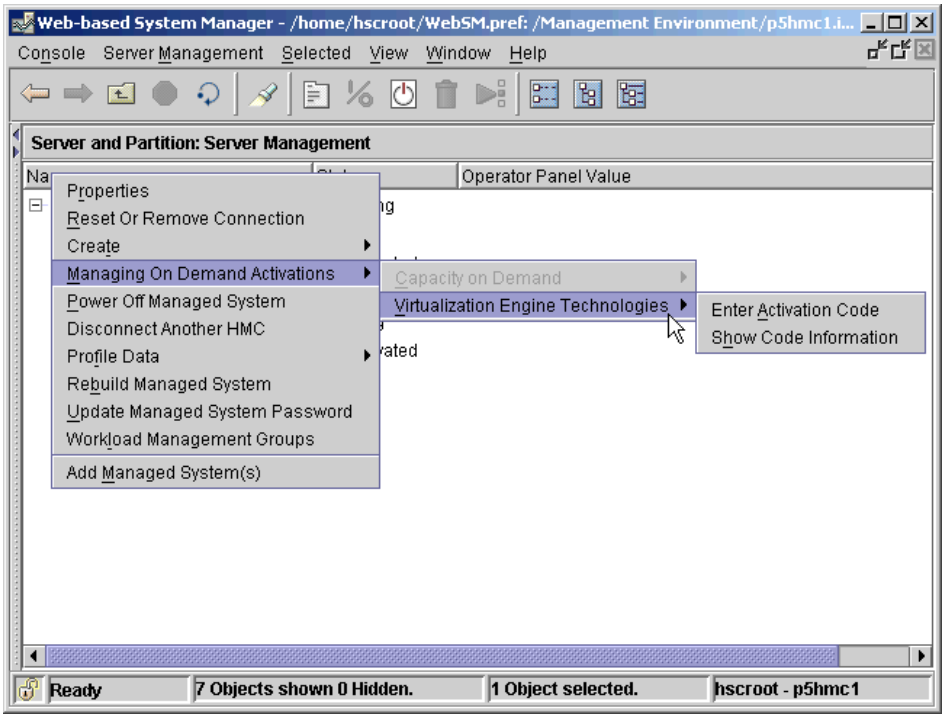

Figure 1-2 HMC panel to enable the Virtualization Engine Technologies

Virtual I/O Server and Partition Load Manager are licensed software components of the Advanced POWER Virtualization feature. They contain one charge unit per installed processor, including software maintenance. The initial software license charge for Virtual I/O Server and PLM is included in the price of the Advanced POWER Virtualization feature. The related hardware features that include Virtual I/O Server and PLM are:

- 9111-520 Feature 7940
- 9113-550 Feature 7941
- 9117-570 Feature 7942

For each Virtual I/O Server V1.1 license ordered, an order for either the one-year (5771-VIO) or three-year (5773-VIO) Software Maintenance (SWMA) is also submitted.

The processor-based license enables you to install multiple Virtual I/O Server partitions on a single server to provide redundancy and to spread the Virtual I/O Server workload across multiple partitions.

The Virtual I/O Server resides in a POWER5 partition as a single function appliance that is created using the Hardware Management Console (HMC). The Virtual I/O Server installation media ships with the POWER5 system that is configured with the Advanced POWER Virtualization feature and supports network install (NIMOL from HMC) or CD installation. It supports AIX 5L Version 5.3, SUSE LINUX Enterprise Server 9 for POWER, and Red Hat Enterprise Linux AS for POWER Version 3 as Virtual I/O clients.

The Virtual I/O Server provides the Virtual SCSI server and Shared Ethernet Adapter virtual I/O function to client partitions (Linux or AIX). This POWER5 partition is not intended to run applications or for general user login.

For each Partition Load Manager V1.1 (5765-G31) license ordered, an order for either the one-year (5771-PLM) or three-year (5773-PLM) Software Maintenance (SWMA) must also be submitted. The Software Maintenance for the Partition Load Manager is priced on a per processor basis, by processor group.

The Partition Load Manager for AIX 5L helps customers to maximize the utilization of processor and memory resources on pSeries and p5 servers that support dynamic logical partitioning. Within the constraints of a user-defined policy, resources are automatically moved to partitions with a high demand, from partitions with a lower demand. Resources that would otherwise go unused can now be more fully utilized.

Partition Load Manager supports management of any dynamic LPAR running AIX 5L Version 5.2 with the 5200-04 Recommended Maintenance Package or AIX 5L Version 5.3 or later. The Partition Load Manager server can run on any system with the appropriate level of IBM AIX.

The summary of features is as follows:

- Automated processor and memory reconfiguration
- Real-time partition configuration and load statistics
- Support for dedicated and shared processor partition groups
- Support for manual provisioning of resources
- Graphical user interface (Web-based System Manager)

1.2 Introduction to the POWER Hypervisor

The POWER Hypervisor is an essential element of the IBM Virtualization Engine system technologies implemented in the POWER5 processor based @server family of products. Combined with features designed into the POWER5 processor, the POWER Hypervisor delivers functions that enable other system technologies including Micro-Partitioning, virtualized processors, IEEE VLAN compatible virtual switch, virtual SCSI adapters, and virtual consoles.

The POWER Hypervisor is a component of system firmware that will always be installed and activated, regardless of system configuration. It operates as a hidden partition, with no processor resources assigned to it.

Newly architected *hypervisor calls* (hcalls) provide a means for the operating system to communicate with the POWER Hypervisor, allowing more efficient use of physical processor capacity.

The POWER Hypervisor is a key component of the functions shown in Figure 1-3. It performs the following tasks:

- Provides an abstraction layer between the physical hardware resources and the logical partitions using them
- Enforces partition integrity by providing a security layer between logical partitions
- Controls the dispatch of virtual processors to physical processors
- Saves and restores all processor state information during logical processor context switch
- Controls hardware I/O interrupts management facilities for logical partitions

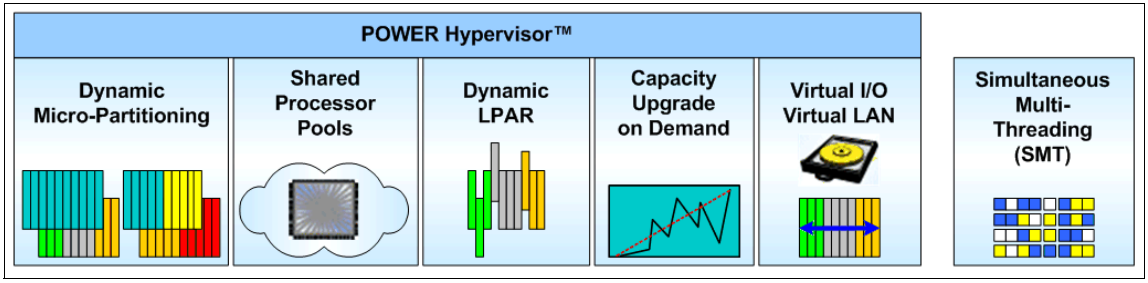

Figure 1-3 POWER Hypervisor functions

The POWER Hypervisor, acting as the abstraction layer between the system hardware and the logical partitions, allows multiple operating systems to run on POWER5 technology with little or no modifications.

1.2.1 POWER Hypervisor implementation

The POWER4™ processor introduced support for logical partitioning with a new privileged processor state called *hypervisor mode*. It is accessed using hypervisor call functions, which are generated by the operating system kernel running in a partition. Hypervisor mode allows for a secure mode of operation that is required for various system functions where logical partition integrity and security are required. The hypervisor validates that the partition has ownership of the resources it is attempting to access, such as processor, memory, and I/O, then completes the function. This mechanism allows for complete isolation of partition resources.

In the POWER5 processor, further design enhancements are introduced that enable the sharing of processors by multiple partitions. The *hypervisor decrementer* (HDECR) is a new hardware facility in the POWER5 design that is programmed to provide the POWER Hypervisor with a timed interrupt independent of partition activity. HDECR interrupts are routed directly to the POWER Hypervisor, and use only POWER Hypervisor resources to capture state information from the partition. The HDECR is used for fine-grained dispatching of multiple partitions on shared processors. It also provides a means for the POWER Hypervisor to dispatch physical processor resources for its own execution.

With the addition of shared partitions and SMT, a mechanism was required to track physical processor resource utilization at a processor thread level. System architecture for POWER5 introduces a new register called the *Processor Utilization Resource Register* (PURR) to accomplish this. It provides the partition with an accurate cycle count to measure activity during timeslices dispatched on a physical processor. The PURR is a POWER Hypervisor resource, assigned one per processor thread, that is incremented at a fixed rate whenever the thread running on a virtual processor is dispatched on a physical processor.

1.2.2 POWER Hypervisor processor dispatch

Multiple logical partitions configured to run with a pool of shared physical processors require a robust mechanism to guarantee the distribution of available processing cycles. The POWER Hypervisor manages this task in the POWER5 processor based system.

Shared processor partition resource definition is discussed in detail in 1.3, “Micro-Partitioning introduction” on page 14. This section introduces the concepts of how the POWER Hypervisor dispatches work from multiple partitions across physical processors.

Each shared processor partition is configured with a specific processor entitlement, based on a quantity of processing units, which is referred to as the partition's *entitled capacity*. The entitled capacity, along with a defined number of virtual processors, defines the physical processor resource that will be allotted to the partition. The POWER Hypervisor uses the POWER5 HDECR, which is programmed to generate an interrupt every 10 ms, as a timing mechanism for controlling the dispatch of physical processors to system partitions. Each virtual processor is guaranteed to get its entitled share of processor cycles during each 10 ms dispatch window.

Primary POWER Hypervisor hcalls

The primary POWER Hypervisor hcalls used by the operating system in the dispatch of a virtual processor are:

cede The cede call is used when a virtual processor or SMT thread becomes idle, allowing the POWER Hypervisor to dispatch other work.

confer The confer call is used to grant the remaining cycles in a dispatch interval to another virtual processor in the partition. It can be used when one virtual processor cannot make forward progress because it is waiting on an event to complete on another virtual processor, such as a lock miss.

prod The prod call is used to activate a virtual processor that has ceded or conferred processor cycles.

A virtual processor will always be in one of four logical states. These states are:

running Currently dispatched on a physical processor

runnable Ready to run, waiting for dispatch

not-runnable Has ceded or conferred its cycles

expired Consumed its full entitled cycles for the current dispatch window

Monitoring POWER Hypervisor hcalls

In AIX 5L Version 5.3, the **lparstat** command using the **-h** and **-H** flags will display hypervisor statistical data about many hcalls, including cede, confer, and prod. Using the **-h** flag adds summary hypervisor statistics to the default **lparstat** output.

The following shows an example of this command, collecting statistics for one five-second interval:

```
# lparstat -h 5 1
System configuration: type=Shared mode=Uncapped smt=On lcpu=4 mem=512 ent=0.50
%user  %sys  %wait  %idle physc %entc  lbusy   app  vcsw phint  %hypv hcalls
-----  ----  -----  ----- ----- ----- ------   ---  ---- -----  ----- ------
  0.0   0.5    0.0   99.4  0.00   1.0    0.0     -  1524     0    0.5   1542
```

Using the **-H** flag displays detailed hypervisor information, including statistics for many hcall functions. The command shows the following for each of these hcalls:

Number of calls Number of hypervisor calls made

Total Time Spent Percentage of total time spent in this type of call

Hypervisor Time Spent Percentage of hypervisor time spent in this type of call

Average Call Time Average call time for this type of call in nanoseconds

Maximum Call Time Maximum call time for this type of call in nanoseconds

```
# lparstat -H 5 1

System configuration: type=Shared mode=Uncapped smt=On lcpu=4 mem=512 ent=0.50

           Detailed information on Hypervisor Calls

Hypervisor        Number of    %Total Time   %Hypervisor   Avg Call    Max Call
  Call              Calls         Spent      Time Spent    Time(ns)    Time(ns)

remove                   2            0.0           0.0         417         550
read                    21            0.0           0.0         148         294
nclear_mod               0            0.0           0.0           1           0
page_init                3            0.0           0.0         774        2280
clear_ref                0            0.0           0.0           1           0
protect                  0            0.0           0.0           1           0
put_tce                 14            0.0           0.1         544         908
xirr                    10            0.0           0.0         536         758
eoi                     10            0.0           0.0         526         695
ipi                      0            0.0           0.0           1           0
cppr                     0            0.0           0.0           1           0
asr                      0            0.0           0.0           1           0
others                   0            0.0           0.0           1           0
enter                    5            0.0           0.0         397         685
cede                  1595            0.5          99.6        7918     1343207
migrate_dma              0            0.0           0.0           1           0
put_rtce                 0            0.0           0.0           1           0
confer                   0            0.0           0.0           1           0
prod                    27            0.0           0.1         672         922
```

get_ppp	1	0.0	0.0	1502	2579
set_ppp	0	0.0	0.0	1	0
purr	0	0.0	0.0	1	0
pic	1	0.0	0.0	309	410
bulk_remove	0	0.0	0.0	1	0
send_crq	0	0.0	0.0	1	0
copy_rdma	0	0.0	0.0	1	0
get_tce	0	0.0	0.0	1	0
send_logical_lan	0	0.0	0.0	1	0
add_logicl_lan_buf	0	0.0	0.0	1	0

The basic concept of this dispatch mechanism is illustrated in Figure 1-4. In this figure there are three logical partitions defined, sharing the processor cycles from two physical processors, spanning two 10 ms hypervisor dispatch intervals.

Logical partition 1 is defined with an entitlement capacity of 0.8 processing units, with two virtual processors. This allows the partition 80% of one physical processor for each 10 ms dispatch window for the shared processor pool. For each dispatch window, the workload is shown to use 40% of each physical processor during each dispatch interval. It is possible for a virtual processor to be dispatched more than one time during a dispatch interval. Note that in the first dispatch interval, the workload executing on virtual processor 1 is not a continuous utilization of physical processor resource. This can happen if the operating system confers cycles, and is reactivated by a prod hcall.

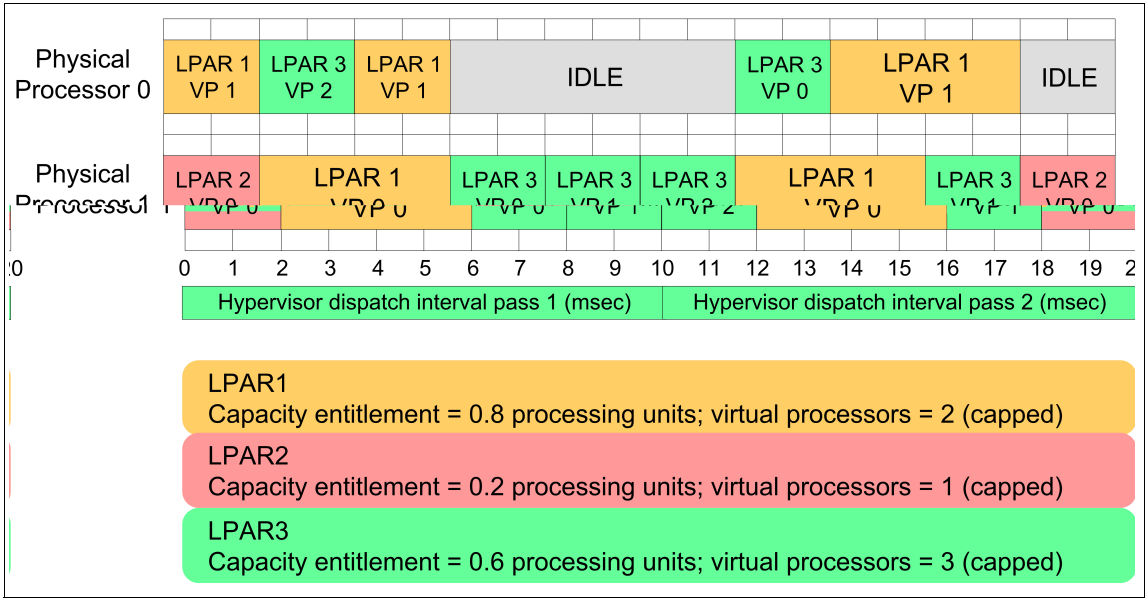

Figure 1-4 Micro-Partitioning processor dispatch

Logical partition 2 is configured with one virtual processor and a capacity of 0.2 processing units, entitling it to 20% usage of a physical processor during each dispatch interval. In this example, a worst case dispatch latency is shown for this virtual processor, where the 2 ms are used in the beginning of dispatch interval 1 and the last 2 ms of dispatch interval 2, leaving 16 ms between processor allocation.

Logical partition 3 contains 3 virtual processors, with an entitled capacity of 0.6 processing units. Each of the partition's three virtual processors consumes 20% of a physical processor in each dispatch interval, but in the case of virtual processor 0 and 2, the physical processor they run on changes between dispatch intervals. The POWER Hypervisor attempts to maintain physical processor affinity when dispatching virtual processors. It will always first try to dispatch the virtual processor on the same physical processor as it last ran on, and depending on resource utilization will broaden its search out to the other processor on the POWER5 chip, then to another chip on the same MCM, then to a chip on another MCM.

Processor dispatch communication

The dispatch mechanism utilizes hcalls to communicate between the operating system and the POWER Hypervisor. Implementing hcalls in the operating system is desirable for performance reasons since they minimize the idle time of a physical processor. Operating systems must be designed to use this new call to exploit the full potential of POWER5.

When a virtual processor is active on a physical processor and the operating system detects an inability to utilize processor cycles, it may *cede* or *confer* its cycles back to the POWER Hypervisor, enabling it to schedule another virtual processor on the physical processor for the remainder of the dispatch cycle. Reasons for a cede or confer may include the virtual processor running out of work and becoming idle, entering a spin loop to wait for a resource to free, or waiting for a long latency access to complete. There is no concept of *credit* for cycles that are ceded or conferred. Entitled cycles not used during a dispatch interval are lost.

A virtual processor that has ceded cycles back to the POWER Hypervisor can be reactivated using a *prod* hcall. If the operating system running on another virtual processor within the logical partition detects that work is available for one of its idle processors, it can use the prod hcall to signal the POWER Hypervisor to make the prodded virtual processor runnable again. Once dispatched, this virtual processor would resume execution at the return from the cede hcall.

In IBM AIX 5L Version 5.3, the **mpstat** command using the **-d** flag displays detailed affinity and migration statistics for AIX threads and dispatching statistics for logical processors.

```
# mpstat -d
System configuration: lcpu=4 ent=0.5

cpu     cs    ics  bound  rq push S3pull S3grd  S0rd  S1rd  S2rd  S3rd  S4rd  S5rd   ilcs   vlcs
  0  68598  38824      0   0    0      0     0  95.6   0.0   0.0   4.4   0.0   0.0 174110 237393
  1    291    244      0   0    0      0     0  90.9   7.4   0.0   1.7   0.0   0.0   1092 237759
  2  54514  30174      1   1    0      0     0  94.0   0.1   0.0   6.0   0.0   0.0   2756  71779
  3    751    624      0   0    0      0     0  91.3   2.9   0.0   5.8   0.0   0.0   1192  72971
ALL 124154  69866      1   1    0      0     0  94.8   0.1   0.0   5.1   0.0   0.0  89575 309951
```

The POWER Hypervisor dispatch affinity domains are defined as follows, and statistics for virtual processor dispatch across these domains is given by the **mpstat** command.

S0 The process redispatch occurs within the same logical processor. This happens in the case of SMT-enabled systems.

S1 The process redispatch occurs within the same physical processor, among different logical processors. This involves sharing of the L1, L2, and L3 cache.

S2 The process redispatch occurs within the same processor chip, but among different physical processors. This involves sharing of the L2 and L3 cache.

S3 The process redispatch occurs within the same MCM module, but among different processor chips.

S4 The process redispatch occurs within the same CEC plane, but among different MCM modules. This involves access to the main memory or L3-to-L3 transfer.

S5 The process redispatch occurs outside of the CEC plane.

As previously stated, the POWER Hypervisor will always first try to dispatch the virtual processor on the same physical processor that it last ran on, and depending on resource utilization will broaden its search out to the other processor on the POWER5 chip, then to another chip on the same MCM, then to a chip on another MCM.

Systems using DCMs share similar boundaries between processor cards, or SMP Flex cables, as MCMs do between MCMs.

1.2.3 POWER Hypervisor and virtual I/O

Detailed discussion of POWER5 *@server* virtual I/O implementation is found in various sections of this publication. This section introduces POWER Hypervisor involvement in the virtual I/O functions.

With the introduction of Micro-Partitioning, the ability to dedicate physical hardware adapter *slots* to each partition becomes impractical. Virtualization of I/O devices allows many partitions to communicate with each other, and access networks and storage devices external to the server, without dedicating physical I/O resources to an individual partition. Many of the I/O virtualization capabilities introduced with the POWER5 processor based *@server* products are accomplished by functions designed into the POWER Hypervisor.

The POWER Hypervisor does not *own* any physical I/O devices, and it does not provide virtual interfaces to them. All physical I/O devices in the system are owned by logical partitions. Virtual I/O devices are owned by the Virtual I/O Server, which provides access to the real hardware that the virtual device is based on.

The POWER Hypervisor implements the following operations required by system partitions to support virtual I/O:

- Provides control and configuration structures for virtual adapter images required by the logical partitions
- Allows partitions controlled and secure access to physical I/O adapters in a different partition

Along with these operations, the POWER Hypervisor allows for the virtualization of I/O interrupts. To maintain partition isolation, the POWER Hypervisor controls the hardware interrupt management facilities. Each logical partition is provided controlled access to the interrupt management facilities using hcalls. Virtual I/O adapters and real I/O adapters use the same set of hcall interfaces.

I/O types supported

Three types of virtual I/O adapters are supported by the POWER Hypervisor:

- SCSI
- Ethernet
- Serial console (virtual TTY)

Virtual I/O adapters are defined by system administrators during logical partition definition. Configuration information for the virtual adapters is presented to the partition operating system by the system firmware.

Virtual SCSI support

The p5 server uses SCSI as the mechanism for virtual storage devices. This is accomplished using two paired adapters: a virtual SCSI server adapter and a virtual SCSI client adapter. These adapters are used to transfer SCSI commands between partitions. To the operating system, the virtual SCSI client adapter is no different than a typical SCSI adapter. The SCSI server, or target, adapter is responsible for executing any SCSI command it receives. It is owned by the Virtual I/O server partition. How this SCSI command is executed is dependent on the hosting partition operating system. See 1.6, “Virtual SCSI introduction” on page 46 for a detailed discussion of virtual SCSI.

Virtual SCSI operation is dependant on two system functions implemented in the POWER Hypervisor.

- Message queuing function that enables the passing of messages between partitions, with an interrupt mechanism to confirm receipt of the message
- DMA function between partitions that provides control structures for secure transfers between logical partition memory spaces

Virtual Ethernet switch support

The POWER Hypervisor provides a Virtual Ethernet switch function that allows partitions on the same server a means for fast and secure communication. It is based on the IEEE 802.1Q VLAN standard. No physical I/O adapter is required when creating a VLAN connection between partitions, and no access to an outside network is required. Each partition operating system sees the VLAN switch as an Ethernet adapter, without the physical link properties and asynchronous data transmit operations. See 1.4, “Virtual Ethernet introduction” on page 32 for a detailed discussion of Virtual Ethernet.

Virtual Ethernet transmit operations are performed synchronously and are complete as soon as the hypervisor call to send a frame returns. Receive operations are performed using a pool of buffers provided to the POWER Hypervisor for receiving frames. As incoming frames are received by the adapter, the POWER Hypervisor sends a virtual interrupt back to the device driver.

Each Virtual Ethernet adapter can also be configured as a trunk adapter. This enables layer-2 bridging to a physical Ethernet adapter to be performed, extending the Virtual Ethernet network outside the server to a physical Ethernet network. If a partition sends an Ethernet frame to a MAC address that is unknown by the POWER Hypervisor virtual switch, it will be forwarded to any trunk adapter defined for the same VLAN ID.

Virtual TTY console support

Each partition needs to have access to a system console. Tasks such as operating system install, network setup, and some problem analysis activities require a dedicated system console. The POWER Hypervisor provides virtual console using a virtual TTY or serial adapter and a set of hypervisor calls to operate on them.

Depending on the system configuration, the operating system console can be provided by the Hardware Management Console (HMC) virtual TTY, or from a terminal emulator connected to physical serial ports on the system's service processor.

1.3 Micro-Partitioning introduction

The concept of Micro-Partitioning allows the resource definition of a partition to allocate fractions of processors to the partition.

- On POWER4 systems, all partitions are considered *dedicated*, in that the processors assigned to a partition can only be in whole multiples and only used by that partition.
- On POWER5 systems, you can choose between dedicated processor partitions and shared processor partitions using Micro-Partitioning.

The benefit of Micro-Partitioning is that it allows increased overall utilization of system resources by automatically applying only the required amount of processor resource needed by each partition. Sales material tends to discuss Micro-Partitioning in terms of fractional processor units; however, resource can also be defined as increments greater than a single processor.

The POWER Hypervisor continually adjusts the amount of processor capacity allocated to each shared processor partition and any excess capacity unallocated based on current partition profiles within a shared pool. Tuning parameters allow the administrator extensive control over the amount of processor resources that each partition can use.

This section discusses the following topics about Micro-Partitioning:

- Shared processor partitions
- Shared pool overview
- CUoD
- Dynamic LPAR
- Limitations of Micro-Partitioning

1.3.1 Shared processor partitions

The virtualization of processors enables the creation of a partitioning model which is fundamentally different from the POWER4 systems, where whole processors are assigned to partitions and are owned by them. In the new model, physical processors are abstracted into virtual processors that are then assigned to partitions, but the underlying physical processors are shared by these partitions.

Virtual processor abstraction is implemented in the hardware and microcode. From an operating system perspective, a virtual processor is indistinguishable from a physical processor. The key benefit of implementing partitioning in the hardware allows any operating system to run on POWER5 technology with little or no changes. Optionally, for optimal performance, the operating system can be enhanced to exploit shared processor pools more in-depth, for instance, by voluntarily relinquishing CPU cycles to the hardware when they are not needed. AIX 5L Version 5.3 is the first version of AIX 5L that includes such enhancements.

Micro-Partitioning allows for multiple partitions to share one physical processor. Partitions using Micro-Partitioning technology are referred to as shared processor partitions.

A partition may be defined with a processor capacity as small as 10 processor units. This represents 1/10 of a physical processor. Each processor can be shared by up to 10 shared processor partitions. The shared processor partitions are dispatched and time-sliced on the physical processors under control of the POWER Hypervisor.

Micro-Partitioning is supported across the entire POWER5 product line from the entry to the high-end systems. Table 1-1 shows the maximum number of logical partitions and shared processor partitions supported on the different models.

Table 1-1 Micro-Partitioning overview on p5 systems

p5 servers	Model 520	Model 550	Model 570
Processors	2	4	16
Dedicated processor partitions	2	4	16
Shared processor partitions	20	40	160

It is important to point out that the maximums stated are supported by the hardware, but the practical limits based on production workload demands may be significantly lower.

Shared processor partitions still need dedicated memory, but the partitions' I/O requirements can be supported through Virtual Ethernet and Virtual SCSI. Utilizing all virtualization features, support for up to 254 shared processor partitions is planned.

The shared processor partitions are created and managed by the HMC. When you start creating a partition you have to choose between a shared processor partition and a dedicated processor partition.

When setting up a partition you have to define the resources that belong to the partition, such as memory and I/O resources. For processor shared partitions you have to configure these additional options:

- Minimum, desired, and maximum processing units of capacity
- The processing sharing mode, either capped or uncapped

  If the partition is uncapped, specify its variable capacity weight.
- Minimum, desired, and maximum virtual processors

These settings are the topic of the following sections.

Processing units of capacity

Processing capacity can be configured in fractions of 1/100 of a processor. The minimum amount of processing capacity which has to be assigned to a partition is 1/10 of a processor.

On the HMC, processing capacity is specified in terms of *processing units*. The minimum capacity of 1/10 of a processor is specified as 0.1 processing units. To assign a processing capacity representing 75% of a processor, 0.75 processing units are specified on the HMC.

On a system with two processors a maximum of 2.0 processing units can be assigned to a partition. Processing units specified on the HMC are used to quantify the minimum, desired, and maximum amount of processing capacity for a partition.

Once a partition is activated, processing capacity is usually referred to as capacity entitlement or entitled capacity.

Figure 1-5 on page 17 shows a graphical view of the definitions of processor capacity.

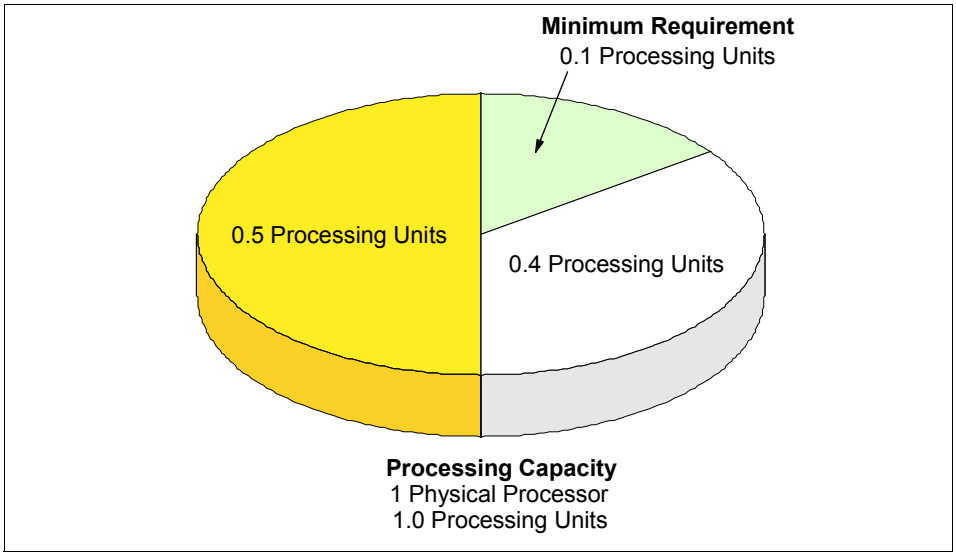

Figure 1-5 Processing units of capacity

Capped and uncapped mode

The next step in defining a shared processor partition is to specify whether the partition is running in a capped or uncapped mode.

Capped mode The processor unit never exceeds the assigned processing capacity.

Uncapped mode The processing capacity may be exceeded when the shared processing pools have available resource.

When a partition is run in an uncapped mode, you must specify the uncapped weight of that partition.

If multiple uncapped logical partitions require idle processing units, the managed system distributes idle processing units to the logical partitions in proportion to each logical partition's uncapped weight. The higher the uncapped weight of a logical partition, the more processing units the logical partition gets.

The uncapped weight must be a whole number from 0 to 255. The default uncapped weight for uncapped logical partitions is 128. A partition's share is computed by dividing its variable capacity weight by the sum of the variable capacity weights for all uncapped partitions. If you set the uncapped weight at 0, the managed system treats the logical partition as a capped logical partition. A logical partition with an uncapped weight of 0 cannot use more processing units than those that are committed to the logical partition.

Virtual processors

Virtual processors are the whole number of concurrent operations that the operating system can use. The processing power can be conceptualized as being spread equally across these virtual processors. Selecting the optimal number of virtual processors depends on the workload in the partition. Some partitions benefit from greater concurrence, while other partitions require greater power.

By default, the number of processing units that you specify is rounded up to the minimum number of virtual processors needed to satisfy the assigned number of processing units. The default settings maintain a balance of virtual processors to processor units. For example:

- If you specify 0.50 processing units, one virtual processor will be assigned.
- If you specify 2.25 processing units, three virtual processors will be assigned.

You also can use the Advanced tab in your partitions profile to change the default configuration and to assign more virtual processors.

At the time of publication, the maximum number of virtual processors per partition is 64.

A logical partition in the shared processing pool will have at least as many virtual processors as its assigned processing capacity. By making the number of virtual processors too small, you limit the processing capacity of an uncapped partition. If you have a partition with 0.50 processing units and 1 virtual processor, the partition cannot exceed 1.00 processing units because it can only run one job at a time, which cannot exceed 1.00 processing units. However, if the same partition with 0.50 processing units was assigned two virtual processors and processing resources were available, the partition could use an additional 1.50 processing units.

Dedicated processors

Dedicated processors are whole processors that are assigned to a single partition. If you choose to assign dedicated processors to a logical partition, you must assign at least one processor to that partition.

You cannot mix shared processors and dedicated processors in one partition.

By default, a powered-off logical partition using dedicated processors will have its processors available to the shared processing pool. When the processors are in the shared processing pool, an uncapped partition that needs more processing power can use the idle processing resources. However, when you power on the dedicated partition while the uncapped partition is using the processors, the activated partition will regain all of its processing resources. If you want to

prevent dedicated processors from being used in the shared processing pool, you can disable this function using the logical partition profile properties panels on the Hardware Management Console.

Note: You cannot disable the "`Allow idle processor to be shared`" function when you create a partition. You need to open the properties for the created partition and change it on the Processor tab.

1.3.2 Shared pool overview

The POWER Hypervisor schedules shared processor partitions from a set of physical processors that is called the shared processor pool. By definition, these processors are not associated with dedicated partitions.

In shared partitions there is no fixed relationship between virtual processors and physical processors. The POWER Hypervisor can use any physical processor in the shared processor pool when it schedules the virtual processor. By default, it attempts to use the same physical processor, but this cannot always be guaranteed. The POWER Hypervisor uses the concept of a home node for virtual processors, enabling it to select the best available physical processor from a memory affinity perspective for the virtual processor that is to be scheduled.

Affinity scheduling is designed to preserve the content of memory caches, so that the working data set of a job can be read or written in the shortest time period possible. Affinity is actively managed by the POWER Hypervisor since each partition has a completely different context. Currently, there is one shared processor pool, so all virtual processors are implicitly associated with the same pool.

Figure 1-6 on page 20 shows the relationship between two partitions using a shared processor pool of a single physical CPU. One partition has two virtual processors and the other a single one. The figure also shows how the capacity entitlement is evenly divided over the number of virtual processors.

When you set up a partition profile, you set up the desired, minimum, and maximum values you want for the profile. When a partition is started, the system chooses the partition's entitled processor capacity from this specified capacity range. The value that is chosen represents a commitment of capacity that is reserved for the partition. This capacity cannot be used to start another shared partition, otherwise capacity could be overcommitted.

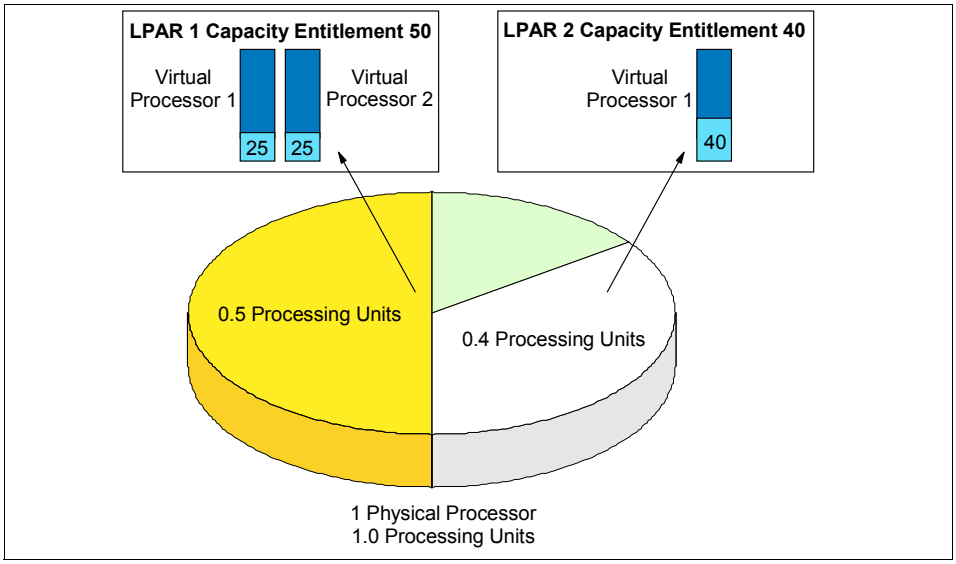

Figure 1-6 Distribution of capacity entitlement on virtual processors

When starting a partition, preference is given to the desired value, but this value cannot always be used because there may not be enough unassigned capacity in the system. In that case, a different value is chosen, which must be greater than or equal to the minimum capacity attribute. Otherwise, the partition cannot be started.

The entitled processor capacity is distributed to the partitions in the sequence the partitions are started. For example, consider a shared pool that has 2.0 processing units available.

Partitions 1, 2, and 3 are activated in sequence:

- Partition 1 activated
  Min. = 1.0, max = 2.0, desired = 1.5
  Allocated capacity entitlement: 1.5
- Partition 2 activated
  Min. = 1.0, max = 2.0, desired = 1.0
  Partition 2 does not start because the minimum capacity is not met.
- Partition 3 activated
  Min. = 0.1, max = 1.0, desired = 0.8
  Allocated capacity entitlement: 0.5

The maximum value is only used as an upper limit for dynamic operations.

Figure 1-7 shows the usage of a capped partition of the shared processor pool. Partitions using the capped mode are not able to assign more processing capacity from the shared processor pool than the capacity entitlement will allow.

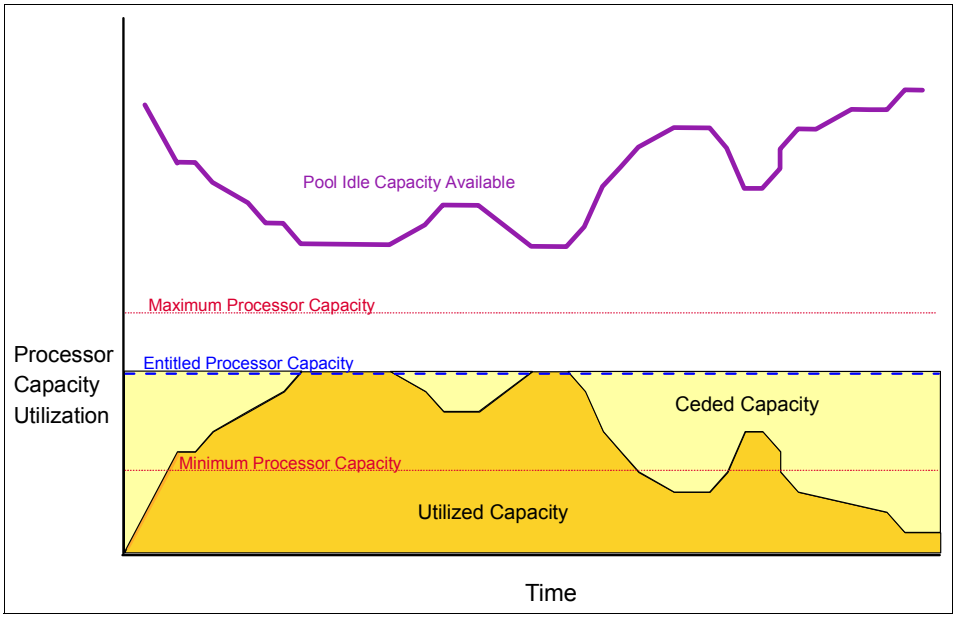

Figure 1-7 Capped shared processor partitions

Figure 1-8 on page 22 shows the usage of the shared processor pool by an uncapped partition. The uncapped partition is able to assign idle processing capacity if it needs more than the entitled capacity.

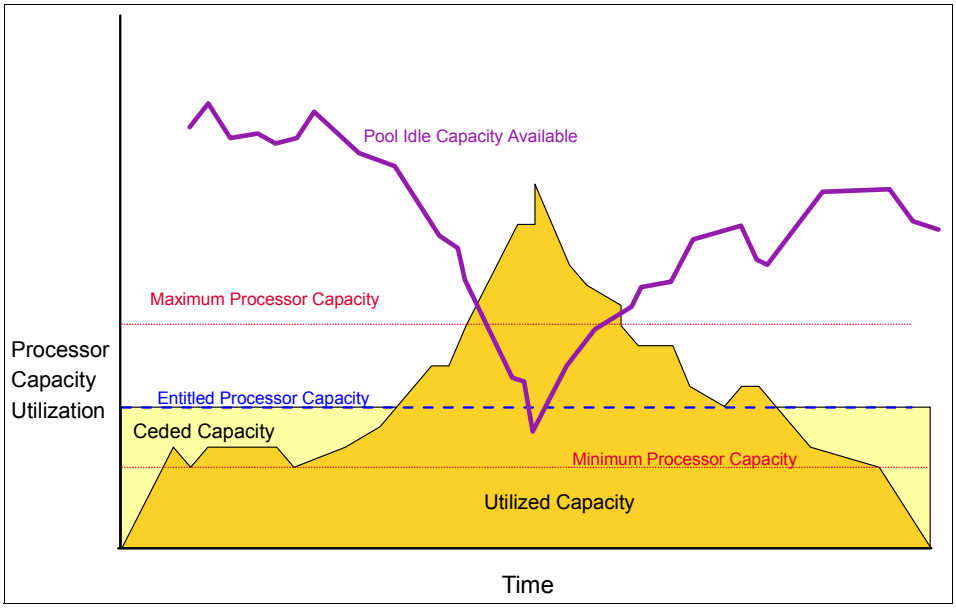

Figure 1-8 Uncapped shared processor partition

1.3.3 Dynamic partitioning

Dynamic partitioning was introduced with AIX 5L Version 5.2. An AIX 5L Version 5.2 based dynamic partition can consist of the following resource elements:

- A dedicated processor
- 256 MB memory region
- I/O adapter slot

Multiple resources can be placed under the exclusive control of a given logical partition. Dynamic LPAR extends these capabilities by allowing this fine-grained resource allocation to occur not only when activating a logical partition, but also while the partitions are running. Individual processors, memory regions, and I/O adapter slots can be released into a *free pool*, acquired from that free pool, or moved directly from one partition to another.

On POWER5 with AIX 5L Version 5.3, a partition can consist of dedicated processors, or virtual processors with a specific capacity entitlement running in capped or uncapped mode, dedicated memory region, and virtual or physical I/O adapter slots.

For dedicated and shared processor partitions it is possible to dynamically:

- Add, move, or remove memory in a granularity of 16 MB regions
- Add, move, or remove physical I/O adapter slots
- Add or remove virtual I/O adapter slots

For a dedicated processor partition it is only possible to dynamically add, move, or remove whole processors. When you dynamically remove a processor from a dedicated partition on a system that uses shared processor partitions it is then assigned to the shared processor pool.

For shared processor partitions it is also possible to dynamically:

- Remove, move, or add entitled shared processor capacity
- Change between capped and uncapped processing
- Change the weight of an uncapped partition
- Add and remove virtual processors

Figure 1-9 on page 24 shows the panel for dynamic reconfiguration of the processor resources on the HMC. Here you can choose to add, remove, or move your resources. Select the partition that you want to change dynamically and press the right mouse button. Then choose **Dynamic Logical Partitioning** from the menu, select **Processor Resources**, and choose the action you want to perform**.**

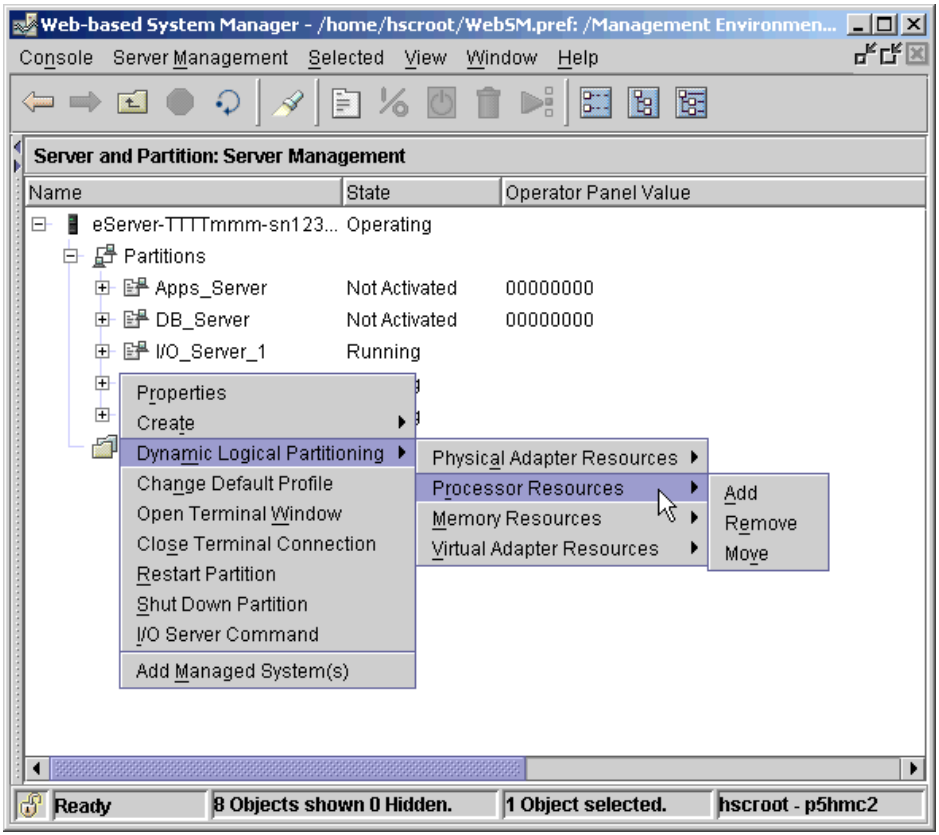

Figure 1-9 HMC panel Dynamic Logical Partitioning

When you select **Processor Resources** → **Add**, the panel shown in Figure 1-10 on page 25 is displayed. Here you can specify the processing units and the number of virtual processors you want to add to the selected partition. The limits for adding processing units and virtual processors are the maximum values defined in the partition profile. This panel also allows you to add variable weight when the partition runs in uncapped mode.

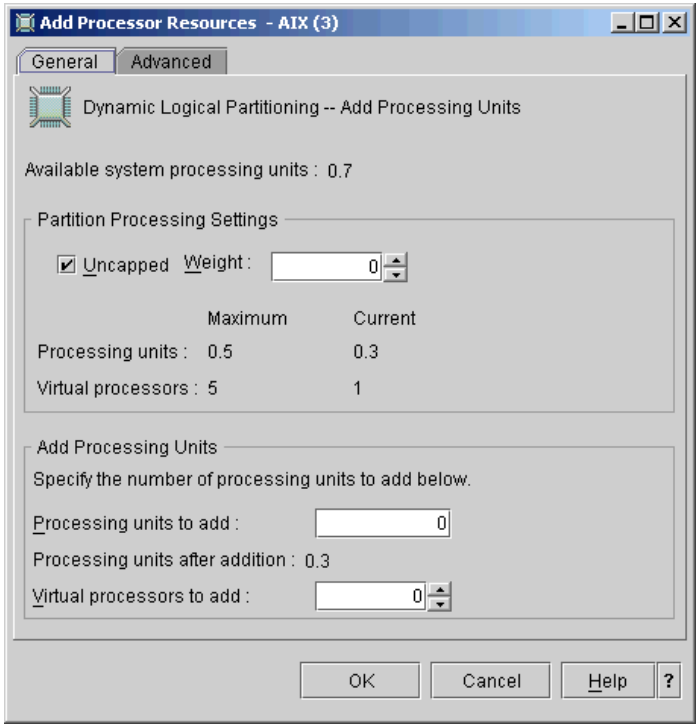

Figure 1-10 Add Processor Resource panel on the HMC

Additionally, the Add Processor Resource panel allows you to dynamically change the partition mode from uncapped to capped or vice versa. To show the actual status of the partition, use the **`lparstat -i`** command from the AIX command line interface of the partition, as shown in the following:

```
# lparstat -i
Node Name                                  : applsrv
Partition Name                             : Apps_Server
Partition Number                           : 4
Type                                       : Shared-SMT
Mode                                       : Uncapped
Entitled Capacity                          : 0.30
Partition Group-ID                         : 32772
Shared Pool ID                             : 0
Online Virtual CPUs                        : 2
Maximum Virtual CPUs                       : 10
Minimum Virtual CPUs                       : 1
Online Memory                              : 512 MB
Maximum Memory                             : 1024 MB
Minimum Memory                             : 128 MB
```

```
Variable Capacity Weight                : 128
Minimum Capacity                        : 0.20
Maximum Capacity                        : 1.00
Capacity Increment                      : 0.01
Maximum Dispatch Latency                : 16999999
Maximum Physical CPUs in system         : 2
Active Physical CPUs in system          : 2
Active CPUs in Pool                     : -
Unallocated Capacity                    : 0.00
Physical CPU Percentage                 : 15.00%
Unallocated Weight                      : 0
```

Figure 1-11 shows the way to change the mode of the partition from uncapped to capped mode. Un-check the Uncapped check box and click **OK**.

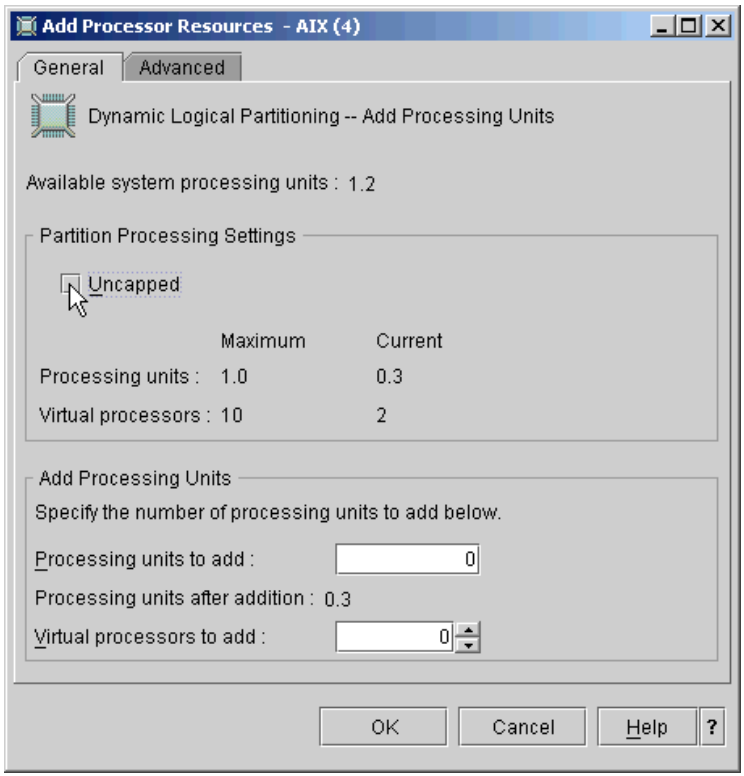

Figure 1-11 Changing the partition mode from Uncapped to Capped

To verify this dynamic action, use the **lparstat -i** command on the selected partition again. The partition mode has changed from uncapped to capped.

```
# lparstat -i
Node Name                                  : applsrv
Partition Name                             : Apps_Server
Partition Number                           : 4
Type                                       : Shared-SMT
Mode                                       : Capped
Entitled Capacity                          : 0.30
Partition Group-ID                         : 32772
Shared Pool ID                             : 0
Online Virtual CPUs                        : 2
Maximum Virtual CPUs                       : 10
Minimum Virtual CPUs                       : 1
Online Memory                              : 512 MB
Maximum Memory                             : 1024 MB
Minimum Memory                             : 128 MB
Variable Capacity Weight                   : 128
Minimum Capacity                           : 0.20
Maximum Capacity                           : 1.00
Capacity Increment                         : 0.01
Maximum Dispatch Latency                   : 16999999
Maximum Physical CPUs in system            : 2
Active Physical CPUs in system             : 2
Active CPUs in Pool                        : -
Unallocated Capacity                       : 0.00
Physical CPU Percentage                    : 15.00%
Unallocated Weight                         : 0
```

Figure 1-12 shows the Remove Processor Resources panel that allows you to dynamically remove processing units and virtual processors. The limits for the removal of processing units and virtual processors are the minimum values defined in the partition profile.

This panel also allows you to remove variable weight when the partition runs in uncapped mode.

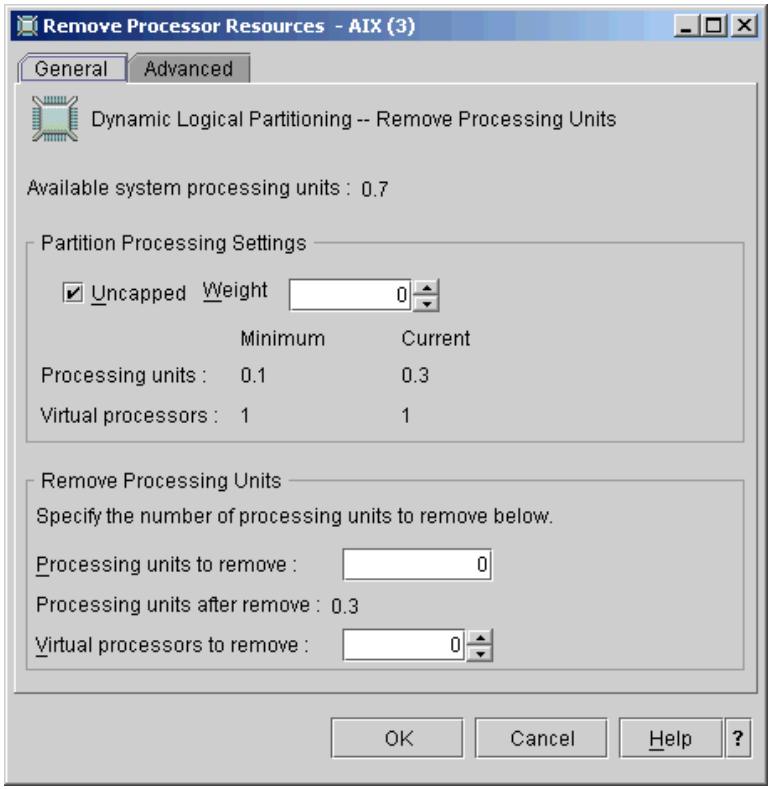

Figure 1-12 Remove Processor Resource panel on the HMC

It is also possible to change the partition mode from capped to uncapped and vice versa from the Remove Processor Resource panel.

When moving processing units you have to select the partition you want the processing units removed from and choose the Move Processor Resource Panel shown in Figure 1-13.

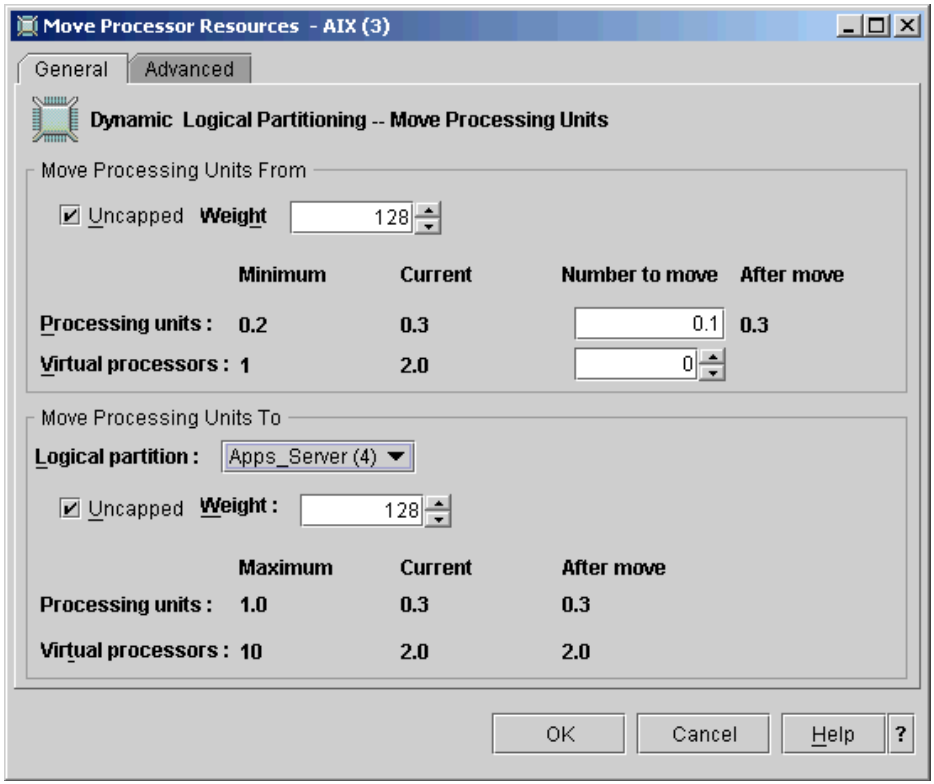

Figure 1-13 Move Processor Resources panel on HMC

In the Processing units field, select the amount of processor capacity you want to remove from the selected partition and move to the partition you select from the drop-down menu under Logical Partition. In this example, we want to move 0.1 processing units from the Web_Server partition to the Apps_Server partition.

You can also choose to move virtual processors to adjust the number of virtual processors of your partition. This does not actually move the virtual processor but removes and adds the defined number of virtual processors to the chosen partitions.

1.3.4 Limitations and considerations

The following limitations must be considered when implementing shared processor partitions:

- The limitation for a shared processor partition is 0.1 processing units of a physical processor, so the number of shared processor partitions you can create for a system depends mostly on the number of processors in a system.
- The maximum number of partitions planned is 254.
- The maximum number of virtual processors in a partition is 64.
- A mix of dedicated and shared processors within the same partition is not supported.
- If you dynamically remove a virtual processor you cannot specify a particular virtual CPU to be removed. The operating system will choose the virtual CPU to be removed.
- Shared processors may render AIX affinity management useless. AIX will continue to utilize affinity domain information as provided by firmware to build associations of virtual processors to memory, and will continue to show preference to redispatching a thread to the virtual CPU that it last ran on.

There is overhead associated with the maintenance of online virtual processors, so you should carefully consider their capacity requirements before choosing values for these attributes.

Virtual processors have dispatch latency since they are scheduled. When a virtual processor is made runnable, it is placed on a run queue by the POWER Hypervisor, where it waits until it is dispatched. The time between these two events is referred to as dispatch latency.

The dispatch latency of a virtual processor depends on the partition entitlement and the number of virtual processors that are online in the partition. The capacity entitlement is equally divided among these online virtual processors, so the number of online virtual processors impacts the length of each virtual processor's dispatch. The smaller the dispatch cycle, the greater the dispatch latency.

At the time of writing, the worst case virtual processor dispatch latency is 18 milliseconds since the minimum dispatch cycle that is supported at the virtual processor level is one millisecond. This latency is based on the minimum partition entitlement of 1/10 of a physical processor and the 10 millisecond rotation period of the Hypervisor's dispatch wheel. It can be easily visualized by imagining that a virtual processor is scheduled in the first and last portions of two 10 millisecond intervals. In general, if these latencies are too great, then clients may increase entitlement, minimize the number of online virtual processors without reducing entitlement, or use dedicated processor partitions.

In general, the value of the minimum, desired, and maximum virtual processor attributes should parallel those of the minimum, desired, and maximum capacity attributes in some fashion. A special allowance should be made for uncapped partitions, since they are allowed to consume more than their entitlement.

If the partition is uncapped, then the administrator may want to define the desired and maximum virtual processor attributes x% above the corresponding entitlement attributes. The exact percentage is installation-specific, but 25 to 50 percent is a reasonable number.

Table 1-2 shows several reasonable settings of number of virtual processors, processing units, and the capped and uncapped mode.

Table 1-2 Reasonable settings for shared processor partitions

Min VPs[a]	Desired VPs	Max VPs	Min PU[b]	Desired PU	Max. PU	Capped
1	2	4	0.1	2.0	4.0	Y
1	3 or 4	6 or 8	0.1	2.0	4.0	N
2	2	6	2.0	2.0	6.0	Y
2	3 or 4	8 or 10	2.0	2.0	6.0	N

a - Virtual processors
b - Processing units

Operating systems and applications running in shared partitions need not be aware that they are sharing processors. However, overall system performance can be significantly improved by minor operating system changes. AIX 5L Version 5.3 provides support for optimizing overall system performance of shared processor partitions.

In a shared partition there is not a fixed relationship between the virtual processor and the physical processor. The POWER Hypervisor will try to use a physical processor with the same memory affinity as the virtual processor, but it is not guaranteed. Virtual processors have the concept of a home physical processor. If it can't find a physical processor with the same memory affinity, then it gradually broadens its search to include processors with weaker memory affinity, until it finds one that it can use. As a consequence, memory affinity is expected to be weaker in shared processor partitions.

Workload variability is also expected to be increased in shared partitions because there are latencies associated with the scheduling of virtual processors and interrupts. SMT may also increase variability, since it adds another level of

resource sharing, which could lead to a situation where one thread interferes with the forward progress of its sibling.

Therefore, if an application is cache-sensitive or cannot tolerate variability, then it should be deployed in a dedicated partition with SMT disabled. In dedicated partitions, the entire processor is assigned to a partition. Processors are not shared with other partitions, and they are not scheduled by the POWER Hypervisor. Dedicated partitions must be explicitly created by the system administrator using the Hardware Management Console.

Processor and memory affinity data is only provided in dedicated partitions. In a shared processor partition, all processors are considered to have the same affinity. Affinity information is provided through RSET APIs, which contain discovery and bind services.

1.4 Virtual Ethernet introduction

Virtual Ethernet enables inter-partition communication without the need for physical network adapters assigned to each partition. Virtual Ethernet allows the administrator to define in-memory point-to-point connections between partitions. These connections exhibit characteristics similar to physical high-bandwidth Ethernet connections and support multiple protocols (IPv4, IPv6, ICMP). Virtual Ethernet requires a p5 system with either AIX 5L Version 5.3 or the appropriate level of Linux and an HMC to define the Virtual Ethernet devices. Virtual Ethernet does not require the purchase of any additional features or software such as the Advanced POWER Virtualization Feature.

The concepts of implementing Virtual Ethernet on p5 systems are discussed in the following sections:

- Virtual LAN
- Virtual Ethernet connections
- Benefits of Virtual Ethernet
- Limitations

1.4.1 Virtual LAN

This section discusses the concepts of Virtual LAN (VLAN) technology with specific reference to its implementation within AIX.

Virtual LAN overview

Virtual LAN is a technology used for establishing virtual network segments on top of physical switch devices. If configured appropriately, a VLAN definition can

straddle multiple switches. Typically, a VLAN is a broadcast domain that enables all nodes in the VLAN to communicate with each other without any L3 routing or inter-VLAN bridging. In Figure 1-14, two VLANs (VLAN 1 and 2) are defined on three switches (Switch A, B, and C). Although nodes C-1 and C-2 are physically connected to the same switch C, traffic between two nodes can be blocked. To enable communication between VLAN 1 and 2, L3 routing or inter-VLAN bridging should be established between them; this is typically provided by an L3 device.

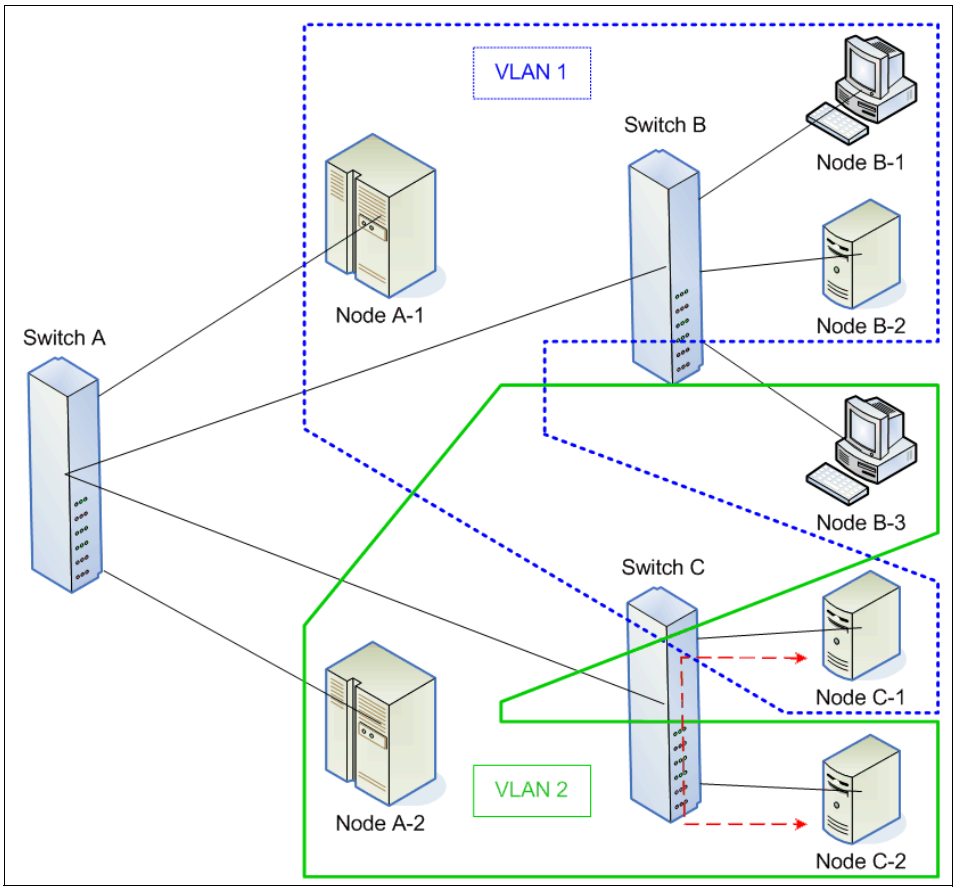

Figure 1-14 Example of a VLAN

The use of VLAN provides increased LAN security and flexible network deployment over traditional network devices.

AIX virtual LAN support

Some of the technologies for implementing VLANs include:

- Port-based VLAN
- Layer 2 VLAN
- Policy-based VLAN
- IEEE 802.1Q VLAN

VLAN support in AIX is based on the IEEE 802.1Q VLAN implementation. The IEEE 802.1Q VLAN is achieved by adding a VLAN ID tag to an Ethernet frame, and the Ethernet switches restricting the frames to ports that are authorized to receive frames with that VLAN ID. Switches also restrict broadcasts to the logical network by ensuring that a broadcast packet is delivered to all ports which are configured to receive frames with the VLAN ID that the broadcast frame was tagged with.

A port on a VLAN-capable switch has a default PVID (Port virtual LAN ID) that indicates the default VLAN the port belongs to. The switch adds the PVID tag to untagged packets that are received by that port. In addition to a PVID, a port may belong to additional VLANs and have those VLAN IDs assigned to it that indicate the additional VLANs the port belongs to.

A port will only accept untagged packets or packets with a VLAN ID (PVID or additional VIDs) tag of the VLANs the port belongs to. A port configured in the untagged mode is only allowed to have a PVID and will receive untagged packets or packets tagged with the PVID. The untagged port feature helps systems that do not understand VLAN tagging communicate with other systems using standard Ethernet.

Each VLAN ID is associated with a separate Ethernet interface to the upper layers (for example IP) and creates unique logical Ethernet adapter instances per VLAN (for example ent1 or ent2).

You can configure multiple VLAN logical devices on a single system. Each VLAN logical device constitutes an additional Ethernet adapter instance. These logical devices can be used to configure the same Ethernet IP interfaces as are used with physical Ethernet adapters.

VLAN communication by example

This section discusses how VLAN communication between partitions and with external networks works in more detail, using the sample configuration in Figure 1-15 on page 35. The configuration is using four client partitions (Partition 1 through Partition 4) and one Virtual I/O Server. Each of the client partitions is defined with one Virtual Ethernet adapter. The Virtual I/O Server has a Shared

Ethernet Adapter which bridges traffic to the external network. The Shared Ethernet Adapter is discussed in more detail in 1.5, “Shared Ethernet Adapter” on page 39.

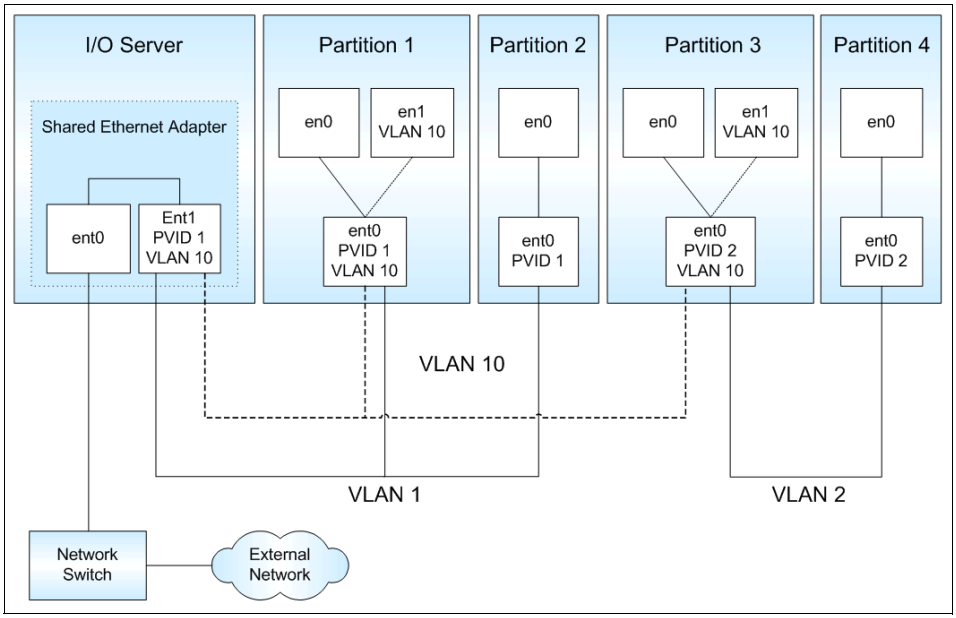

Figure 1-15 VLAN configuration

Interpartition communication

Partition 2 and Partition 4 are using the PVID (Port virtual LAN ID) only. This means that:

- Only packets for the VLAN specified as PVID are received.
- Packets sent have a VLAN tag added for the VLAN specified as PVID by the Virtual Ethernet adapter.

In addition to the PVID, the Virtual Ethernet adapters in Partition 1 and Partition 3 are also configured for VLAN 10 using a specific network interface (en1) created through **`smitty vlan`**. This means that:

- Packets sent through network interfaces en1 are added a tag for VLAN 10 by the network interface in AIX.
- Only packets for VLAN 10 are received by the network interfaces en1.
- Packets sent through en0 are automatically tagged for the VLAN specified as PVID.
- Only packets for the VLAN specified as PVID are received by the network interfaces en0.

Table 1-3 lists which *client* partitions can communicate which each other through what network interfaces.

Table 1-3 Interpartition VLAN communication

VLAN	Partition / Network interface
1	Partition 1 / en0 Partition 2 / en0
2	Partition 3 / en0 Partition 4 / en0
10	Partition 1 / en1 Partition 3 / en1

Communication with external networks

The Shared Ethernet Adapter is configured with PVID 1 and VLAN 10. This means that untagged packets that are received by the Shared Ethernet Adapter are tagged for VLAN 1. Handling of outgoing traffic depends on the VLAN tag of the outgoing packets.

- Packets tagged with the VLAN which matches the PVID of the Shared Ethernet Adapter are untagged before being sent out to the external network.
- Packets tagged with a VLAN *other than* the PVID of the Shared Ethernet Adapter are sent out with the VLAN tag unmodified.

In our example, Partition 1 and Partition 2 have access to the external network through network interface en0 using VLAN 1. Since these packets are using the PVID, the Shared Ethernet Adapter will remove the VLAN tags before sending the packets to the external network.

Partition 1 and Partition 3 have access to the external network using network interface en1 and VLAN 10. These packets are sent out by the Shared Ethernet Adapter with the VLAN tag. Therefore, only VLAN-capable destination devices will be able to receive the packets. Table 1-4 lists this relationship.

Table 1-4 VLAN communication to external network

VLAN	Partition / Network interface
1	Partition 1 / en0 Partition 2 / en0
10	Partition 1 / en1 Partition 3 / en1

1.4.2 Virtual Ethernet connections

Virtual Ethernet connections supported in POWER5 systems use VLAN technology to ensure that the partitions can only access data directed to them. The POWER Hypervisor provides a Virtual Ethernet switch function based on the IEEE 802.1Q VLAN standard that allows partition communication within the same server. The connections are based on an implementation internal to the hypervisor that moves data between partitions. This section describes the various elements of a Virtual Ethernet and implications relevant to different types of workloads. Figure 1-16 is an example of an inter-partition VLAN.

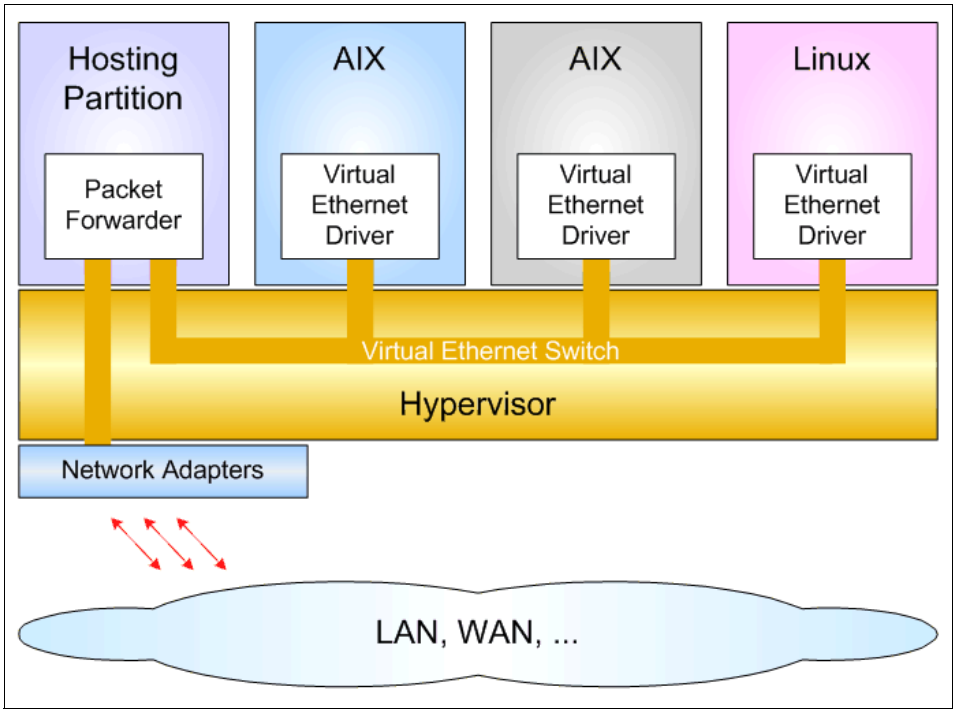

Figure 1-16 Logical view of an inter-partition VLAN

Virtual Ethernet adapter concepts

Partitions that communicate through a Virtual Ethernet channel will need to have an additional in-memory channel. This requires the creation of an in-memory channel between partitions on the HMC. The kernel creates a virtual device for each memory channel indicated by the firmware. The AIX configuration manager creates the device special files. A unique Media Access Control (MAC) address is also generated when the Virtual Ethernet device is created. A *prefix* value can be assigned for the system so that the generated MAC addresses in a system

consist of a common system prefix, plus an algorithmically-generated unique part per adapter.

The Virtual Ethernet can also be used as a bootable device to allow such tasks as operating system installations to be performed using NIM.

Performance considerations

The transmission speed of Virtual Ethernet adapters is in the range of 1-3 Gigabits per second, depending on the transmission (MTU) size. A partition can support up to 256 Virtual Ethernet adapters with each Virtual Ethernet capable of being associated with up to 21 VLANs (20 VID and 1 PVID).

The Virtual Ethernet connections generally take up more processor time than a local adapter to move a packet (DMA versus copy). For shared processor partitions, performance will be gated by the partition definitions (for example, entitled capacity and number of processors). Small partitions communicating with each other will experience more packet latency due to partition context switching. In general, high bandwidth applications should *not* be deployed in small shared processor partitions. For dedicated partitions, throughput should be comparable to a 1 Gigabit Ethernet for small packets providing much better performance than 1 Gigabit Ethernet for large packets. For large packets, the Virtual Ethernet communication is copy bandwidth limited.

1.4.3 Benefits of Virtual Ethernet

Because the number of partitions possible on many systems is greater than the number of I/O slots, Virtual Ethernet is a convenient and cost saving option to enable partitions within a single system to communicate with one another through a VLAN. The VLAN creates logical Ethernet connections between one or more partitions and is designed to help prevent a failed or malfunctioning operating system from being able to impact the communication between two functioning operating systems. The Virtual Ethernet connections may also be *bridged* to an external network to permit partitions without physical network adapters to communicate outside of the server.

1.4.4 Dynamic partitioning for Virtual Ethernet devices

Virtual Ethernet resources can be assigned and removed dynamically. On the HMC, Virtual Ethernet target and server adapters can be assigned and removed from a partition using dynamic logical partitioning. The mapping between physical and virtual resources on the Virtual I/O Server can also be done dynamically.

1.4.5 Limitations and considerations

The following are limitations that must be considered when implementing a Virtual Ethernet:

- A maximum of up to 256 Virtual Ethernet adapters are permitted per partition.
- Virtual Ethernet can be used in both shared and dedicated processor partitions provided the partition is running AIX 5L Version 5.3 or Linux with the 2.6 kernel or a kernel that supports virtualization.
- A mixture of Virtual Ethernet connections, real network adapters, or both are permitted within a partition.
- Virtual Ethernet can only connect partitions within a single system.
- Virtual Ethernet requires a POWER5 system and an HMC to define the Virtual Ethernet adapters.
- Virtual Ethernet connections from AIX or Linux partitions to an i5/OS™ partition may work; however, at the time of writing, these capabilities were unsupported.
- Virtual Ethernet uses the system processors for all communication functions instead of offloading that load to processors on network adapter cards. As a result, there is an increase in system processor load generated by the use of Virtual Ethernet.

1.5 Shared Ethernet Adapter

A Shared Ethernet Adapter can be used to connect a physical Ethernet to the Virtual Ethernet. It also provides the possibility for several client partitions to share one physical adapter.

The following sections discuss the various aspects of Shared Ethernet Adapters such as:

- Connecting Virtual Ethernet to external networks
- Ethernet adapter sharing
- Benefits of Shared Ethernet Adapters
- Using Link Aggregation (EtherChannel) for external network interface
- Limitations and considerations

1.5.1 Connecting a Virtual Ethernet to external networks

There are two ways you can connect the Virtual Ethernet that enables the communication between logical partitions on the same server to an external network.

Routing

By enabling the AIX routing capabilities (ipforwarding network option) one partition with a physical Ethernet adapter connected to an external network can act as *router*. Figure 1-17 shows a sample configuration. In this type of configuration the partition that routes the traffic to the external work does not necessarily have to be the Virtual I/O Server as in the pictured example. It could be any partition with a connection to the outside world. The client partitions would have their default route set to the partition which routes traffic to the external network.

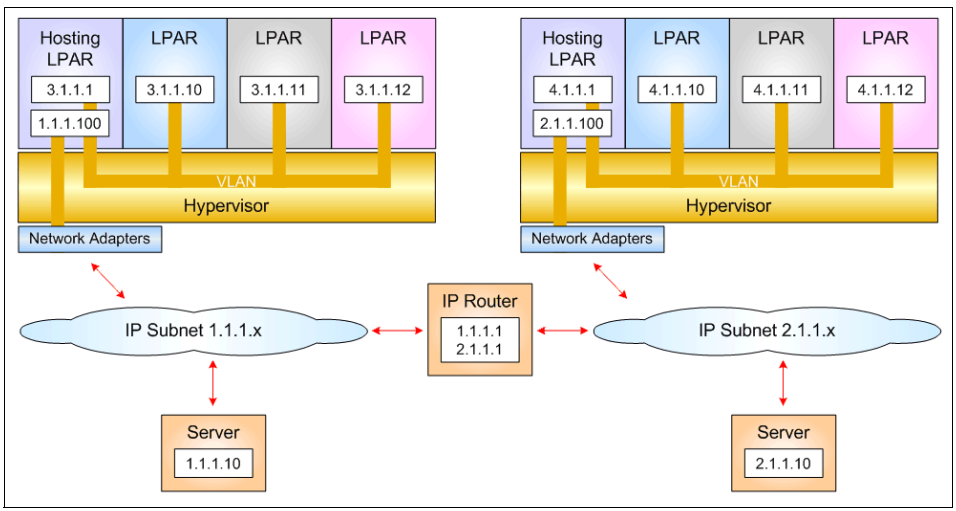

Figure 1-17 Connection to external network using AIX routing

Shared Ethernet Adapter

Using a Shared Ethernet Adapter (SEA) you can connect internal and external VLANs using one physical adapter. The Shared Ethernet Adapter hosted in the Virtual I/O Server acts as a layer 2 switch between the internal and external network.

Shared Ethernet Adapter is a new service that acts as a layer 2 network bridge to securely transport network traffic from a Virtual Ethernet to a real network adapter. The Shared Ethernet Adapter service runs in the Virtual I/O Server. It cannot be run in a general purpose AIX partition.

Shared Ethernet Adapter requires the POWER Hypervisor component of POWER5 systems and therefore cannot be used on POWER4 systems. It also cannot be used with AIX 5L Version 5.2 because the device drivers for Virtual Ethernet are only available for AIX 5L Version 5.3 and Linux. Thus there is no way to connect an AIX 5L Version 5.2 system to a Shared Ethernet Adapter.

The Shared Ethernet Adapter allows partitions to communicate outside the system without having to dedicate a physical I/O slot and a physical network adapter to a client partition. The Shared Ethernet Adapter has the following characteristics:

- Virtual Ethernet MAC addresses are visible to outside systems.
- Broadcast/multicast is supported.
- ARP and NDP can work across a shared Ethernet.

In order to bridge network traffic between the Virtual Ethernet and external networks, the Virtual I/O Server has to be configured with at least one physical Ethernet adapter. One Shared Ethernet Adapter can be shared by multiple VLANs, and multiple subnets can connect using a single adapter on the Virtual I/O Server. Figure 1-18 shows a configuration example. A Shared Ethernet Adapter can include up to 16 Virtual Ethernet adapters that share the physical access.

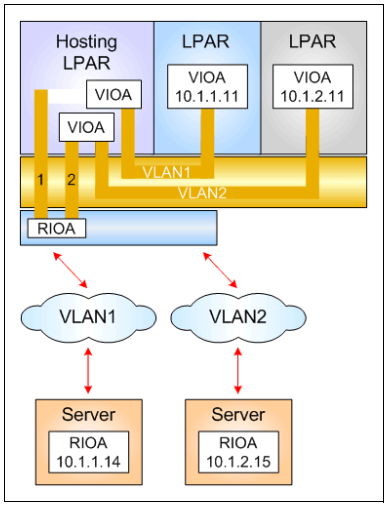

Figure 1-18 Shared Ethernet Adapter configuration

A Virtual Ethernet adapter connected to the Shared Ethernet Adapter must have the trunk flag set. Once an Ethernet frame is sent from the Virtual Ethernet adapter on a client partition to the POWER Hypervisor, the POWER Hypervisor searches for the destination MAC address within the VLAN. If no such MAC

address exists within the VLAN, it forwards the frame to the trunk Virtual Ethernet adapter that is defined on the same VLAN. The trunk Virtual Ethernet adapter enables a layer 2 bridge to a physical adapter.

The shared Ethernet directs packets based on the VLAN ID tags. It learns this information based on observing the packets originating from the virtual adapters. One of the virtual adapters in the Shared Ethernet adapter is designated as the default PVID adapter. Ethernet frames without any VLAN ID tags are directed to this adapter and assigned the default PVID.

When the shared Ethernet receives IP (or IPv6) packets that are larger than the MTU of the adapter that the packet is forwarded through, either IP fragmentation is performed and the fragments forwarded or an ICMP packet too big message is returned to the source when the packet cannot be fragmented.

Theoretically, one adapter can act as the only contact with external networks for all client partitions. Depending on the number of client partitions and the network load they produce, performance can become a critical issue. Because the Shared Ethernet Adapter is dependant on Virtual I/O, it consumes processor time for all communications. A significant amount of CPU load can be generated by the use of Virtual Ethernet and Shared Ethernet Adapter.

There are several different ways to configure physical and virtual Ethernet adapters into Shared Ethernet Adapters to maximize throughput.

- Using Link Aggregation (EtherChannel), several physical network adapters can be aggregated. See 1.5.2, “Using link aggregation (EtherChannel) to external networks” on page 43 for more details.
- Using several Shared Ethernet Adapters provides more queues and more performance. An example for this configuration is shown in Figure 1-19 on page 43.

Other aspects which have to be taken into consideration are availability and the possibility to connect to different networks.

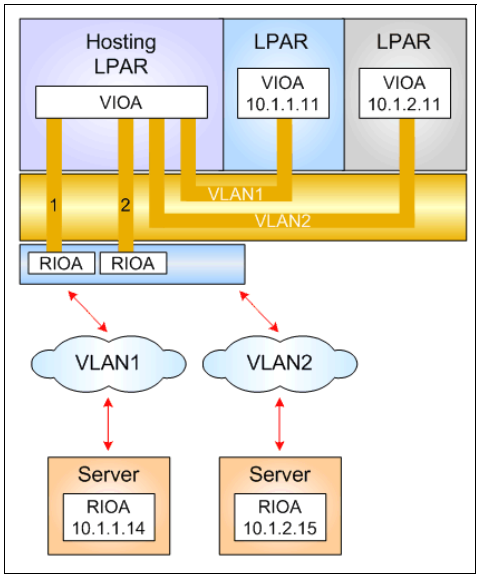

Figure 1-19 Multiple Shared Ethernet Adapter configuration

1.5.2 Using link aggregation (EtherChannel) to external networks

Link aggregation is network port aggregation technology that allows several Ethernet adapters to be aggregated together to form a single pseudo-Ethernet device. This technology can be used to overcome the bandwidth limitation of a single network adapter and avoid bottlenecks when sharing one network adapter among many client partitions.

For example, ent0 and ent1 can be aggregated to ent3. Interface en3 would then be configured with an IP address. The system considers these aggregated adapters as one adapter. Therefore, IP is configured as on any other Ethernet adapter. In addition, all adapters in the link aggregation are given the same hardware (MAC) address, so they are treated by remote systems as though they were one adapter. The main benefit of link aggregation is that the network bandwidth of all of the adapters is in a single network presence. If an adapter fails, the packets are automatically sent on the next available adapter without disruption to existing user connections. The adapter is automatically returned to service on the link aggregation when it recovers.

You can use EtherChannel (EC) or IEEE 802.3ad Link Aggregation (LA) to aggregate network adapters. While EC is an AIX-specific implementation of adapter aggregation, LA follows the IEEE 802.3ad standard. Table 1-5 on page 44 shows the main differences between EC and LA.

Table 1-5 Main differences between EC and LA aggregation

EtherChannel	IEEE 802.3ad Link Aggregation
Requires switch configuration.	Little, if any, configuration of switch required to form aggregation. Some initial setup of the switch may be required.
Supports different packet distribution modes.	Supports only standard distribution mode.

The main benefit of using LA is, that if the switch supports the *Link Aggregation Control Protocol* (LACP) no special configuration of the switch ports is required. The benefit of EC is the support of different packet distribution modes. This means it is possible to influence the load balancing of the aggregated adapters. In the remainder of this document, we use Link Aggregation where possible since that is considered a more universally understood term.

Note: Only outgoing packets are subject to the following discussion; incoming packets are distributed by the Ethernet switch.

Standard distribution mode selects the adapter for the outgoing packets by algorithm. The adapter selection algorithm uses the last byte of the destination IP address (for TCP/IP traffic) or MAC address (for ARP and other non-IP traffic). Therefore all packets to a specific IP address will always go through the same adapter. There are other adapter selection algorithms based on source, destination, or a combination of source and destination ports available. EC provides one further distribution mode called *round robin*. This mode will rotate through the adapters, giving each adapter one packet before repeating. The packets may be sent out in a slightly different order than they were given to the EC. It will make the best use of its bandwidth, but consider that it also introduces the potential for out-of-order packets at the receiving system. This risk is particularly high when there are few, long-lived, streaming TCP connections. When there are many such connections between a host pair, packets from different connections could be intermingled, thereby decreasing the chance of packets for the same connection arriving out-of-order.

To avoid the loss of network connectivity by switch failure, EC and LA can provide a backup adapter. The backup adapter should be connected to a different switch than the adapter of the aggregation. Now in case of switch failure the traffic can be moved with no disruption of user connections to the backup adapter.

Figure 1-20 on page 45 shows the aggregation of three plus one adapters to a single pseudo-Ethernet device including a backup feature.

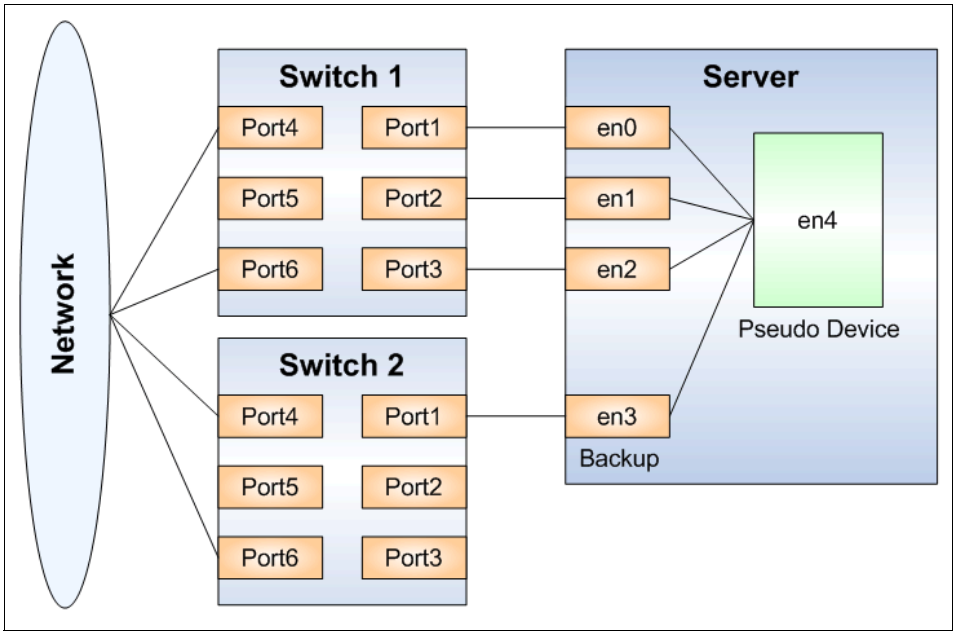

Figure 1-20 Link Aggregation (EtherChannel) pseudo device

1.5.3 Limitations and considerations

You must consider the following limitations when implementing Shared Ethernet Adapters in the Virtual I/O Server:

- Because Shared Ethernet Adapter depends on Virtual Ethernet, which uses the system processors for all communications functions, a significant amount of system processor load can be generated by the use of Virtual Ethernet and Shared Ethernet Adapter.
- One of the virtual adapters in the Shared Ethernet Adapter on the Virtual I/O Server must be defined as the default adapter with a default PVID. This virtual adapter is designated as the PVID adapter and Ethernet frames without any VLAN ID tags are assigned the default PVID and directed to this adapter.
- Up to 16 Virtual Ethernet adapters with 21 VLANs (20 VID and 1 PVID) on each can be shared on a single physical network adapter. There is no limit on the number of partitions that can attach to a VLAN, so the theoretical limit is very high. In practice, the amount of network traffic will limit the number of clients that can be served through a single adapter.

For performance and latency related information refer to *Advanced POWER Virtualization on IBM ℮server p5 Servers Architecture and Performance Considerations*, SG24-5768.

1.6 Virtual SCSI introduction

Available at the time of writing is the first support of Virtual I/O, which pertains to a virtualized implementation of the SCSI protocol.

Virtual SCSI requires POWER5 hardware with the Advanced POWER Virtualization feature activated. It provides Virtual SCSI support for AIX 5L Version 5.3 and Linux.

The driving forces behind virtual I/O are:

- The advanced technological capabilities of today's hardware and operating systems like POWER5 and IBM AIX 5L Version 5.3.
- The value proposition enabling on demand computing and server consolidation. Virtual I/O also provides a more economical I/O model by using physical resources more efficiently through sharing.

At the time of writing, the virtualization features of the POWER5 platform support up to 254 partitions, while the server hardware only provides up to 160 I/O slots per machine. With each partition typically requiring one I/O slot for disk attachment and another one for network attachment, this puts a constraint on the number of partitions. To overcome these physical limitations, I/O resources have to be shared. Virtual SCSI provides the means to do this for SCSI storage devices.

Furthermore, virtual I/O allows attachment of previously unsupported storage solutions. As long as the Virtual I/O Server supports the attachment of a storage resource, any client partition can access this storage by using Virtual SCSI adapters.

For example, if there is no native support for EMC storage devices on Linux, running Linux in a logical partition of a POWER5 server makes this possible.

A Linux client partition can access the EMC storage through a Virtual SCSI adapter. Requests from the virtual adapters are mapped to the physical resources in the Virtual I/O Server. Driver support for the physical resources is therefore only needed in the Virtual I/O Server.

Note: You will see different terms in this publication that refer to the various components involved with virtual SCSI. These terms vary depending on the context. With SCSI, usually the terms *initiator* and *target* are used, so you may see terms such as *virtual SCSI initiator* and *virtual SCSI target*. On the HMC, the terms *virtual SCSI server adapter* and *virtual SCSI client adapter* are used. Basically they refer to the same thing. When describing the client/server relationship between the partitions involved in virtual SCSI, the terms *hosting partition* (meaning the Virtual I/O Server) and *hosted partition* (meaning the client partition) are used.

The terms *Virtual I/O Server partition* and *Virtual I/O Server* both refer to the Virtual I/O Server. The terms are used interchangeably in this section.

1.6.1 Partition access to virtual SCSI devices

The following sections describe the virtual SCSI architecture and the protocols used.

Virtual SCSI client and server architecture overview

Virtual SCSI is based on a client/server relationship. The Virtual I/O Server owns the physical resources and acts as the server or, in SCSI terms, target device. The logical partitions access the virtual SCSI resources provided by the Virtual I/O Server as clients.

The virtual I/O adapters are configured using an HMC. The provisioning of virtual disk resources is provided by the Virtual I/O Server.

Often the Virtual I/O Server is also referred to as the hosting partition and the client partitions as hosted partitions.

Physical disks owned by the Virtual I/O Server can either be exported and assigned to a client partition whole, or can be partitioned into several logical volumes. The logical volumes can then be assigned to different partitions. Therefore, Virtual SCSI enables sharing of adapters as well as disk devices.

To make a physical or a logical volume available to a client partition it is assigned to a virtual SCSI server adapter in the Virtual I/O Server.

The client partition accesses its assigned disks through a virtual SCSI client adapter. The virtual SCSI client adapter sees standard SCSI devices and LUNs through this virtual adapter. The commands in the following example show how the disks appear on an AIX client partition.

```
# lsdev -Cc disk -s vscsi
hdisk2 Available  Virtual SCSI Disk Drive
```

```
# lscfg -vpl hdisk2
hdisk2 111.520.10DDEDC-V3-C5-T1-L810000000000 Virtual SCSI Disk Drive
```

Figure 1-21 shows an example where one physical disk is partitioned into two logical volumes inside the Virtual I/O Server. Each of the two client partitions is assigned one logical volume which it accesses through a virtual I/O adapter (vSCSI Client Adapter). Inside the partition the disk is seen as normal hdisk.

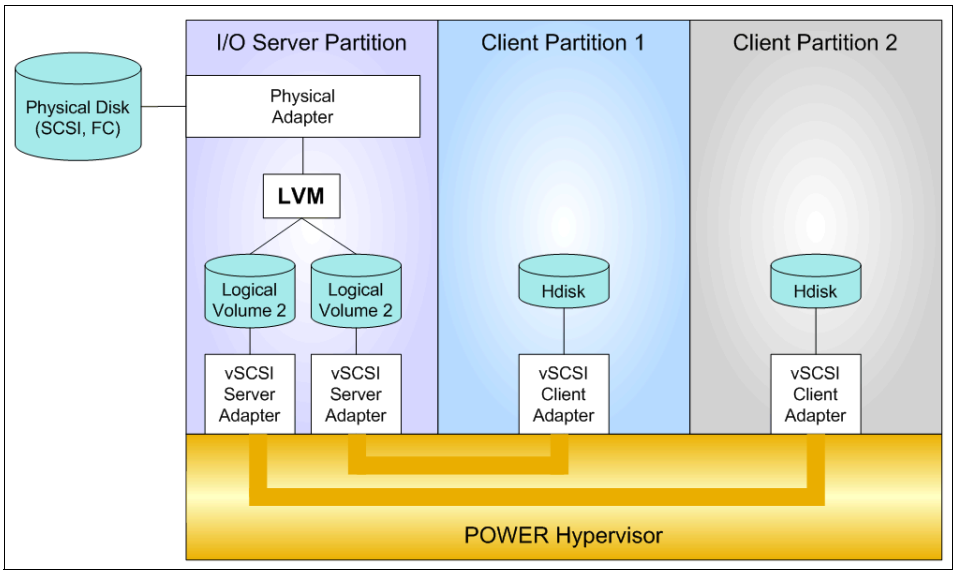

Figure 1-21 Virtual SCSI architecture overview

SCSI Remote Direct Memory Access

The SCSI family of standards provides many different transport protocols that define the rules for exchanging information between SCSI initiators and targets. Virtual SCSI uses the SCSI RDMA Protocol (SRP) which defines the rules for exchanging SCSI information in an environment where the SCSI initiators and targets have the ability to directly transfer information between their respective address spaces.

SCSI requests and responses are sent using the virtual SCSI adapters that communicate through the POWER Hypervisor.

The actual data transfer, however, is done directly between a data buffer in the client partition and the physical adapter in the Virtual I/O Server using the Logical Remote Direct Memory Access (LRDMA) protocol.

Figure 1-22 shows how the data transfer using LRDMA appears.

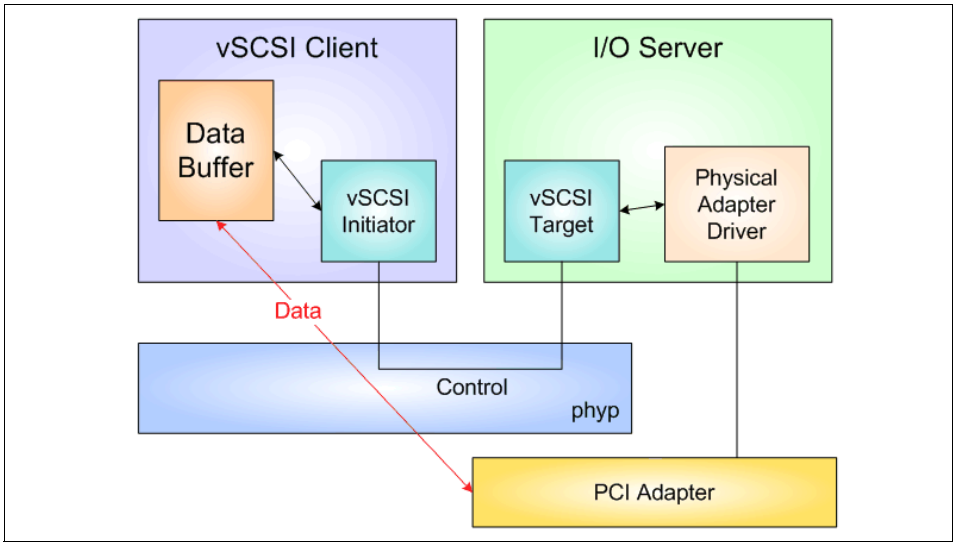

Figure 1-22 Logical Remote Direct Memory Access

AIX device configuration for virtual SCSI

The virtual I/O adapters are connected to a virtual host bridge which AIX treats much like a PCI host bridge. It is represented in the ODM as a bus device whose parent is sysplanar0. The virtual I/O adapters are represented as adapter devices with the virtual host bridge as their parent.

On the Virtual I/O Server, each logical volume or physical volume that is exported to a client partition is represented by a virtual target device that is a child of a Virtual SCSI server adapter.

On the client partition, the exported disks are visible as normal hdisks, but they are defined in subclass vscsi. They have a virtual SCSI client adapter as parent.

Figure 1-23 and Figure 1-24 on page 50 show the relationship of the devices used by AIX for virtual SCSI and their physical counterparts.

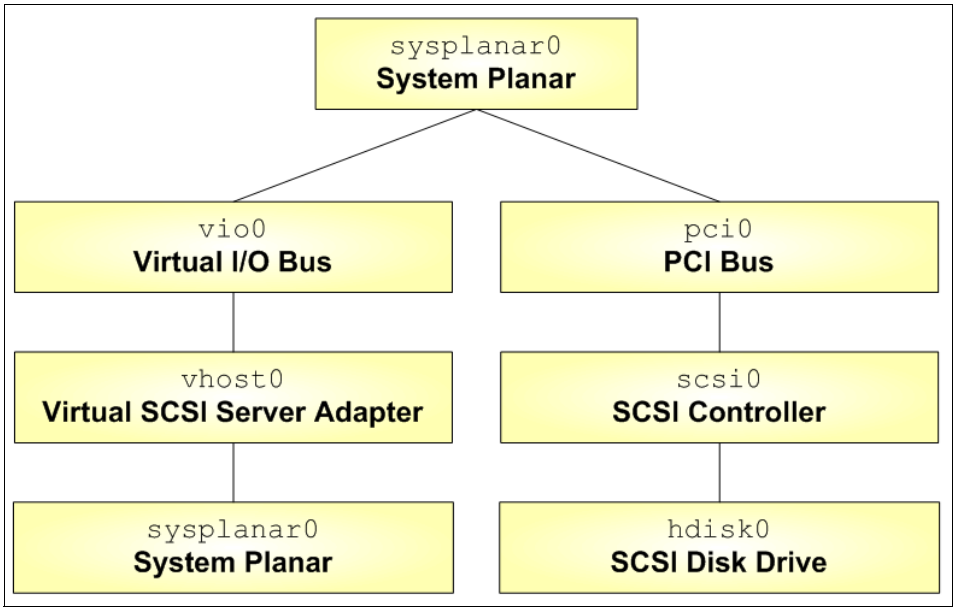

Figure 1-23 Virtual SCSI device relationship on Virtual I/O Server

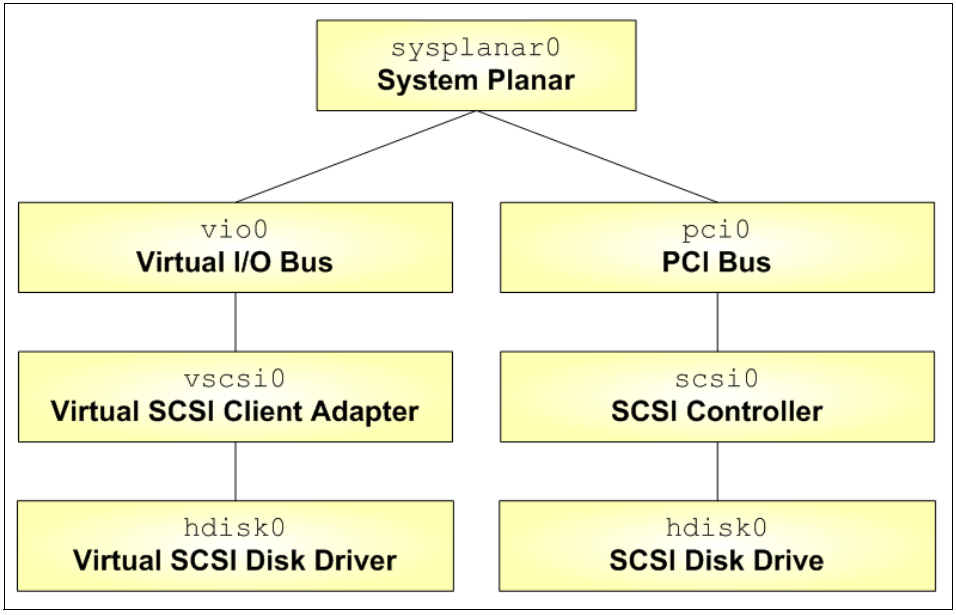

Figure 1-24 Virtual SCSI device relationship on AIX client partition

Dynamic partitioning for virtual SCSI devices

Virtual SCSI resources can be assigned and removed dynamically. On the HMC, Virtual SCSI target and server adapters can be assigned and removed from a partition using dynamic logical partitioning.

The mapping between physical and virtual resources on the Virtual I/O Server can also be done dynamically.

1.6.2 Limitations and considerations

The following areas should be considered when implementing virtual SCSI:

- At the time of writing virtual SCSI supports Fibre Channel, parallel SCSI, and SCSI RAID devices. Other protocols such as SSA or tape and CD-ROM devices are not supported.
- Virtual SCSI itself does not have any limitations in terms of number of supported devices or adapters. However, the Virtual I/O Server supports a maximum of 65535 virtual I/O slots. A maximum of 256 virtual I/O slots can be assigned to a single partition.

  Every I/O slot needs some resources to be instantiated. Therefore, the size of the Virtual I/O Server puts a limit to the number of virtual adapters that can be configured. For details see *Advanced POWER Virtualization on IBM @server p5 Servers Architecture and Performance Considerations*, SG24-5768.
- The SCSI protocol defines mandatory and optional commands. While virtual SCSI supports all the mandatory commands, not all optional commands are supported.
- There are performance implications when using virtual SCSI devices. It is important to understand that, due to the overhead associated with POWER Hypervisor calls, virtual SCSI will use additional CPU cycles when processing I/O requests. When putting heavy I/O load on virtual SCSI devices, this means you will use considerably more CPU cycles. Provided that there is sufficient CPU processing capacity available the performance of virtual SCSI should be comparable to dedicated I/O devices.

  Suitable applications for virtual SCSI include boot disks for the operating system or Web servers which will typically cache a lot of data. When designing a virtual I/O configuration, performance is an important aspect which should be given careful consideration. For a more in-depth discussion of performance issues see *Advanced POWER Virtualization on IBM @server p5 Servers Architecture and Performance Considerations*, SG24-5768.

1.7 Partition Load Manager introduction

The Partition Load Manager (PLM) software is part of the Advanced POWER Virtualization feature and helps customers maximize the utilization of processor and memory resources of DLPAR-capable logical partitions running AIX 5L on pSeries servers.

The Partition Load Manager is a resource manager that assigns and moves resources based on defined policies and utilization of the resources. PLM manages memory, both dedicated processors and partitions using Micro-Partitioning technology, to readjust the resources. This adds additional flexibility on top of the micro-partitions flexibility added by the POWER Hypervisor.

PLM, however, has no knowledge about the importance of a workload running in the partitions and cannot readjust priority based on the changes in types of workloads. PLM does not manage Linux and i5/OS partitions. Figure 1-25 shows a comparison of features between PLM and the POWER Hypervisor.

PLM Differentiation	Capability	PLM	P5 PHYP
HW Support	**POWER4** PLM automates DLPAR adjustment for P4 install base	X	
	POWER5	X	X
OS Support	**AIX 5.2** PLM runs on AIX 5.2 on P4 and P5 systems (through PRPQ)	X	
	AIX 5.3	X	X
	pLinux		X
Physical Processor Management	**Dedicated** PLM runs on AIX 5.2 and/or P4 systems	X	
	Capped shared	X	
	Uncapped shared	X	X
Virtual Processor Management	**Virtual processor minimization for efficiency**	X	
	Virtual processor adjustment for physical processor growth	X	
Physical Memory Management	**Share-based**	X	
	Minimum and maximum entitlements	X	
Management Policy	**Entitlement-based**	X	X
	Goal-based		
	Application/middleware instrumentation required		
Management Domains	**Multiple management domains on a single CEC**	X	
	Cross platform (CEC)		
Administration	**Simple administration**	X	X
	Centralized LPAR monitoring (PLM command provides usage stats)	X	
	TOD-driven policy adjustment (PLM command supports new policy load based as TOD)	X	

Figure 1-25 Comparison of features of PLM and POWER Hypervisor

Partition Load Manager is set up in a partition or on another system running AIX 5L Version 5.2 ML4 or AIX 5L Version 5.3. Linux or i5OS support for PLM and the clients is not available. You can have other installed applications on the partition or system running the Partition Load Manager as well. A single instance of the Partition Load Manager can only manage a single server.

You can use the command line interface to configure Partition Load Manager, or the Web-based System Manager for graphical setup.

Partition Load Manager uses a client/server model to report and manage resource utilization. The clients (managed partitions) notify the PLM server when resources are either under- or over-utilized. Upon notification of one of these events, the PLM server makes resource allocation decisions based on a policy file defined by the administrator.

Partition Load Manager uses the Resource Monitoring and Control (RMC) subsystem for network communication, which provides a robust and stable framework for monitoring and managing resources. Communication with the HMC to gather system information and execute commands PLM requires a configured SSH connection.

Figure 1-26 shows an overview of the components of Partition Load Manager.

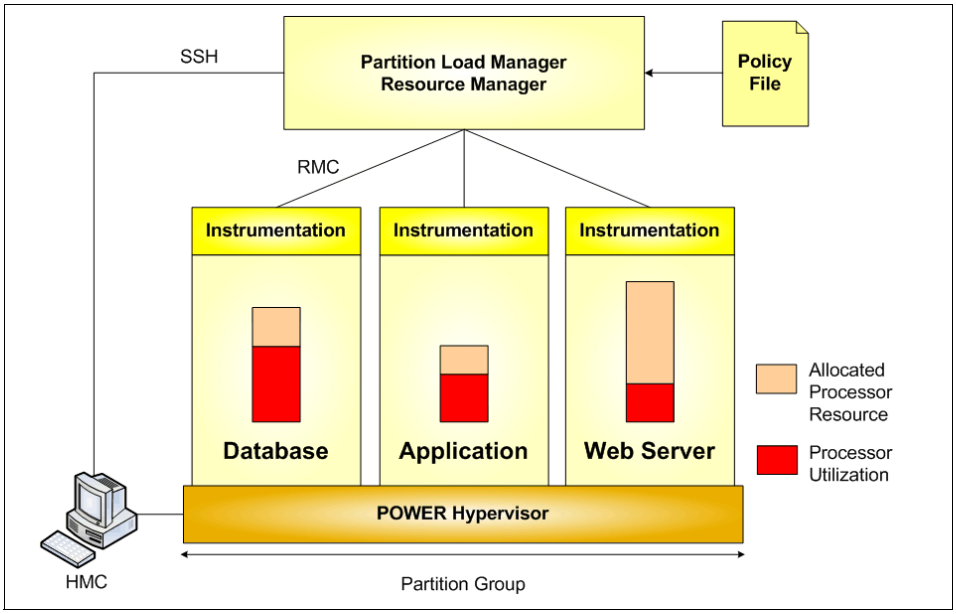

Figure 1-26 Partition Load Manager overview

The policy file defines managed partitions, their entitlements and their thresholds, and organizes the partitions into groups. Every node managed by PLM must be defined in the policy file along with several associated attribute values:

- Optional maximum, minimum, and guaranteed resource values
- The relative priority or weight of the partition
- Upper and lower load thresholds for resource event notification

For each resource (processor and memory), the administrator specifies an upper and a lower threshold for which a resource event should be generated. You can also choose to manage only one resource.

Partitions that have reached an upper threshold become resource *requesters*. Partitions that have reached a lower threshold become resource *donors*. When a request for a resource is received, it is honored by taking resources from one of three sources when the requester has not reached its maximum value:

- A pool of free, unallocated resources
- A resource donor
- A lower priority partition with excess resources over entitled amount

As long as there are resources available in the free pool, they will be given to the requester. If there are no resources in the free pool, the list of resource donors is checked. If there is a resource donor, the resource is moved from the donor to the requester. The amount of resource moved is the minimum of the delta values for the two partitions, as specified by the policy. If there are no resource donors, the list of excess users is checked.

When determining if resources can be taken from an excess user, the weight of the partition is determined to define the priority. Higher priority partitions can take resources from lower priority partitions. A partition's priority is defined as the ratio of its excess to its weight, where excess is expressed with the formula (current amount - desired amount) and weight is the policy-defined weight. A lower value for this ratio represents a higher priority. Figure 1-27 on page 55 shows an overview of the process for partitions.

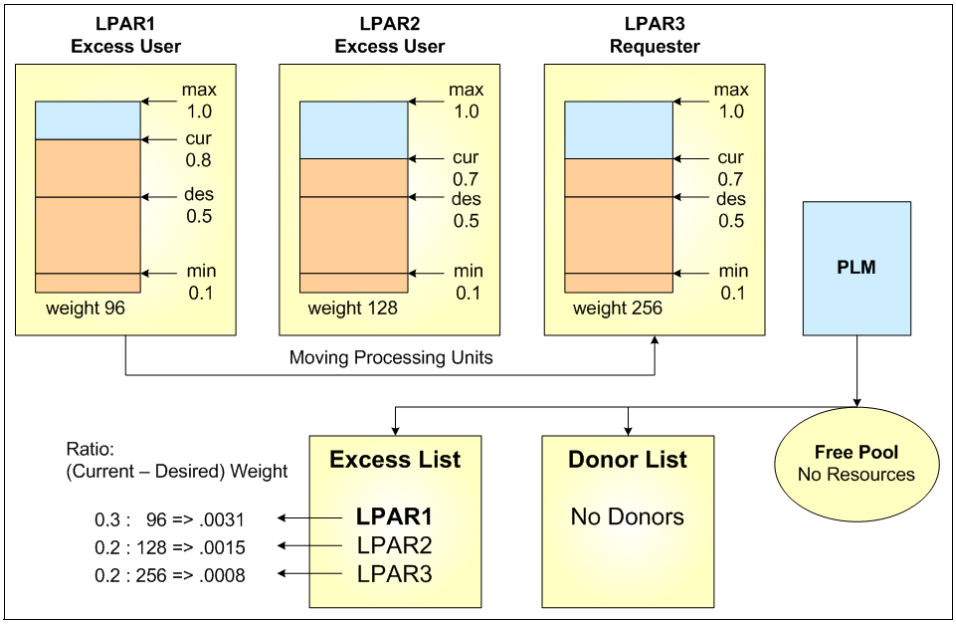

Figure 1-27 PLM resource distribution for partitions

In Figure 1-27, all partitions are capped partitions. LPAR3 is under heavy load and over its high CPU average threshold value for becoming a requestor. There are no free resources in the free pool and no donor partitions available. PLM now checks the excess list to find a partition having resources allocated over its guaranteed value and with a lower priority. Calculating the priority, LPAR1 has the highest ratio number and therefore the lowest priority. PLM deallocates resources from LPAR1 and allocates them to LPAR3.

If the request for a resource cannot be honored, it is queued and re-evaluated when resources become available. A partition cannot fall below its minimum or rise above its maximum definition for each resource.

The policy file, once loaded, is static, and has no knowledge of the nature of the workload on the managed partitions. A partition's priority does not change upon the arrival of high priority work. The priority of partitions can only be changed by some action, external to PLM, loading a new policy.

Partition Load Manager handles memory and both types of processor partitions: dedicated and shared processor partitions. All the partitions in a group must be of the same processor type.

1.7.1 Memory management

PLM manages memory by moving Logical Memory Blocks (LMBs) across partitions. To determine when there is demand for memory, PLM uses two metrics:

- Utilization percentage (ratio of memory in use to available)
- The page replacement rate

For workloads that result in significant file caching, the memory utilization on AIX may never fall below the specified lower threshold. With this type of workload, a partition may never become a memory donor, even if the memory is not currently being used.

In the absence of memory donors, PLM can only take memory from excess users. Since the presence of memory donors cannot be guaranteed, and is unlikely with some workloads, memory management with PLM may only be effective if there are excess users present. One way to ensure the presence of excess users is to assign each managed partition a low guaranteed value, such that it will always have more than its guaranteed amount. With this sort of policy, PLM will always be able to redistribute memory to partitions based on their demand and priority.

1.7.2 Processor management

For dedicated processor partitions, PLM moves physical processors, one at a time, from partitions that are not utilizing them to partitions that have demand for them. This enables dedicated processor partitions running AIX 5L Version 5.2 and AIX 5L Version 5.3 to better utilize their resources. If one partition needs more processor capacity, PLM automatically moves processors from a partition that has idle capacity.

For shared processor partitions, PLM manages the entitled capacity and the number of virtual processors (VPs) for capped or uncapped partitions. When a partition has requested more processor capacity, PLM will increase the entitled capacity for the requesting partition if additional processor capacity is available. For uncapped partitions, PLM can increase the number of virtual processors to increase the partition's potential to consume processor resources under high load conditions. Conversely, PLM will also decrease entitled capacity and the number of virtual processors under low-load conditions, to more efficiently utilize the underlying physical processors.

With the goal of maximizing a partition's and the system's ability to consume available processor resources, the administrator now has two choices:

1. Configure partitions that have high workload peaks as uncapped partitions with a large number of virtual processors. This has the advantage of allowing these partitions to consume more processor resource when it is needed and available, with very low latency and no dynamic reconfiguration. For example, consider a 16-way system utilizing two highly loaded partitions configured with eight virtual processors each, in which case, all physical processors could have been fully utilized. The disadvantage of this approach is that when these partitions are consuming at or below their desired capacity there is an overhead associated with the large number of virtual processors defined.
2. Use PLM to vary the capacity and number of virtual processors for the partitions. This has the advantages of allowing partitions to consume all of the available processor resource on demand, and it maintains a more optimal number of VPs. The disadvantage to this approach is that since PLM performs dynamic reconfiguration operations to shift capacity to and from partitions, there is a much higher latency for the reallocation of resources. Though this approach offers the potential to more fully utilize the available resource in some cases, it significantly increases the latency for redistribution of available capacity under a dynamic workload, since dynamic reconfiguration operations are required.

1.7.3 Limitations and considerations

You must consider the following limitations when managing your system with the Partition Load Manager:

- The Partition Load Manager can be used in partitions running AIX 5L Version 5.2 ML4 or AIX 5L Version 5.3. Linux or i5OS support is not available.
- A single instance of the Partition Load Manager can only manage a single server. However, multiple instances of the Partition Load Manager can be run on a single system, each managing a different server.
- The Partition Load Manager cannot move I/O resources between partitions. Only processor and memory resources can be managed by Partition Load Manager.
- Partition Load Manager requires HMC Release 3 Version 2.6 or newer on an HMC and an IBM @server p5 system.

2

Application development

AIX 5L provides several enhancements that assist you in developing your own software. Among the enhancements described in this chapter are:

- POSIX Realtime compliant features
- Enhancements to memory allocation
- Marking of executable read/write sections
- Java™ enhancements

2.1 POSIX Realtime functions

One standard that has been included in AIX 5L Version 5.3 is the POSIX REALTIME extension. The options available within this standard are described in the section on System Interfaces, Issue 6, 2003 IEEE and the Open Group book, available on the following Web site:

http://www.opengroup.org

AIX 5L Version 5.3 provides system interfaces for the following options:

- Memlock
- Spin locks
- Clocks
- Priority scheduling
- Shared memory objects
- Semaphores
- Timers
- Barriers
- Thread options
- Message passing
- Advisory info

These options are discussed in detail in the sections that follow.

2.1.1 Memlock

The Memlock option uses POSIX MEMLOCK and POSIX MEMLOCK RANGE interfaces to provide a POSIX-compliant implementation for the memory locking and unlocking function. These interfaces allow repeated locking and unlocking of memory within the lifetime of a process to support applications that undergo major mode changes where, in one mode, locking is required, but in another it is not. The locking function guarantees the residence in memory of all, or portions of, a process address space including the text, data and stack areas, shared libraries and shared memory areas.

Memory locking is used to eliminate the indeterminacy introduced by paging and swapping.

AIX 5L Version 5.3 provides the following interfaces for memlock:

mlock() Causes the whole pages containing any part of the address space of the process starting at a specifiable address and continuing for a specifiable number of bytes to be memory-resident until unlocked or until the process exits or executes another process image.

munlock() Unlocks the whole pages containing any part of the address space of the process starting at a specifiable address and continuing for a specifiable number of bytes, regardless of how many times mlock() has been called by the process for any of the pages in the specified range.

If any of the pages in the range specified in a call to the munlock() function are also mapped into the address spaces of other processes, any locks established on those pages by another process are unaffected by the call of this process to the munlock() function. If any of the pages in the range specified by a call to the munlock() function are also mapped into other portions of the address space of the calling process outside the range specified, any locks established on those pages through other mappings are also unaffected by this call.

mlockall() Causes all of the pages mapped by the address space of a process to be memory-resident until unlocked or until the process exits or executes another process image.

munlockall() Unlocks all currently mapped pages of the address space of the process. Any pages that become mapped into the address space of the process after a call to the munlockall function are not locked, unless there is an intervening call to the mlockall function. If pages mapped into the address space of the process are also mapped into the address spaces of other processes and are locked by those processes, the locks established by the other processes are unaffected by a call to the munlockall function.

Note: Stacking calls to mlock() and mlockall() in a single process has the same effect as a single call to one of these interfaces.

Stacking calls to munlock() and munlockall() in a single process has the same effect as a single call to one of these interfaces.

These functions are available by default. They use the standard C library (libc.a). There are no tunables to turn them on or off. However, the calling process must have the root user authority to use these functions.

In general, plock() and mlock() calls should not be mixed. In certain situations they can run into conflict with each other. For instance:

- It is forbidden to call the POSIX memory locking interfaces while a flag is set in the U_lock field of the user structure, due to a call to plock(*LOCK).
- It is forbidden to call plock() interface while a memory region is still locked due to a call to mlockall() or mlock().

2.1.2 Spin locks

Spin locks are synchronization objects used to allow multiple threads to serialize their access to shared data. They represent an extremely low-level synchronization mechanism suitable primarily for use on shared memory multi-processors. When a caller requests a spin lock that is already held, it typically spins in a loop testing whether the lock has become available. Such spinning wastes processor cycles. Spin locks provide an efficient locking mechanism for short-duration locks.

Locking mechanisms typically must trade off processor resources consumed while setting up to block the thread and the processor resources consumed by the thread while it is blocked. Spin locks require very little resources to set up the blocking of a thread compared to mutexes.

AIX 5L Version 5.3 provides the following interfaces for a POSIX compliant implementation of POSIX SPIN LOCKS:

pthread_spin_destroy()	Destroy a spin lock object. The pthread_spin_destroy function destroys the spin lock and releases any resources used by the lock.
pthread_spin_init()	Initialize a spin lock object. The pthread_spin_init function allocates any resources required to use the spin lock and initializes the lock to an unlocked state.
pthread_spin_lock()	Lock a spin lock object.
pthread_spin_trylock()	Lock a spin lock object. The pthread_spin_lock function locks the spin lock. The calling thread shall acquire the lock if it is not held by another thread. Otherwise, the thread spins (that is, does not return from the pthread_spin_lock call) until the lock becomes available. The pthread_spin_trylock function locks the spin lock if it is not held by any thread. Otherwise, the function fails.

pthread_spin_unlock() Lock a spin lock object.

The pthread_spin_unlock function releases the spin lock referenced by the lock parameter which was locked using the pthread_spin_lock function or the pthread_spin_trylock function.

The result of referring to copies of a spin lock object in calls to pthread_spin_destroy function, pthread_spin_lock function, pthread_spin_trylock function, or the pthread_spin_unlock function is undefined.

You can use the dbx program to debug spinlocks. To do so, set the AIXTHREAD_SPINLOCKS environment variable to ON.

2.1.3 Clocks

The IEEE standard defines functions for obtaining system time. Implicit behind these functions is a mechanism for measuring the passage of time. The specification makes this mechanism explicit and calls it a clock. The CLOCK_REALTIME clock required by POSIX is a higher resolution version of the clock that maintains system time. This is a *system-wide* clock in the sense that it is visible to all processes.

The interface was extended to define additional clocks. This was done because many realtime platforms support multiple clocks. However, such clocks do not necessarily represent hardware devices nor are they necessarily system-wide.

POSIX_MONOTONIC_CLOCK, POSIX_CLOCK_SELECTION, POSIX_CPUTIME, and POSIX_THREAD_CPUTIME provide a POSIX-compliant implementation of the CLOCKS option as part of the POSIX Realtime support.

POSIX_MONOTONIC_CLOCK provides a clock whose value cannot be set using clock_settime() and that cannot have backward clock jumps.

POSIX_CLOCK_SELECTION provides the following three interfaces:

clock_nanosleep() Suspend current thread from execution until the time specified has elapsed.

pthread_condattr_getclock() Get the clock selection condition variable attribute.

pthread_condattr_setclock() Set the clock selection condition variable attribute.

The clocks supported by these interfaces are:

- The realtime clock, CLOCK_REALTIME
- The monotonic clock, CLOCK_MONOTONIC

CPU-time clocks are not supported by these interfaces.

POSIX_CPUTIME provides access to a process CPU-time clock that can be invoked to measure execution time of a process using the clock_getcpuclockid() interface.

POSIX_THREAD_CPUTIME provides support for a thread CPU-time clock that measures execution time of threads with the pthread_getcpuclockid() interface. It is a thread-wide clock rather than a process- or system-wide clock.

2.1.4 Priority scheduling

Realtime applications require that process scheduling be fast and deterministic: the highest-priority process is always run first and, among processes of equal priority, the process that has been runnable for the longest time is executed first.

POSIX PRIORITY SCHEDULING option provides a POSIX-compliant implementation of POSIX process scheduling extensions.

POSIX PRIORITY SCHEDULING provides the following interfaces:

sched_setparam()	Sets the scheduling parameters of a process.
sched_getparam()	Returns the scheduling parameters of a process.
sched_setscheduler()	Sets the scheduling policy and parameters of a process.
sched_getscheduler()	Returns the scheduling policy of a process.
sched_get_priority_min()	Returns the minimum priority value for a scheduling policy.
sched_get_priority_max()	Returns the maximum priority value for a scheduling policy.
sched_rr_get_interval()	Returns the execution time limit of a process.
sched_yield()	Makes the calling thread yield the processor.

It should be noted that AIX 5L Version 5.2 already conforms to POSIX Realtime Process Scheduling option to a large extent. Version 5.3 fills in the gaps.

2.1.5 Shared memory objects

Shared memory is used to share data among several processes. For virtual memory systems, a shared memory interface is required to prevent processes from accessing each other's data. However, in unprotected systems such as are found in embedded controllers, a shared memory interface is needed to provide a portable mechanism to allocate a region of memory to be shared and then to

communicate the address of that region to other processes. Therefore, to support realtime applications, a shared memory interface must be able to allocate and name an object to be mapped into memory for potential sharing and make the memory object available within the address space of a process. The interface must also provide a mechanism for deleting the name of the sharable object that was previously created.

POSIX SHARED MEMORY OBJECTS option fulfils these requirements by means of the following interfaces:

shm_open()	Establish a connection between a shared memory object and a file descriptor. If it does not already exist, first create the object.
shm_unlink()	Remove a shared memory object.

2.1.6 Semaphores

A realtime system requires a robust synchronization mechanism between the processes within the same overall application. Such synchronization has to allow more than one scheduled process mutually-exclusive access to the same resource. POSIX SEMAPHORES option provides a POSIX-compliant implementation for the POSIX semaphores which offer such a synchronization mechanism for realtime applications. The POSIX standard for realtime inter-process communication describes two types of semaphores: named semaphores and unnamed semaphores.

POSIX SEMAPHORES provides the following interfaces:

sem_close()	Close a named semaphore.
sem_destroy()	Destroy an unnamed semaphore.
sem_getvalue()	Get the value of a semaphore.
sem_init()	Initialize an unnamed semaphore.
sem_open()	Establish a connection between a named semaphore and a process.
sem_post()	Unlock a semaphore.
sem_trywait()	Lock the semaphore only if the semaphore is currently not locked.
sem_unlink()	Remove a named semaphore.
sem_wait()	Lock the semaphore by performing a semaphore lock operation on that semaphore.
sem_timedwait()	Lock a semaphore within a specified timeout. If the semaphore cannot be locked without waiting for another

process or thread to unlock the semaphore by performing a sem_post() function, this wait shall be terminated when the specified timeout expires.

2.1.7 Timers

Two types of timers are required for a system to support realtime applications.

One-shot	A one-shot timer is a timer that is armed with an initial expiration time, either relative to the current time or at an absolute time. The timer expires once and then is disarmed.
Periodic	A periodic timer is a timer that is armed with an initial expiration time, either relative or absolute, and a repetition interval. When the initial expiration occurs, the timer is reloaded with the repetition interval and continues counting.

POSIX TIMERS option provides a POSIX-compliant implementation for the timers extension. This option relies for timer management not only on the real time clock, but also on the other clocks defined in other POSIX Extensions.

POSIX TIMERS provides the following interfaces:

clock_getres()	Returns the resolution of a specific clock.
clock_gettime()	Returns the time as measured by a specific clock.
clock_settime()	Sets the time of a specific clock.
nanosleep()	Sleeps for an amount of time as measured by the real time clock as a relative interval.
timer_create()	Creates a per-process timer using a specific clock as the timing base.
timer_delete()	Deletes a per-process timer previously created by timer_create.
timer_getoverrun()	Returns the timer overrun count for a timer previously created by timer_create.
timer_gettime()	Returns the amount of time until a timer previously created by timer_create expires, as well as the reload value of the timer.
timer_settime()	Sets the time until next expiration of a timer and its reload value.

2.1.8 Barriers

POSIX BARRIERS provide a POSIX-compliant implementation for the POSIX barriers option. They are used for pthreads synchronization purposes and are typically used in parallel DO/FOR loops to ensure that all threads have reached a particular stage in a parallel computation before allowing any to proceed to the next stage. Highly efficient implementation of barriers is possible on machines which support a *Fetch and Add* operation. Although they can be implemented with mutexes and condition variables, such implementations of barriers are significantly less efficient than is possible.

POSIX BARRIERS provide the following interfaces:

pthread_barrier_init()	Initialize a barrier object.
pthread_barrier_destroy()	Destroy a barrier object.
pthread_barrierattr_init()	Initialize the barrier attributes object.
pthread_barrierattr_destroy()	Destroy the barrier attributes object.
pthread_barrierattr_setpshared()	Set the process-shared attribute of the barrier attributes object.
pthread_barrierattr_getpshared()	Get the process-shared attribute of the barrier attributes object.
pthread_barrier_wait()	Synchronize at a barrier.

The symbol _POSIX_BARRIERS is defined to 200112L in <unistd.h>.

Barriers can be debugged by setting environment variable AIXTHREAD_BARRIER_DEBUG to ON.

2.1.9 Thread options

POSIX THREAD PRIORITY SCHEDULING, POSIX THREAD PRIO PROTECT, and POSIX THREAD PRIO INHERIT options provide a POSIX implementation of these options as part of POSIX Realtime support. This option also provides additional thread interfaces from the POSIX TIMEOUTS option.

POSIX THREAD PRIORITY SCHEDULING provides the following interfaces:

pthread_attr_getinheritsched()	Get the inheritsched attribute.
pthread_attr_setinheritsched()	Set the inheritsched attribute.
pthread_attr_getschedpolicy()	Get the schedpolicy attribute.
pthread_attr_setschedpolicy()	Set the schedpolicy attribute.
pthread_attr_getscope()	Get the contentionscope attribute.

pthread_attr_setscope()	Set the contentionscope attribute.
pthread_getschedparam()	Get the current schedparam attributes of a thread.
pthread_setschedparam()	Set the current schedparam attributes of a thread.
pthread_setschedprio()	Set the scheduling priority for the thread.

POSIX THREAD PRIO PROTECT (and INHERIT) provide the following interfaces:

pthread_mutex_getprioceiling()	Get the priority ceiling of a mutex.
pthread_mutex_setprioceiling()	Set the priority ceiling of a mutex.
pthread_mutexattr_getprioceiling()	Get the prioceiling attribute of the mutex attributes object.
pthread_mutexattr_setprioceiling()	Set the prioceiling attribute of the mutex attributes object.
pthread_mutexattr_getprotocol()	Get the protocol attribute of the mutex attributes object.
pthread_mutexattr_setprotocol()	Set the protocol attribute of the mutex attributes object.

POSIX TIMEOUTS provide the following interfaces:

pthread_mutex_timedlock()	Locks a mutex, with a timed wait if the mutex is already locked.
pthread_rwlock_timedrdlock()	Locks a read-write lock for reading.
pthread_rwlock_timedwrlock()	Locks a read-write lock for writing.

2.1.10 Message passing

Many applications, including both realtime and database applications, require a means of passing data between processes of the same application, executing on one or more processors. Most conventional interfaces for inter-process communication are insufficient for such data passing for realtime applications. POSIX MESSAGE PASSING option provides interfaces that enable such data passing. It allows processes to communicate through system-wide queues.

POSIX MESSAGE PASSING provides the following interfaces:

mq_close()	Close a message queue.
mq_getattr()	Get message queue attributes.

mq_notify()	Register the calling process to be notified of message arrival at an empty message queue.
mq_open()	Establish a connection between a process and a message queue.
mq_receive()	Receive the oldest of the highest priority messages from the message queue.
mq_send()	Send a message to a message queue.
mq_setattr()	Set message queue attributes.
mq_timedreceive()	Receive a message from a message queue within a specified timeout.
mq_timedsend()	Send a message to a message queue within a specified timeout.
mq_unlink()	Remove a message queue.

2.1.11 Advisory Info

The purpose of the Advisory Info option is to provide a method for an application to advise the OS on its (the application's) future behavior with respect to a given file. In addition, it provides a way for the application to pre-allocate certain resources from the OS as well as specify the alignment of memory allocated from the system.

It provides the following interfaces:

posix_fadvise()	Advises the system on the expected future behavior of the application with regards to a given file. The system may take this advice into account when performing operations on file data specified by this function.
posix_fallocate()	Reserves storage space on the file system media for a given File Descriptor.
posix_memalign()	Allocates memory with specified alignment.
posix_madvise()	Advises the system on the expected future behavior of the application with regard to a given range of memory. The system may take this advice into account when performing operations on the data in memory specified by this function.

The posix_fallocate() and posix_memalign() functions actually define behaviors while the advise functions simply provide the OS the opportunity to optimize future behaviors.

Advice must be one of the following values specified in fcntl.h:

- POSIX_FADV_NORMAL
- POSIX_MADV_NORMAL
- POSIX_FADV_SEQUENTIAL
- POSIX_MADV_SEQUENTIAL
- POSIX_FADV_WILLNEED
- POSIX_MADV_WILLNEED
- POSIX_FADV_RANDOM
- POSIX_MADV_RANDOM
- POSIX_FADV_DONTNEED
- POSIX_MADV_DONTNEED
- POSIX_FADV_NOREUSE

POSIX_FADV_NORMAL and POSIX_MADV_NORMAL mean the application has no advice to give on its behavior with respect to the specified data. It is the default characteristic if no advice is given for an open file.

The POSIX_FADV_SEQUENTIAL and POSIX_MADV_SEQUENTIAL advice tells the OS to expect serial access. Typically the system will prefetch the next several serial accesses in order to overlap I/O. It may also free previously accessed serial data if memory is tight. If the application is not doing serial access it can use POSIX_FADV_WILLNEED and POSIX_MADV_WILLNEED to accomplish I/O overlap, as required. When the application advises POSIX_FADV_RANDOM or POSIX_MADV_RANDOM behavior, the OS usually tries to fetch a minimum amount of data with each request and it does not expect much locality. POSIX_FADV_DONTNEED and POSIX_MADV_DONTNEED allow the system to free up caching resources since the data will not be required in the near future.

POSIX_FADV_NOREUSE tells the system that caching the specified data is not optimal because the application expects to access the specified data once and then not reuse it thereafter.

2.2 Enhanced libc.a

AIX 5L Version 5.3 has implemented a number of new APIs for the libc.a library. A number of existing functions in this library have been modified as well. This has been done to improve system performance, reliability, and servicability for the identification and authentication services provided by this library.

The following list describes new and modified APIs. Functions ending in the letter *x* do everything that their old counterparts did and more, as described here.

authenticate*x*()	Introduces the ability to communicate authentication method information between library routines.
chpass*x*()	Introduces the ability to communicate authentication information about the mechanisms which were actually used during the authentication process.
loginrestrictions*x*()	Introduces the ability to examine all methods used in an authentication decision and query all of those methods to verify that all of the methods grant the user access.
passwdexpired*x*()	Introduces the ability to examine all methods used in an authentication decision and query all of those methods to determine which passwords have expired, and then update those passwords using chpassx().
getgroupattrs()	Allows an application to request one or more group attributes with a single library call.
getuserattrs()	Allows an application to request one or more user attributes with a single library call.
putgroupattrs()	Allows an application to modify (update) one or more group attributes with a single library call.
putuserattrs()	Allows an application to modify (update) one or more user attributes with a single library call.
getgroupattr()	This function has been modified to make a single call to getgroupattrs() using the supplied name, attribute, and type information.
getuserattr()	This function has been modified to make a single call to getuserattrs() using the supplied name, attribute, and type information.
putgroupattr()	This function has been modified to make a single call to putgroupattrs() using the supplied name, attribute, and type information.
putuserattr()	This function has been modified to make a single call to putuserattrs() using the supplied name, attribute, and type information.
usersec.h	This header file has been updated to include function prototypes for each of the new functions listed above.

2.3 New malloc() algorithm

A new malloc allocator called Watson Malloc has been implemented in AIX 5L Version 5.3. Watson Malloc provides improvement over the default allocator in the areas of memory fragmentation and speed performance in massively multi-threaded applications. In addition, two new options have been added for the default allocator, a thread cache front end and a new buckets-based front end.

The current default allocator is the Yorktown allocator. With respect to speed, the Yorktown allocator does not efficiently handle repeated small to medium size requests. This deficiency was previously addressed by adding the Malloc Buckets algorithm. Malloc Buckets, however, provides no way to consolidate freed memory into the heap. The new bucket allocator will allow freed memory to be reclaimed. Through the use of the new bucket allocator, the Watson Allocator handles small requests quickly and with comparatively little wasted memory. The Watson Allocator also performs quicker than the default allocator, and with less internal fragmentation than the old bucket allocator.

The Watson Allocator can be configured in three distinct ways to try and identify which sections reveal the largest gains. It can be enabled with caching mechanisms (a per thread cache and an adaptive heap cache) and with the new bucket allocator. It can be enabled without the caching mechanisms and with the new bucket allocator. It can also be enabled without the caching mechanisms and without the new bucket allocator. The new bucket allocator and thread cache have been adapted to work with the Yorktown allocator.

These enhancements lead to increased performance and reduced resource overhead for large multithreaded applications. However, any of these configurations can be disabled if they do not prove to be beneficial in regard to speed or memory usage.

Only one MALLOCTYPE can be specified at a time. The MALLOCTYPE is specified prior to process startup and cannot be changed for a running process as in the following:

```
MALLOCTYPE=watson
```

2.4 Improvements to malloc subsystem

Currently user actions required to enable the malloc environment are quite confusing. They are complicated by the fact that some of the environment variables that the malloc subsystem supports often have conflicting purposes. This situation has been simplified in AIX 5L Version 5.3 by reducing the number of environment variables to three and redefining the attributes they can assume.

They are MALLOCTYPE, MALLOCOPTIONS, and MALLOCDEBUG. MALLOCOPTIONS is a new environment variable that has been added to take care of all current and future options to the MALLOCTYPE allocators. It supplants the MALLOCBUCKETS, MALLOCMULTIHEAP, and MALLOCDISCLAIM environment variables which have been deprecated since they really are just algorithmic options which conceptually should not require a separate environment variable.

The following are the definitions of these three environment variables and some of the attributes they can assume:

MALLOCTYPE	Used to select allocator type. Attributes include: 3.1, default allocator, Watson, user.
MALLOCOPTIONS	Allocator options. Attributes include: buckets, disclaim, threadcache, multiheap.
MALLOCDEBUG	debugging options. Attributes include: catch_overflow, report_allocations, Malloc Log, Malloc Trace, validate_ptrs.

AIX 5L Version 5.3 has also implemented the following enhancements:

- Threadcache option which reduces contention for heap lock is now available for applications using the default (Yorktown) as well as the Watson Allocator.
- Malloc Log and Malloc Trace functions have been enhanced. Malloc Log offers an extended log and provides for automatic logging of allocations and metadata. Malloc Trace now logs more data including heap and thread IDs.
- The output debug option

  Currently, all malloc debugging options which produce printed output send their output to stderr. The output option provides a mechanism to instruct the subsystem to send printed output to a different file stream as in the following:

  ```
  $ MALLOCDEBUG=output:/dev/null
  $ MALLOCDEBUG=output:stdout
  ```

- The continue debug option

  Currently, many malloc debugging options call the abort() procedure when they encounter an error. It is often the case, however, that a developer may wish to debug other classes of errors first and would prefer that less serious errors do not produce fatal flaws. With the continue option enabled, synchronous errors do not call the abort() function. Error messages are still logged to the appropriate channels as in the following:

  ```
  $ MALLOCDEBUG=continue
  ```

- Watson allocator now supports the following debugging features:

  catch_overflow This is a debugging option that attempts to catch writes and reads past the end of a buffer. This is done by allocating an extra guard page and positioning the allocation in such a manner that the end of the allocation is flush against the guard page. The guard page is then protected so that any access will cause a segfault.

  Malloc Log This is a debugging option that creates a live database of active malloc allocations that can be accessed through the use of a debugger.

- DBX malloc command

  Malloc debugging features have been integrated into DBX. This would allow a developer to query the current state of the malloc subsystem without resorting to the creation of complex, unwieldy scripts requiring internal knowledge of the malloc subsystem. An application developer will be able to get information on such items as specific malloc allocations, organization of the heaps, and the free space tree. These features are supported in both live processes and core file dbx sessions. DBX malloc command works in tandem with Malloc Log and allows key-based searches for such items as heap, address, size, and PID.

  The following are some examples:

```
(dbx) malloc  prints out options, heap statistics, brk statistics
(dbx) malloc freespace prints out all free nodes in the process's heaps
```

2.5 Block device mapping

A device is treated like an ordinary file in many respects, but previous AIX releases never had the ability to map a device. AIX 5L Version 5.3 extends the mapping capability of the mmap subroutine to block devices.

A block device is a special file that provides access to a device driver that presents a block interface. A block interface to a device driver requires data access in blocks of a fixed size. The interface is typically used for data storage devices.

AIX 5L Version 5.3 utilizes block device mapping to improve performance in the file system maintenance commands. The Version 5.3 JFS2 **fsck** command in particular makes use of this new capability to reduce the access time to file system meta data. Also, the control of the memory used for buffering is greatly simplified by mapping the device that the file system resides on.

The user of block device mapping should be aware of the following characteristics and restrictions:

- All operations valid on memory resulting from mmap() of a file will be valid on memory resulting from mmap() of a block device.
- The mapped device is accessed using the device files in /dev. The mapping function is limited to devices opened using the block special file.
- The mmap subroutine allows mapping of a device by only one process at a time.
- The mapped device data cache will not be coherent with respect to read and write system calls to the device. Nor will it be coherent with respect to the file system cache when accessing a device that contains a mounted file system.

2.5.1 System calls and subroutines for block device mapping

The new block device mapping capability has the following impact on system calls and subroutines:

open()

A program that wants to do device mapping will open the special file in the normal way. No special flags or arguments are needed. All existing interfaces and interactions with the file descriptor returned by open will continue to operate as before.

mmap()

The mapping function is initiated by a call to mmap specifying the file descriptor for the device. The mmap call is the same as for a regular file mapping. The resultant mapping obeys the semantics of regular file mapping. If the file descriptor refers to a device that has already been mapped, the mmap call will fail and set errno to EAGAIN.

msync()

The data that is written to the mapped device segment can be flushed to disk using the msync call. The flags and arguments used for regular files also apply to devices.

disclaim()

The disclaim subroutine can be used to release pages in the mapped segment.

2.6 Eclipse Runtime Environment

Eclipse Runtime Environment is shipped with AIX 5L Version 5.3. This provides a platform on which Eclipse-based tools, including the new **procmon** performance monitoring tool, will run. AIX 5L does not support Eclipse as a development platform.

2.7 Cryptographic sum command

Users often use the **sum** command to generate a checksum to verify the integrity of a file. However, it is possible that two distinct files will generate the same checksum. A cryptographic sum command, **csum**, has been implemented in AIX 5L Version 5.3 that offers a more reliable tool to verify file integrity. This command allows users to generate message-digests using the AIX Cryptographic Library. A cryptographic checksum is considered secure because it is computationally infeasible to construct data to generate a known checksum, and vice-versa.

The **csum** command allows users the option to select the algorithm they prefer, including both MD5 and SHA-1, which are considered secure. It is estimated that the order of 2^{64} operations would be required to derive two different files which generate the same MD5 message-digest and that the order of 2^{128} operations would be needed to derive a file that would generate a specified MD5 message-digest.

csum will aid in improving the AIX e-fix upgrade process by offering users a mechanism to verify that a file has not been tampered with or corrupted during download.

The **csum** command is installed as part of the bos.rte.commands fileset.

2.8 Perl 5.8.2

The latest version of Perl that ships with AIX 5L is Perl 5.8.0, which was first shipped on AIX 5.2.0. A number of reported problems in Perl 5.8.0 have been fixed by the Perl community as well as the AIX Perl team. Perl 5.8.2, the latest version of the code, is shipped with AIX 5L Version 5.3, as can be shown with the following command:

```
# perl -v
This is perl, v5.8.2 built for aix-thread-multi
Copyright 1987-2003, Larry Wall
```

```
Perl may be copied only under the terms of either the Artistic License or the
GNU General Public License, which may be found in the Perl 5 source kit.
Complete documentation for Perl, including FAQ lists, should be found on this
system using `man perl' or `perldoc perl'.  If you have access to the Internet,
point your browser at http://www.perl.com/, the Perl Home Page.
```

You can run the **perldoc** command to view the delta in changes between two Perl versions. The first command line in the following displays the delta in changes between Perl version 5.8.0 and 5.8.1 and the second one does the same between Perl version 5.8.1 and 5.8.2:

```
$ perldoc perl581delta
$ perldoc perldelta
```

The Perl environment is packaged and shipped in two filesets: perl.rte and perl.man.en_US. The perl.rte fileset is on the first AIX CD and is automatically installed when installing AIX 5L Version 5.3 or when migrating from a previous level of AIX.

The man pages fileset (perl.man.en_US) is shipped on the second CD and is not automatically installed.

Useful information is documented in the /usr/lpp/perl.rte/README.perl.aix file.

Perl installs in /usr/opt/perl5. Links for the Perl executables in /usr/opt/perl5/bin to /usr/bin have been added.

The 64-bit and 32-bit versions are packaged together, with the 32-bit version being the default version. Both versions reside under the /usr/opt/perl5 directory. Both versions are Perl thread capable, built using the newer ithreads (interpreter threads) and also have built-in support for PerlIO.

This version of Perl also supports loading of large files into memory using the maxdata linker option passed in at compile time.

The 64-bit version was built using both optimum 64-bit support along with 64-bit integer support.

Also of interest is the libperl.a library located in the following places under /usr/opt/perl5:

- ./lib/5.8.2/aix-thread-multi/CORE/libperl.a
- ./lib64/5.8.2/aix-thread-multi-64all/CORE/libperl.a

Switching to use the 64-bit version only requires redirecting the Perl symbolic links in /usr/bin to point to the 64-bit versions of the same command in /usr/opt/perl5/bin.

The /usr/opt/perl5 directory includes scripts you can use to change the links for you.

To switch from 32-bit to 64-bit Perl, use the following script:

```
/usr/opt/perl5/link_perl_64
```

To switch from 64-bit to 32-bit Perl, use the following script:

```
/usr/opt/perl5/link_perl_32
```

Note: You must run these scripts as root for them to work.

2.9 SVR4 linking affinity

Nowadays, there are two major object file formats used by the operating systems:

XCOFF The COFF enhanced by IBM and others. The original COFF (Common Object FIle Format) was the base of SVR3 and BSD 4.2 systems.

ELF Executable and Linking Format that was developed by AT&T and is a base for SVR4 UNIX.

The modifications done to the static and dynamic linker are to make linking on AIX look and behave more like that on the SVR4.

2.9.1 dlsym() function

The dlsym() function accepts, as a first argument, either the value returned from the call to dlopen(), or one of the special handles RTLD_DEFAULT, RTLD_NEXT, or RTLD_SELF. It also accepts, as a second argument, a special symbol named RTLD_ENTRY. These special handles and special symbol name are defined in the <dlfcn.h> header file as follows:

```
#define RTLD_DEFAULT (-1)
#define RTLD_NEXT    (-2)
#define RTLD_SELF    (-3)
#define RTLD_ENTRY   ((const char *)(-1)
```

In the case of the special handle RTLD_DEFAULT, dlsym() searches for the named symbol starting with the first object loaded and proceeding through the list of initial loaded objects, and any global objects obtained with dlopen(), until a match is found. This search follows the default model employed to relocate all objects within the process.

In the case of the special handle RTLD_NEXT, dlsym() searches for the named symbol in the objects that were loaded following the object from which the dlsym() call is being made.

In the case of the special handle RTLD_SELF, dlsym() searches for the named symbol in the objects that were loaded starting with the object from which the dlsym() call is being made.

In the case of the special symbol name RTLD_ENTRY, dlsym() returns the module's entry point. The entry point is a value of the module's loader section symbol marked as entry point (bit LDR_ENTRY is set in l_smtype member of the LDSYM structure - see the <loader.h> header file).

In the case of RTLD_DEFAULT, RTLD_NEXT, and RTLD_SELF, if the objects being searched have been loaded from dlopen() calls, dlsym() searches the object only if the caller is part of the same dependency hierarchy, or if the object was given global search access, that is, it has been opened in dlopen() RTLD_GLOBAL mode by dlopen().

2.9.2 New options of the ld command for SVR4 affinity

The static linker utility, the **ld** command, is enhanced by the following new options:

-b svr4 Sets the SVR4 affinity of the linker. The option changes the meaning of some other options on the command line and the standard behavior of the linker.

-V Prints the version string of the static linker to stderr.

The behavior of the following options were modified and are valid only when -b svr4 is specified:

-dn Specifies that **ld** uses static linking. This option is equivalent to -b nso option.

-dy Specifies that **ld** uses dynamic linking. Dynamic linking is a default. This option is equivalent to -b so option.

-R *path* A colon-separated list of directories used to specify a runtime library search path. If present and not NULL, it is recorded in the output file's loader section. This is used when linking an executable with shared libraries. Multiple instances of this option are concatenated together. Run time linker uses this path to locate shared libraries at runtime.

-z defs Forces fatal error if any undefined symbols remain at the end of the link. This is the default when an executable is

built. It is also useful when building a shared library to assure that the object is self-contained, that is, all its symbolic references are resolved internally.

-z nodefs Allows undefined symbols. This is the default when a shared library is built. When used with executable, the behavior of references to such undefined symbols is unspecified. It is equivalent to -b erok option.

-z multidefs Allows multiple symbol definitions. By default, multiple symbol definitions that occur between relocatable objects (.o files) will result in a fatal error condition. This option suppresses the error condition and allows the first symbol definition to be taken.

-z text In dynamic mode only, it forces a fatal error if any relocations against non-writable sections remain.

-z nowarntext In dynamic mode only, it allows relocations against all mappable sections, including non-writable ones. This is the default when building a shared library.

-z warntext In dynamic mode only, it warns if any relocations against the .text section remain. This is the default when building an executable.

When the -b svr4 flag is set, the following behavior is changed to assure better compatibility:

- The -b E or -b export options take precedence over -b expall or -b expfull. This means that only symbols specified by -b E or -b export options are exported.
- There is no need to specify the -b noentry option when building a shared library. It is the default.
- When producing modules with the -b rtl option specified, multiple definitions are not reported when the symbol is defined in different modules. They are reported only when they occur in the same module.
- The library search path specified with the -L option is not included in the runtime library search path recorded in the file's loader section. Only /lib:/usr/lib path is recorded in the file. The path specified with -b libpath option is recorded in the loader section. It is similar to the -R option, however, unlike the -R option, in case of multiple -b libpath options, only the last path is recorded. Multiple paths are not concatenated.

No matter if the -b svr4 flag is set, during symbol resolution if a symbol is found to belong to an object file within an archive library, all global symbols from this object file are exported.

2.10 Marking executable's read/write sections

The new F_NONEXEC bit can be set in the f_flags field in the XCOFF header as follows:

```
#define F_NONEXEC 0x8000
```

When this bit is set, the read/write sections (.data, .bss) of the executable, shared library, or object file are not executable. If the bit is not set, all read/write sections are executable. You can set or un-set the bit using the **ld** or **ldedit** commands. Explanation of the commands follows.

2.10.1 Updated flags for the ld command

The **ld** command is updated by new flags that enable you to set the F_NONEXEC flag as follows:

-b rwexec Marks the read/write sections in the file as executable. It does not set the F_NONEXEC bit in the XCOFF file header. This is the default.

-b norwexec This is a complimentary option to -b rwexec. It marks all read/write sections in the file as non-executable and indicates to the loader to make the stack non-executable. This option sets the F_NONEXEC bit in the XCOFF file header.

2.10.2 Updated flags for the dump command

The **dump -ov** binary utility command additionally displays the value of the F_NONEXEC flag as RWNONEXEC. When the flag is set, you will get the following format as highlighted in Example 2-1.

Example 2-1 Example of dump -ov command

```
# dump -ov a.out
a.out:
                         ***Object Module Header***
# Sections      Symbol Ptr      # Symbols       Opt Hdr Len     Flags
         4      0x0000190c            236                72     0x9002
Flags=( EXEC DYNLOAD RWNONEXEC )
Timestamp = "Jul 16 13:35:57 2004"
Magic = 0x1df  (32-bit XCOFF)

                         ***Optional Header***
Tsize        Dsize       Bsize       Tstart      Dstart
0x00000acc  0x00000368  0x00001004  0x10000128  0x20000bf4
```

```
SNloader     SNentry     SNtext      SNtoc       SNdata
0x0004       0x0002      0x0001      0x0002      0x0002

TXTalign     DATAalign   TOC         vstamp      entry
0x0005       0x0003      0x20000ed8  0x0001      0x20000ec4

maxSTACK     maxDATA     SNbss       magic       modtype
0x00000000   0x00000000  0x0003      0x010b       1L
```

2.10.3 Updated flags for the ldedit command

If the executable is linked with the RWNONEXEC flag set in the f_flags field of the XCOFF header, then it will have its writable and mappable sections executable. This is not always desirable. The **ldedit** command allows you to set or reset this bit in the file without need to re-link the file.

Use the **ldedit** command to clear the F_NONEXEC flag as follows:

```
ldedit -b rwexec
```

Use the **ldedit** command to set the F_NONEXEC flag as follows:

```
ldedit -b norwexec
```

2.11 Thread-based credentials

GRID applications need the operating system to support thread-based credentials. AIX 5L Version 5.3 introduced a pthread-based API to set and destroy thread-based credential information, such as UID and GID.

Per-thread credentials is the property by which different threads in a process own different credentials and the access rights to a resource are checked at the thread-level, ignoring any process-level credentials. This is a very useful concept because it allows a single process to handle requests requiring different credentials by creating threads with the appropriate credentials for each request.

Traditionally on systems that do not support per-thread credentials, this is accomplished by having the main process run as root and forking another process. There are several disadvantages with this approach, most notably the fact that creating a process and acquiring a reduced or different set of credentials involves many system calls and is usually an expensive operation.

AIX 5L Version 5.3 enables you to create threads with a reduced set of credentials. The pthread_create_withcred_np() function allows you to create a thread and switch the base credentials for the thread. The call of the thread

creation by itself is very similar to the pthread_create() function. Comparing to pthread_create(), the pthread_create_withcred_np() additionally accepts the structure that defines the credentials. The syntax is the following:

```
#include <pthread.h>
#include <sys/cred.h>
int pthread_create_withcred_np(pthread_t *thread, const pthread_attr_t *,
        void *(*start_subroutine)(void), void *arg,
        struct __pthrdscreds *credp)
```

The struct __pthrdscreds *credp input to the function is a structure where you set the credentials. The remaining inputs of the system call are the same as for the pthread_create() system call. The following credentials can be modified:

- Effective, real, and saved user IDs
- Effective, real, and saved group IDs
- Supplementary group IDs

The following program excerpt provides an example. The program creates a thread that runs our cred_thread() function with credentials of the user with UID=203 and group with GID=1.

```
# more pt.c
#include <sys/types.h>
#include <sys/cred.h>
#include <pthread.h>
#include <sys/cred.h>

cred_thread()
{
 ... write your thread ...
}

int main()
{
  pthread_t     thread;
  struct __pthrdscreds  thrcrs;
  int rc=0;
...
  thrcrs.__pc_flags=PTHRDSCREDS_INHERIT_UIDS | PTHRDSCREDS_INHERIT_GIDS;
  thrcrs.__pc_cred.crx_ruid=203;
  thrcrs.__pc_cred.crx_uid=203;
  thrcrs.__pc_cred.crx_suid=203;
  thrcrs.__pc_cred.crx_luid=203;
  thrcrs.__pc_cred.crx_acctid=203;
  thrcrs.__pc_cred.crx_gid=1;
  thrcrs.__pc_cred.crx_rgid=1;
  thrcrs.__pc_cred.crx_sgid=1;
```

```
 rc=pthread_create_withcred_np(&thread, NULL, &cred_thread, NULL, &thrcrs);
...
}
```

When creating threads with credentials of other users, only processes entitled to set credentials will succeed. Root users can allow the applications in the context of another identity to use this feature. This is enabled through capabilities of the system infrastructure. Typically a relevant application, like GRID, will pose a question to the root user to provide this authority to take advantage of this feature. It will also require root user to set any other relevant parameters such as noswitchuid and noswitchgid. By defining the *noswitchuid* and *noswitchgid* system parameters the administrator can set the lowest user ID and group ID that a process with credentials capability is allowed to switch to.

2.12 Scalability enhancements

This section describes the improvements made to the operating system to help achieve better scalability.

2.12.1 Kernel round robin locks

The kernel round robin locks (krlock) were introduced in AIX 5L. Version 5.3 makes performance optimizations to them to enable systems to scale over 48 CPUs. The new krlock reduces cache bounces and contentions while spinning on the lock.

When there is a lock miss during a simple_lock() function call, krlock processing takes place. The simple_lock() function is not changed and current applications using simple_lock() on Version 5.2 are compatible with Version 5.3.

2.12.2 Increased resource limit values in /etc/security/limits file

The restrictions for resource limits specified in the /etc/security/limits file have been removed along with the corresponding checks in the **mkuser** and **chuser** commands. Now all resource limits can have a maximum value of 2147483647 or 2^31-1. When combined with the assumed block size of 512 bytes, this results in a maximum of 2**40-1.

Table 2-1 provides a list of the old and new resource limit maximums allowed in the /etc/security/limits file in AIX 5L Version 5.3.

Table 2-1 Resource limits in /etc/security/limits file

Limit	Old value	New value
CPU time	2.147.483.647 (2^31-1)	2.147.483.647 (2^31-1)
File size	4.194.303	2.147.483.647 (2^31-1)
Stack size	523.264	2.147.483.647 (2^31-1)
Memory size	2.147.483.647 (2^31-1)	2.147.483.647 (2^31-1)
Open files	2.147.483.647 (2^31-1)	2.147.483.647 (2^31-1)
Data size	4.194.303	2.147.483.647 (2^31-1)
Core size	2.147.483.647 (2^31-1)	2.147.483.647 (2^31-1)

2.12.3 CPU time limits

In prior releases, a program could avoid a cputime limit set with the **`ulimit`** command by simply blocking, ignoring, or handling the SIGXCPU signal, that says *you're out of time* to the process exceeding the limit. Now, however, such a program can get away with doing this to get around a SOFT limit, but not a HARD limit. The ability to handle an out of time condition is useful if a program wants to format its own debugging information in response to the SIGXCPU.

The SIGXCPU signal is generated not only when the soft limit is reached, but periodically afterwards (the old system did this, too) until the hard limit is reached. At this time, however, the ability to block, ignore, or handle the signal is overridden by the kernel, so the SIGXCPU will inescapably terminate the process.

Note: The process that reaches the cpu_hard limit is killed by the SIGXCPU signal and not with the SIGKILL signal.

This might create a binary compatibility problem, since a program that did not get killed will now get killed! The system administrator has an easy way to control this. If the old behavior is preferred, the administrator can set just a soft limit, and leave the hard limit at infinity. To make the limit absolute, the administrator can set the hard and soft limits to the same value. And to allow a program a grace period in which to clean up, the administrator can set the hard limit a little bit higher than the soft limit.

2.12.4 New heap size and late staggering

The vmgetinfo() function is enhanced by adding the following options that can be passed to the function:

- VM_STAGGER_DATA
- VM_NEW_HEAP_SIZE

The use of these options with the vmgetinfo() function may improve the performance, for example the TPCC numbers for databases, when using large pages.

The VM_STAGGER_DATA option staggers the calling process's current sbreak value by a cumulative per-MCM stagger value. This stagger value must be set through the **vmo** command option data_stagger_interval. The out and arg arguments should be NULL and 0, respectively. An example follows:

```
#include <sys/vminfo.h>
...
int rc=0;
rc=vmgetinfo(NULL, VM_STAGGER_DATA, NULL);
```

The VM_NEW_HEAP_SIZE option sets a new preferred page size for future sbreak allocations for the calling process's private data heap. This page size setting is advisory. The out parameter should be a pointer to a psize_t structure that contains the preferred page size, in bytes, to use to back any future sbreak allocations by the calling process. Presently, only 16 M (0x1000000) and 4 K (0x1000) are supported. The arg parameter should be that of the sizeof(psize_t). An example follows:

```
#include <sys/vminfo.h>
...
 int rc=0;
 psize_t buffer=0x1000000;   // Set to 16M value for LGPG
rc=vmgetinfo(&buffer, VM_STAGGER_DATA, sizeof(psize_t));
```

2.13 Increased inter-process communication limits

In AIX, upper limits are defined for the Inter-Process Communication (IPC) mechanisms, which are not configurable. The individual IPC data structures are allocated and deallocated as needed, so memory requirements depend on the current system usage of IPC mechanisms.

AIX 5L releases prior to Version 5.3 defined the maximum number of semaphore IDs, shared memory segment IDs, and message queue IDs to be 131072 (128 K) for the 64-bit kernel.

To cope with anticipated future scalability demands, AIX 5L Version 5.3 preventively increases the maximum number of data structures for each of the IPC identifier types to 1048576 (1024 K).

This enhancement changes the scalability of the subroutines which are used to get the IPC data structure IDs:

semget()	Returns the semaphore identifier
shmget()	Returns the shared memory identifier
msgget()	Returns the message queue identifier

The maximum numbers of IPC data structure IDs for all three subroutines are as follows:

- 4096 for operating system releases before AIX 4.3.2 (32-bit kernel).
- 131072 for releases AIX 4.3.2 through AIX 5L Version 5.2 (32-bit and 64-bit kernel).
- 1048576 for release AIX 5L Version 5.3 and later (64-bit kernel only).

2.14 Thread support in gmon.out

When applications consisting of multiple steps in which different executables, all built with -p or -pg flags to generate profiling information, are invoked in a sequence, each executable causes the previous gmon.out file to be overwritten. It is often difficult to work through the nested scripts and source code to insert commands or subroutine calls to rename the gmon.out files so that they are not overwritten.

In AIX 5L Version 5.3, the gmon.out file has been made thread-safe, so that each thread from a multi-threaded application has its data in it. The gmon.out file thus created contains one histogram per thread and arc count per thread in the call graph.

The current design of **gprof** provides process-level interpretation of gmon.out. With a thread-level gmon.out **gprof** enables the user to analyze complex multi-threaded applications as well as retain profiling information for a large number of programs all executing from within the same directory.

2.15 Enhanced DBX

Major enhancements have been made to DBX in AIX 5L Version 5.3. The following list identifies some of these enhancements:

- Users now have additional control over the way DBX uses debug events. Debug events are the user-defined breakpoints, tracepoints, and watchpoints that control the execution of the debugged process. Should a user need to disable the effects of a debug event, DBX currently requires that the event be completely removed. In a scenario where an event must be repeatedly removed and added, a user must manually reenter the event each time it is to be enabled. This inconvenience has been removed with the addition of two new DBX subcommands: **`disable`** and **`enable`**. These subcommands allow the DBX user to temporarily disable debug events without permanently removing them.
- The function of the **`map`** subcommand has been extended. The new function adds to DBX the ability to resolve addresses and certain types of symbols to their respective loaded modules. The extensions to the **`map`** subcommand also include facilities for requesting loaded module information for single modules, ranges of modules, and with various levels of detail. In addition, a new internal variable $mapformat has been introduced which allows user customization of the default verbosity level for the **`map`** subcommand.
- The **`stop`** and **`trace`** subcommands have been extended to allow stops and traces that depend on changes in loaded module information. The new debug event types allow user notification upon load or unload of specific modules, or upon load or unload of all modules. The new function also enables DBX to report the affected modules for each load change.
- Currently, the DBX **`use`** subcommand sets the list of directories to be searched when the DBX debug program looks for source files. These source files, when located, are used by the DBX **`list`**, **`edit`** and **`file`** subcommands.

  The “@” (at-sign) is a special symbol that, when included in the list of directories specified with **`use`**, directs DBX to search the directory specified in the object file (the file's location at compile-time) for the source file in question.

  This method of providing source code locations to DBX is cumbersome. If source files reside in nested directory structures and the structures have been relocated (for instance, when debugging remotely with the source mounted), it would be more favorable to simply tell DBX the details of the relocation rather than have to communicate to DBX each and every directory to search.

  The function of the **`use`** subcommand has been extended by allowing the user to specify, along with the search directories, mappings in the form of [<oldpath>=<newpath>]. These mappings are then used in conjunction with the special @ directory. Path replacements are performed in all cases where the beginning of the full-path name as recorded in the object file matches <oldpath>.

- When using DBX to debug a corefile, it is important that dependent modules that were loaded in the process at core creation time are available and unchanged at debug time. DBX relies on the information in these dependent modules to provide correct definitions and locations of text and data values.

  Currently, if any dependent module referenced by the corefile is not available at debug time, DBX displays the fatal error "`cannot open <dependent module path>`" and exits. This behavior prevents any debugging of the corefile from occurring until all of the dependent modules are located. Frequently, however, this restriction is impractical. A missing dependent module may not ultimately contribute any meaningful information to the debug session.

  Furthermore, if a dependent module has changed since core creation time, the definitions and locations of values in the process can be misrepresented to the user by DBX. This usually has the effect of confusing the user and leads them to believe that the debugger is functioning incorrectly. In the extreme case, such as when the incorrect version of the libpthreads.a library is used during debugging, DBX cannot report pthread information to the user. Since DBX does not check for mismatched libraries, and thus does not warn the user, the user has no idea why DBX may be acting erratically.

  DBX now allows the debug session to continue even if any number of dependent modules referenced by the corefile are unreadable. A notification message is displayed by DBX during initialization for each missing dependent module. The message includes the name of the missing module.

  DBX also sends notification messages upon initialization and use of any dependent module referenced in the corefile that is determined to be different than at core file creation. These messages include the name of the mismatched module.

- Several new subcommands have been added to DBX. These include **`proc`**, **`kthread`**, and **`fd`**. The purpose of these subcommands is to display data contained within the data structures used by the kernel to keep track of processes and threads. This is beneficial because it makes information about the debug available that was previously very difficult for users to obtain. Much of the data within the user process or file descriptor structures might have been accessible with various DBX subcommands, but with these new interfaces, all of the data is obtainable from one interface.

  It is expected that these new subcommands will be particularly useful for debugging core files. Previously, a wealth of the information locked in the corefile would be very difficult to extract. The new subcommands expose this data in a highly legible format.

- To facilitate debugging pthreaded code, the scope of DBX function has been enhanced by providing several new subcommands that reveal information about pthread objects and display them in a readable, user-friendly format. These include **`handler`**, **`onceblock`**, and **`resource`** subcommands.

- DBX has been enhanced by adding a subcommand, **corefile**, that will display information about a debuggee <corefile>. Without arguments, the **corefile** subcommand will print basic information about the <corefile>. With arguments, more detailed information can be requested. Some function may be limited in cases where FULLCORE was not enabled at dump-time. In these cases, a warning message is printed, as appropriate.

  The **corefile** subcommand displays information from the header of a corefile, including the executable name, corefile format versioning information, flags indicating which data is available, the signal that caused the crash, and the execution mode of the process that dumped core.

- DBX can now list spinlocks by number, tid, or current state (similar to mutex). It can also list all barriers, just one, or those with/without waiters (similar to condition)

2.16 The tcpdump command

The **tcpdump** command has been updated to Version 3.8 for AIX 5L Version 5.3. As a consequence of this upgrade, **iptrace** and **ipreport** were also changed to use the new upgraded libcap library (version 0.8) for packet capture and dump reading.

The **tcpdump** command prior to AIX 5L Version 5.3 displayed packet timestamps down to 1 ns (10^{-9}s). The open source **tcpdump** command displays timestamps at 10^{-6}s, as BPF (Berkeley Packet Filter) supplies 10^{-6}s accuracy on most platforms. The new **tcpdump** command will have 10^{-6}s time stamp resolution.

A total of 87 protocol printers have been included to facilitate printing when using the **tcpdump** command.

Support for the latest SP switch has been added.

A number of new flags have been added to the **tcpdump** command. Some of the most important ones are:

-D Lists supported link types

-r Does not require root privileges

-X Dumps packets in hex and ASCIII

-vvv More verbose than -v and -vv flags

-E Decrypts ipsec ESP packets

2.17 gprof thread support

AIX 5L Version 5.3 provides support for threads in **gprof**. This enhancement is aimed at adding new function to **gprof** so that it can interpret additional information available for thread-level profiled programs. The additional information would include per thread function call count, per thread arc execution count, and amount of time spent executing a function per thread. The current design of **gprof** provides process-level interpretation of gmon.out. The enhancement to **gprof** is to interpret call graph and histogram information on a per thread basis. The enhancement aims at enabling the user to analyze complex multi-threaded applications as well as retain profiling information about a large number of programs all executing from within the same directory.

The new mode is controlled with a new environment variable, GPROF, with the following syntax:

```
GPROF=[profile:{process|thread}][,][scale:<scaling_factor>][,][file:{one|multi|
multithread}]
```

Where:

- `profile` indicates whether thread or process level profiling is to be done.
- `scaling_factor` represents the granularity of the profiling data collected.
- `file` indicates whether a single or multiple gmon.out files should be generated:
  - multi: creates a file for each process (for each fork or exec)
    - `gmon.out.<progname>.<pid>`
  - multithread: creates a file for each pthread
    - `gmon.out.<progname>.<pid>.Pthread<pid>`
    - Can be used to look at one pthread at a time with **gprof** or **xprofiler**
- The default values are:
  - profile: process
  - scale: 2 for process level and 8 for thread level

    Thread level profiling consumes considerably more memory
  - file: one

The following new flags have been added to optionally separate output into multiple files:

- -g filename
  - Writes the call graph information to the specified output filename.

  - Suppresses the profile information unless -p is used.
- -p filename
  - Writes flat profile information to the specified output filename.
  - Suppresses the call graph information unless -g is used.
- -i filename
  - Writes the routine index table to the specified output filename.
  - If this flag is not used, the index table goes either:
    - At the end of the standard output
    - At the bottom of the filename(s) specified with -p and -g

Format of data itself is unchanged. It is presented in multiple sets. The first set has cumulative data followed by one set of data per pthread.

2.18 Java 1.4.2

Both 32-bit and 64-bit versions of Java 1.4.1 are shipped with AIX 5L Version 5.3. However Java 1.4.2. is supported on AIX 5L Version 5.3 (and 5.1 and 5.2) and is available from:

http://www.ibm.com/developerworks/java/jdk/aix/service.html

2.19 Java 3D

Java 3D version 1.3.1 is part of Java 1.4.2 which is supported on AIX 5L Version 5.3 and is available from:

http://www.ibm.com/developerworks/java/jdk/aix/service.html

2.20 Improved man command presentation

AIX 5L Version 5.3 includes the following improvements to the **man** command presentation of man pages that are shipped on the AIX base media and the documentation CD:

- Sections are now indented. For example, the text of the Purpose section is indented from the heading Purpose, which is lined up in the first column. The same is true of all the sections.
- The description of each flag is indented so it is clear where the description of the flag begins and ends.

- There is now visual differentiation between command names, parameters, file names, and normal descriptive text. This makes the descriptive text easy to read.
- Multi-column tables are now neatly formatted.

3

Storage management

AIX 5L introduces several new features for the current and emerging storage requirements. These enhancements include:

- LVM enhancements
  - Performance improvement of LVM commands
  - Removal of classical concurrent mode support
  - Scalable volume groups
  - Striped column support for logical volumes
  - Volume group pbuf pools
  - Variable logical track group
- JFS2 enhancements
  - Disk quotas support for JFS2
  - JFS2 file system shrink
  - JFS2 extended attributes Version 2 support
  - JFS2 ACL support for NFS V4
  - ACL inheritance support
  - JFS2 logredo scalability
  - JFS2 file system check scalability

3.1 LVM enhancements

The following sections discuss the LVM enhancements in detail.

3.1.1 Performance improvement of LVM commands

A large number of changes and enhancements have been implemented in AIX 5L Version 5.3 to reduce the command execution time of several Logical Volume Manager (LVM) high-level commands. All volume group types will benefit from the improvements made to the following commands:

- **extendvg**
- **importvg**
- **mkvg**
- **varyonvg**
- **chlvcopy**
- **mklvcopy**
- **lslv**
- **lspv**

The scalability enhancements which were made to the LVM of AIX over the past years required a constant effort to adjust the performance of user level commands and library functions to the growing number and size of the utilized resources. The prospect to configure and use a scalable volume group in AIX 5L Version 5.3 which holds up to 1024 hard disk drives initiated several new code changes and additions.

LVM metadata, which is critical for system integrity and operation, needs to be stored in the volume group descriptor area (VGDA) and the volume group status area (VGSA) of every single disk which belongs to a configured volume group. AIX 5L Version 5.3 reduces the number of access incidences to VGDAs and VGSAs that are necessary for LVM operations on volume groups and logical volumes. Required read or write operations on metadata are now executed in parallel by an automatically scaled number of threads. Many other improvements focus on the efficiency of metadata access and optimize the effectiveness of their usage. Specific focus was given to the following commands: **extendvg**, **importvg**, **mkvg**, **varyonvg**, **chlvcopy**, **mklvcopy**, and **lslv**.

To speed up the creation of logical partition copies, the **mklvcopy** script was rewritten in the C programing language. The new **mklvcopy** executables utilize the optimized metadata access routines that were mentioned previous but also exploit a new generic hashtable function integrated in the previous LVM library.

The default algorithm for the **importvg** command was further enhanced to reduce the execution time while maintaining a maximum of integrity protection. With AIX 5L Version 5.3, it is no longer the default to scan every disk of a system for an import operation. In Version 5.3 the **importvg** command uses the **redefinevg** command to get all physical volume identifiers (PVIDs) by reading the VGDA of the key device that is related to the volume group in question and after that only the initial LVM records for those physical volumes are examined. The default method of previous AIX releases used to read the LVM record of every disk in the system trying to match the disks that are listed in the VGDA. Beginning with AIX 5L Version 5.3, this method will be an error path to try other disks in the system if needed.

The **lspv** command, when used without any flags or arguments, applies additional hashing logic while assembling a complete list of all physical disks of a given RS/6000, pSeries, or p5 system.

3.1.2 Removal of classical concurrent mode support

In conjunction with the **importvg** command redesign focused on execution time improvement, the support for classical concurrent mode volume groups has been removed. When trying to import a classical concurrent mode volume group, an error message will inform the system administrator to convert the volume group to an enhanced concurrent mode capable volume group.

The classical concurrent mode volume groups are only supported on Serial DASD and SSA disks in conjunction with the 32-bit kernel. Beginning with AIX 5L Version 5.1 the enhanced concurrent mode volume group was introduced to extend the concurrent mode support to all other disk types. The enhanced concurrent volume groups use AIX group services. AIX 5L Version 5.2 is no longer allowed to create classical concurrent mode volume groups, and finally Version 5.3 removes the support for that volume groups type entirely.

3.1.3 Scalable volume groups

Originally AIX allowed you to configure volume groups (VGs) with a maximum of 32 physical volumes (PVs), no more than 1016 physical partitions (PPs) per disk, and an upper limit of 256 logical volumes (LVs) per VG. This VG type is commonly referred to as the *standard volume group*. To handle the increase in hard disk drive capacity over time, AIX Version 4.3.1 implemented a new volume group factor, which can be specified by the **-t** flag of the **mkvg** command, that allows you to increase the maximum number of PPs per disk proportional to the given integer multiplier value. However, the maximum number of PVs is adversely affected by this multiplier and decreases proportionally to the specified factor. AIX Version 4.3.2 expanded the LVM scalability by introducing big volume

groups. A big VG can have up to 128 physical volumes and a maximum of 512 logical volumes defined with it.

AIX 5L Version 5.3 takes the LVM scalability to the next level and offers a new *scalable volume group (scalable VG)* type. A scalable VG can accommodate a maximum of 1024 PVs and raises the limit for the number of LVs to 4096. The maximum number of PPs is no longer defined on a per disk basis, but applies to the entire VG. This opens up the prospect of configuring VGs with a relatively small number of disks, but fine-grained storage allocation options through a large number of PPs which are small in size. The scalable VG can hold up to 2097152 (2048 K) PPs. Optimally, the size of a physical partition can also be configured for a scalable VG. As with the older VG types the size is specified in units of megabytes and the size variable must be equal to a power of 2. The range of PP sizes starts at 1 (1 MB) and goes up to 131072 (128 GB), which is more than two orders of magnitude above the 1024 (1 GB) maximum for AIX 5L Version 5.2. (The new maximum PP size provides an architectural support for 256 petabyte disks.) Table 3-1 shows the variation of configuration limits with the different VG types. Note that the maximum number of user definable LVs is given by the maximum number of LVs per VG minus 1 because one LV is reserved for system use. Consequently, System administrators can configure 255 LVs in normal VGs, 511 in big VGs, and 4095 in scalable VGs.

Table 3-1 Configuration limits for volume groups

VG type	Maximum PVs	Maximum LVs	Maximum PPs per VG	Maximum PP size
Normal VG	32	256	32512 (1016 * 32)	1 GB
Big VG	128	512	130048 (1016 * 128)	1 GB
Scalable VG	1024	4096	2097152	128 GB

Metadata structure for scalable VGs

In AIX the first 128 sectors of a PV are reserved to store various types of information which is indispensable for proper device integration and usage. The physical sector number (PSN) layout for hard disk drives is defined by the /usr/include/sys/hd_psn.h header file. AIX 5L Version 5.3 adds the new global variable PSN_SVG_MWC_REC that defines the PSN of the first MWC cache record for scalable VGs to start with sector 100.

There are no other changes to the PSN layout so you still find the LVM record to start at sector 7 on the disk. However, the LVM record for a scalable VG differs substantial in format and content from the LVM record for a standard or big VG. All LVM record structures are defined in the /usr/include/lvmrec.h file.

Like in previous AIX releases the VGSA and the VGDA areas follow after the first 128 reserved sectors, but the structure of both entities has changed considerably.

The VGSA for scalable VGs consists of three areas: PV missing area (PVMA), MWC dirty bit area (MWC_DBA), and PP status area (PPSA). The overall size reserved for the VGSA is independent of the configuration parameters of the scalable VG and stays constant. However, the size of the contained PPSA changes proportional to the configured maximum number of PPs:

PV missing areas PVMA tracks if any of the disks are missing.

MWC dirty bit area MWC_DBA holds the status for each LV if passive mirror write consistence is used.

PP status area PPSA logs any stale PPs.

The new type VGDA area starts at sector number 650 and consists of 5 different areas. Again, while the overall size reserved for the VGDA stays constant, the sizes of the contained areas are variable in proportion to the configured maximum number of PPs and LVs for the scalable VG:

VG information area Holds VG-related metadata such as the current number of LVs, the allowed maximum number of LVs, the number of PVs, the allowed maximum number of PVs, and other information of significance for a VG.

PV information area Keeps metadata related to PVs, such as the state of every PV which belongs to a scalable VG.

LVCB information area Maintains a record of logical volume control block (LVCB) metadata for each LV defined on a scalable VG but which is only related to the file system space.

LV entry area Stores the LVM-related metadata such as strictness, inter- and intra-policy for each LV in a scalable VG.

PP entry area Records information pertinent to physical partitions, such as their state.

Note: The LVCB contains metadata about a logical volume. For standard VGs the LVCB resides in the first block of the user data within the LV. Big VGs keep additional LVCB information in the ondisk VGDA. The LVCB structure on the first LV user block and the LVCB structure within the VGDA are similar but not identical. (If a big VG was created with the **-T 0** option of the **mkvg** command, no LVCB will occupy the first block of the LV.) With scalable VGs, logical volume control information is no longer stored on the first user block of any LV. All relevant logical volume control information is kept in the VGDA as part of the LVCB information area and the LV entry area. Therefore, no precautions have to be met when using raw logical volumes because there is no longer a need to preserve the information held by the first 512 bytes of the logical device.

Command interface changes

The scalable volume group implementation in AIX 5L Version 5.3 provides configuration flexibility with respect to the number of physical and logical volumes that can be accommodated by a given instance of the new volume group type. The configuration options allow any scalable VG to contain 32, 64, 128, 256, 512, 768, or 1024 disks and 256, 512, 1024, 2048, or 4096 LVs. The maximum values of 1024 PVs and 4096 LVs do not need to be configured at the time of VG creation to account for future growth. The initial settings can always be increased at a later date as required.

The following user commands were changed in support for the new scalable volume group type: **chvg**, **lsvg**, and **mkvg**.

mkvg command changes

The enhanced **mkvg** command implementation provides four new command flags, -S, -v, -P, and -I and three flags,-t, -L, and -s that are modified to adjust for the option to create a scalable VG. The following example shows the syntax statement of the **mkvg** command manual page and describes the new and modified flags in more detail.

```
...
mkvg Command

Purpose

Creates a volume group.

Syntax

mkvg [ -d MaximumPhysicalVolumes ] [ -B ] [ -t factor ] [ -S [ -v
LogicalVolumes ] [ -P Partitions ] ] [ -C ] [ -G ] [ -f ] [ -i ] [ -I ][ -c ]
```

```
[ -x ] [ -L LTGSize ] [ -m MaxPvSize ] [ -n ] [ -s Size ] [ -V MajorNumber ] [
-y VolumeGroup ] PhysicalVolume ...
...
```

-S The -S flag creates a scalable-type volume group. By default, this volume group can accommodate up to 1024 physical volumes, 256 logical volumes, and 32768 physical partitions. To increase the number of logical volumes, use the -v option. To increase the number of physical partitions, use the -P option.

Note: Increasing the maximum number of logical volumes and the number of physical partitions beyond the default values for a scalable volume group can significantly increase the size of the VGDA proportionately. These values should only be increased as needed because they cannot be decreased. Meanwhile, as the VGDA space increases, all VGDA update operations (creating a logical volume, changing a logical volume, adding a physical volume, and so on) can take longer to run.

-v The -v flag defines the number of logical volumes that can be created. The corresponding value is specified by the logical volumes variable which directly follows the flag. Valid values are 256, 512, 1024, 2048 and 4096. The default is 256. The **chvg** command can be used to increase the number of logical volumes up to the maximum of 4096. This option is only valid with the -S option.

-P The -P flag defines the total number of partitions in a scalable volume group. The corresponding value is specified by the partition variable that directly follows the flag. The partitions variable is represented in units of 1024 partitions. Valid values are 32, 64, 128, 256, 512 768, 1024, and 2048. The default is 32 K (32768 partitions). The **chvg** command can be used to increase the number of partitions up to the maximum of 2048 K (2097152 partitions). This option is only valid with the -S option.

-I The -I flag creates a volume group that can be imported to AIX 5L Version 5.1 and 5.2. The logical track group (LTG) size will behave as if the volume group had been created prior to AIX 5L Version 5.3. If the logical volumes are later created with a stripe width that is larger than the supported stripe size on AIX 5L Version 5.1 or 5.2, then attempting to import the volume group back to AIX 5L Version 5.1 or 5.2 is not supported.

-t The -t flag changes the limit of the number of physical partitions per physical volume for standard and big volume groups. The flag is directly followed by the factor value. The concept to configure the number of PPs

per disk does not ably to scalable volume groups and consequently the -t option is ignored with the -S option.

-L For volume groups created on AIX 5L Version 5.3 without the -I flag, the -L flag is ignored. When the volume group is varied on the logical track group size will be set to the common max transfer size of the disks.

For volume groups created on Version 5.3 with the -I flag or for volume groups created prior to Version 5.3, the -L flag is immediately followed by the logical track group size value which is expressed in units of kilobytes, and which must be 128, 256, 512, or 1024. In addition, the LTG value should be less than or equal to the maximum transfer size of all disks in the volume group. The default LTG size is 128 kilobytes.

-s Sets the number of megabytes in each physical partition. The -s flag is immediately followed by the size variable. The size is expressed in units of megabytes from 1 (1 MB) through 131072 (128 GB) for scalable VGs. The maximum supported PP size for standard and big VGs stays unchanged at 1024 MB. The size variable must be equal to a power of 2 (for example 1, 2, 4, 8, and so on). The default value for 32 and 128 PV volume groups will be the lowest value to remain within the limitation of 1016 physical partitions per PV. The default value for scalable volume groups will be the lowest value to accommodate 2040 physical partitions per PV.

chvg command changes

The enhanced **chvg** command implementation provides four new command flags, -G, -P, -v, and -I; and two flags, -L, and -t that were modified to adjust for the option to change a scalable VG. The new flags allow you to convert standard and big VGs to scalable VGs and to increase the maximum number of physical partitions and logical volumes for existing scalable VGs if desired.

The conversion of a previously configured standard or big VG to a scalable VG type requires the volume group to be varied off (with the **varyoffvg** command).

This is different compared to the conversion of a standard VG to a big VG, which can be carried out without taking the source VG offline. The **chvg** command returns the following error message if you make an attempt to convert to a scalable VG while the source VG is still in an active state:

```
# chvg -G testvg
0516-1707 chvg: The volume group must be varied off during conversion to
        scalable volume group format.
0516-732 chvg: Unable to change volume group testvg.
```

This behavior accounts for the substantial transformations which are necessary to convert the source VG's metadata to the new scalable VG metadata format.

The new -P and -v flags of the **chvg** command provide the analogous configuration options with the same set and range of values as the -P and -v flags of the **mkvg** command. The -t flag of the **chvg** command will be ignored for scalable VGs, as is the case with the **mkvg** command. The -L flag has no effect on volume groups created on AIX 5L Version 5.3 but allows the LTG size to change on standard and big VGs which were created on previous AIX releases. The -L flag combined with a flag value of 0 can be used to enable the variable LTG option on VGs which were created prior to Version 5.3. Refer to 3.1.6, "Variable logical track group" on page 111 for more information about this particular option.

The new -G and -I flag of the **chvg** command offer the following functions:

-G Changes the volume group to scalable VG format. The conversion of a standard or big VG to a scalable VG type requires the volume group to be varied off (with the **varyoffvg** command).

Note:

- The -G flag cannot be used if there are any stale physical partitions.
- Once the volume group is converted, it cannot be imported into AIX 5L Version 5.2 or previous versions.
- The -G flag cannot be used if the volume group is varied on in concurrent mode.
- There must be enough free partitions available on each physical volume for the VGDA expansion for this operation to be successful.
- Since the VGDA resides on the edge of the disk and it requires contiguous space for expansion, the free partitions are required on the edge of the disk. If those partitions are allocated for user usage, they will be migrated to other free partitions on the same disk. The rest of the physical partitions will be renumbered to reflect the loss of the partitions for VGDA usage. This will change the mappings of the logical to physical partitions in all the PVs of this VG. If you have saved the mappings of the LVs for a potential recovery operation, you should 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 using those maps, the restore operation may fail since the partition number may no longer exist (due to reduction). It is recommended that backup is taken before the conversion, and right after the conversion if the map option is utilized.
- Because the VGDA space has been increased substantially, every VGDA update operation (creating a logical volume, changing a logical volume, adding a physical volume, and so on) may take considerably longer to run.

-I The -I flag modifies the volume group so that it can be imported to AIX 5L Version 5.1 and 5.2. The LTG size will behave as if the volume group had been created prior to AIX 5L Version 5.3. This operation might fail if the volume group contains striped logical volumes whose stripe size is larger than the supported stripe size on AIX 5L Version 5.1 or 5.2. If logical volumes are later created with a stripe width that is larger than the supported stripe width on AIX 5L Version 5.1 or 5.2, then attempting to import the volume group back to AIX 5L Version 5.1 or 5.2 is not supported. The -I flag modifies the specified standard or big VG to no longer allow the LVM to configure the optimum LTG size dynamically during the varyon process. The -I flag will not work for scalable VGs.

lsvg command changes

The **lsvg** command output on AIX 5L Version 5.3 has changed to show the maximum physical partitions per volume group for all volume group types. The corresponding number is listed as the value for the MAX PPs per VG field. The **lsvg** output for scalable VGs will not display the maximum number of PPs per PV because for the scalable VG type the limit for PPs is defined per VG and not per PV.

SMIT and graphical user interface changes

The new scalable volume group is fully supported by the System Management Interface Tool (SMIT) and the Web-based System Manager graphical user interface.

Existing SMIT panels which are related to VG management tasks are changed and many new panels are added to account for the new scalable VG type. For example, you can use the new SMIT fast path _mksvg to directly access the Add a Scalable Volume Group SMIT menu. Figure 3-1 on page 105 depicts the appearance of the new menu when invoked by the **smitty _mksvg** command.

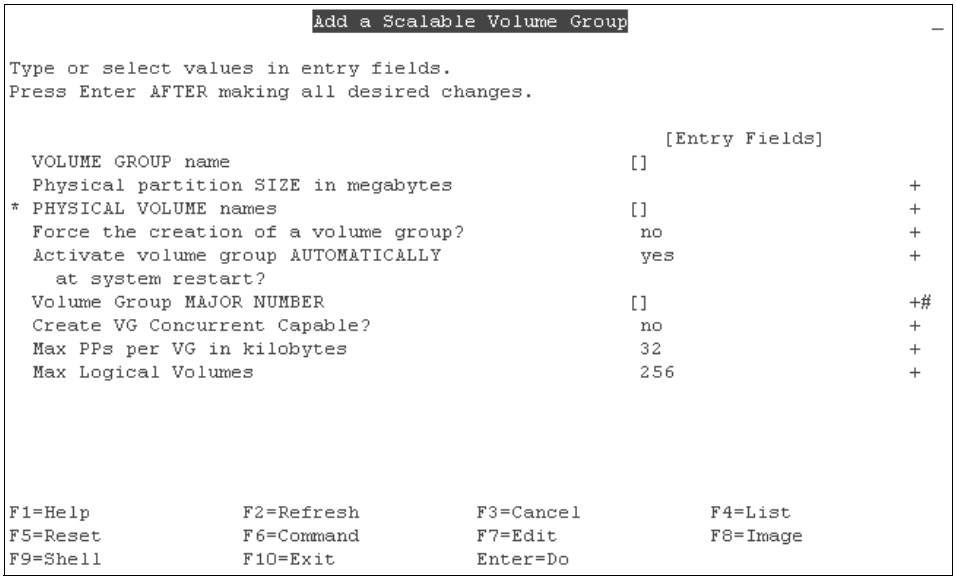

Figure 3-1 New SMIT menu: Add a Scalable Volume Group

AIX 5L Version 5.3 implements changes to the Volume Group container of the Web-based System Manager to account for the new VG type.

Like in the previous AIX release, the wizard that supports the creation of new VGs is dedicated to the standard VG type, but with Version 5.3 additional usage information regarding the available VG types is provided and you are referred to the advanced method for big VG or scalable VG creation.

The new task guide of the advanced VG creation method now offers the additional option to create a scalable VG. You can activate the task guide by selecting **Volumes** → **New** → **Volume Group (Advanced Method)...** in the menu bar of the Volume Group container. Figure 3-2 on page 106 shows the new inserted top-level panel that allows you to choose the type of VG you want to create. Other Web-based System Manager dialogs impacted by the introduction of the scalable VG type are related to the Volume Group Properties panel.

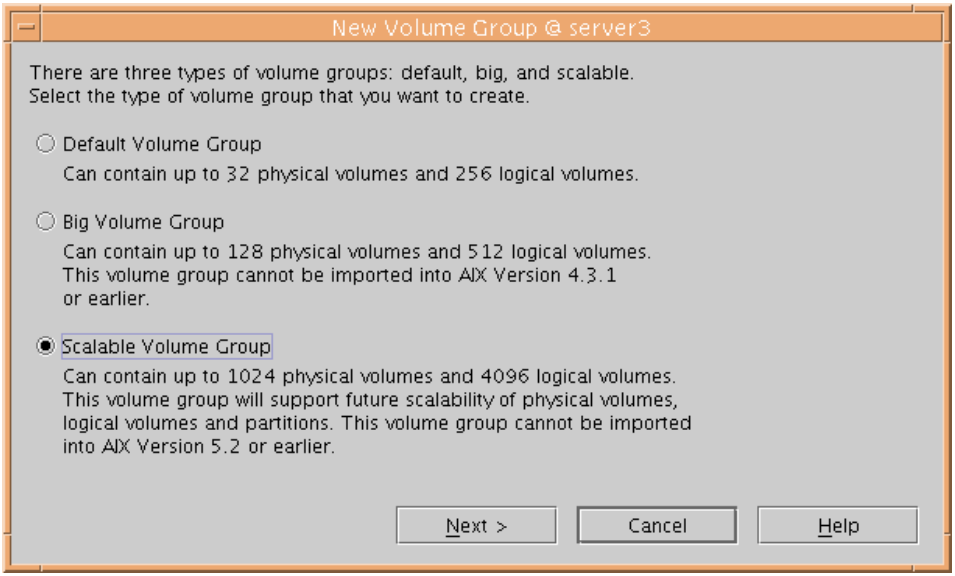

Figure 3-2 New Web-based System Manager panel: New Volume Group

3.1.4 Striped column support for logical volumes

Redundant Array of Independent Disks (RAID) is a term used to describe the technique of improving data availability and data I/O rates through the use of arrays of disks and various data-striping methodologies. RAID algorithms can be implemented as part of the operating system's file system software, or as part of a disk device driver.

AIX LVM supports the following RAID options:

RAID 0 Striping

RAID 1 Mirroring

RAID 10 or 0+1 Mirroring and striping

AIX 5L Version 5.3 further enhances the LVM RAID 0 and the LVM RAID 10 implementation and provides *striped columns* support for logical volumes. This new feature allows to extend a striped logical volume even if one of the physical volumes in the disk array became full.

In previous AIX releases you could enlarge the size of a striped logical volume with the **`extendlv`** command as long as enough physical partitions were available within the group of disks which define the RAID disk array. Rebuilding the entire LV was the only way to expand a striped logical volume beyond the hard limits imposed by the disk capacities. This work-around required to backup and delete

the striped LV and then to recreate the LV with a larger stripe width followed by a restore operation of the LV data.

To overcome the disadvantages of this time-consuming procedure, AIX 5L Version 5.3 introduces the concept of striped columns for LVs.

Prior to Version 5.3 the stripe width of a striped LV was determined at the time of LV creation by either of the following two methods:

- Direct specification of all physical volume names
- Specification of the maximum number of physical volumes allocated to the striped logical volume

The maximum number of PVs allocated to the striped LV is determined by the upper bound value specified with the -u flag of the **mklv** command. If you use the -u flag and also directly specify the PV names, the later takes precedence and the -u flag will be ignored. It was not permitted to configure a striped logical volume with an upper bound larger than the stripe width.

In AIX 5L Version 5.3 the upper bound can be a multiple of the stripe width. One set of disks, as determined by the stripe width, can be considered as one striped column. Note that the upper bound value is not related to the number of mirror copies in case you are using a RAID 10 configuration. Figure 3-3 illustrates the new concept of striped columns for a logical volume with stripe width 3 and an upper bound of 6.

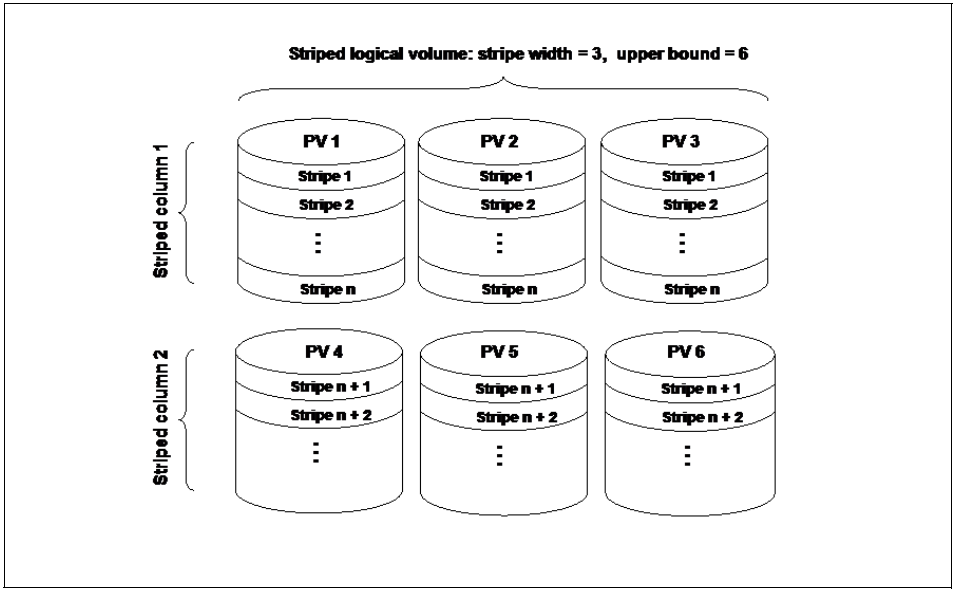

Figure 3-3 Logical volume with a stripe width of 3 and 2 striped columns

If you use the **extendlv** command to extend a striped logical volume beyond the physical limits of the first striped column, an entire new set of disks will be used to fulfill the allocation request for additional logical partitions. If you further expand the LV more striped columns may get added as required and as long as you stay within the upper bound limit. The **chlv -u** command allows you to increase the upper bound limit to provide additional headroom for striped LV expansion. The -u flag of the enhanced **extendlv** command can be used to raise the upper bound and extend the LV all in one operation.

Command interface changes

The following commands are affected by the introduction of the new striped column feature in AIX 5L Version 5.3:

- **mklv**
- **chlv**
- **extendlv**
- **mklvcopy**

The **mklv** command supports the striped columns concept for logical volumes with a new -C flag and changed characteristics for the existing -u flag. The following paragraph shows the usage information of the **mkvg** command as it is returned when entered without any flag or parameter:

```
# mklv
0516-606 mklv: Volume group name not entered.
Usage: mklv [-a IntraPolicy] [-b BadBlocks] [-c Copies]
        [-d Schedule] [-e InterPolicy] [-i] [-L Label] [-m MapFile]
        [-r Relocate] [-s Strict][-t Type] [-u UpperBound]
        [-v Verify] [-x MaxLPs] [-y LVname] [-S StripeSize] [-Y Prefix]
        [-o Overlapping IO] [-C StripeWidth] [-T IOoption] VGname NumberOfLPs
[PVname...]
Makes a logical volume.
```

The new -C flag directly defines the intended stripe width for the LV. The -u flag continues to be used to specify the maximum number of physical volumes for new allocation, but beginning with AIX 5L Version 5.3 the given upper bound value can be larger than the stripe width. If the striped LV has to be extended to the next column a full new stripe width of disks will be used; for that reason, the upper bound value must be specified in multiples of the stripe width. The following list describes the new -C command flag and the modified -u flag in more detail.

-C Stripe_width Sets the Stripe width of the logical volume. If the Stripe_width variable value is not entered it is assumed to

be the UpperBound variable value or the total number of disks specified on the command line.

-u UpperBound Sets the maximum number of physical volumes for new allocation. The value of the UpperBound variable should be between one and the total number of physical volumes. When using super strictness, the upper bound indicates the maximum number of physical volumes allowed for each mirror copy. When using striped logical volumes, the UpperBound value must be a multiple of the Stripe_width value. If upper bound is not specified, it is assumed to be Stripe_width for striped logical volumes.

As mentioned previously the -u flag of the **chlv**, **extendlv**, and **mklvcopy** commands will now allow you to change the upper bound to be a multiple of the stripe width. The **extendlv -u** command can be used to change the upper bound and to extend the LV in a single operation.

3.1.5 Volume group pbuf pools

The Logical Volume Manager (LVM) uses a construct named *pbuf* to control a pending disk I/O. Pbufs are pinned memory buffers and one pbuf always is used for each individual I/O request, regardless of the amount of data that is supposed to be transferred. AIX creates extra pbufs when a new physical volume is added to a volume group.

In previous AIX releases, the pbuf pool was a system-wide resource, but starting with AIX 5L Version 5.3 the LVM assigns and manages one pbuf pool per volume group. This enhancement supports advanced scalability and performance for systems with a large number of VGs and applies to all VG types. As a consequence of the new pbuf pool implementation, AIX displays and manages additional LVM statistics and tuning parameters. AIX 5L Version 5.3 introduces the **lvmo** command to provide support for the new pbuf pool-related administrative tasks.

The **lvmo** command can be used to display pbuf and blocked I/O statistics and the settings for pbuf tunables regardless of whether the scope of the entity is system-wide or VG-specific. However, the **lvmo** command only allows the settings to change for the LVM pbuf tunables that are dedicated to specific VGs. The sole pbuf tunable with system-wide scope continues to be managed by the **ioo** command. Also, the system wide number of I/Os that were blocked due to lack of free pbufs can still be displayed by the **vmstat -v** command like in AIX releases prior to Version 5.3.

Use **lvmo -?** to display the command usage statement:

```
# lvmo -?
lvmo: Not a recognized flag: ?
lvmo -v VgName -o Tunable [ =NewValue ]
```

The **lvmo** command displays and tunes the following items for the VG specified by the -v flag:

pv_pbuf_count Number of pbufs added when the specified VG is extended by one PV. This parameter is tunable with the **lvmo** command and takes effect immediately at run time. The pv_pbuf_count default values are 256 for the 32-bit kernel and 512 for the 64-bit kernel. The **lvmo** command performs no range checking.

total_vg_pbufs Number of pbufs currently available for the specified VG. This parameter is tunable with the **lvmo** command and takes effect after the VG has been varied off and varied on again.

max_vg_pbuf_count Maximum number of pbufs that can be allocated for the specified VG. This parameter is tunable with the **lvmo** command and takes effect after the VG has been varied off and varied on again.

pervg_blocked_io_count Number of I/Os that were blocked due to lack of free pbufs for the specified VG. This parameter provides statistics information about congestion for the VG pbuf pool and can serve as an indicator for required adjustments to the pv_pbuf_count and total_vg_pbufs tunables. This counter will be reset whenever the VG is varied on.

Note: The pbuf pools are not necessarily incorrectly sized just because LVM uses up all of its pbufs and as a result I/Os are blocked due to the lack of free pbufs. As a rule of thumb you can say that having an LVM pbuf pool resource which is larger than 2 or 3 times the queue depth of the underlying devices may provide little if any increase in overall throughput. A pbuf resource that is too large could adversely affect the performance of other VG on the same adapters.

The **lvmo** command displays the following system-wide items:

global_pbuf_count Minimum number of pbufs that are added when a PV is added to any VG. This system-wide parameter is tunable

with the **ioo** command but not with the **lvmo** command. Changes to the parameter take effect for any VG that has been varied off and varied on again after the new value was configured through the **ioo** command. Note that the global_pbuf_count parameter label of the **lvmo** command maps to the pv_min_pbuf parameter label of the **ioo** command. The global_pbuf_count default values are the same as for the pv_pbuf_count parameter: 256 for the 32-bit kernel and 512 for the 64-bit kernel.

global_blocked_io_count System-wide number of I/Os that were blocked due to lack of free pbufs. This parameter provides accumulated statistics on congestion for the combined pbuf pool resources on your system and can serve as an indicator for required adjustments to the global_pbuf_count, pv_pbuf_count, and total_vg_pbufs tunables. The global_blocked_io_count parameter value is also displayed by the **vmstat -v** command. The counter is reset at system startup.

If both the VG-specific pv_pbuf_count and the system-wide global_pbuf_count are configured, the larger value takes precedence over the smaller.

Use the **lvmo -a** command to display the current value of the pbufs statistics and tuning parameters. The following shows an example output for the rootvg of a given system:

```
# lvmo -a
vgname = rootvg
pv_pbuf_count = 512
total_vg_pbufs = 512
max_vg_pbuf_count = 16384
pervg_blocked_io_count = 1
global_pbuf_count = 512
global_blocked_io_count = 1
```

3.1.6 Variable logical track group

When the Logical Volume Manager receives a request for an I/O it breaks the I/O down into what is called logical track group (LTG) sizes before it passes the request down to the device driver of the underlying disks. The LTG is the maximum transfer size of a logical volume and is common to all the logical volumes in the volume group since it is a volume group attribute.

AIX 5L Version 5.2 accepted LTG values of 128 KB, 256 KB, 512 KB, and 1024 KB. However, many disks now support transfer sizes larger than 1 MB. To

take advantage of these larger transfer sizes and get better disk I/O performance AIX 5L now accepts values of 128 KB, 256 KB, 512 KB, 1 MB, 2 MB, 4 MB, 8 MB, and 16 MB for the LTG size.

In contrast to previous releases AIX 5L Version 5.3 also allows the stripe size of a Logical Volume to be larger than the LTG size in use and expands the range of valid stripe sizes significantly. Version 5.3 adds support for 2 MB, 4 MB, 8 MB, 16 MB, 32 MB, 64 MB, and 128 MB stripe sizes to complement the 4 KB, 8 KB, 16 KB, 32 KB, 64 KB, 128 KB, 256 KB, 512 KB, and 1 MB stripe size options of the former release.

In AIX 5L Version 5.2 the LTG size was set by the -L flag on the **chvg** or **mkvg** command. In AIX 5L Version 5.3 it is set by **varyonvg** using the flag -M. The LTG size thus created is called the variable logical track group size.

The following command sets the LTG size of the tmpvg volume group at 512 KB:

```
# varyonvg -M512K tmpvg
```

The LTG size is specified either in K or M units implying KB or MB respectively.

When the LTG size is set using the -M flag, the **varyonvg** and **extendvg** commands may fail if an underlying disk has a maximum transfer size that is smaller than the LTG size.

If the -M flag is not used, the **varyonvg** command will select the optimal LTG size automatically. This optimal size is the largest common maximum transfer size among the underlying disks. If an **extendvg** or **reducevg** command is subsequently executed and the optimal LTG size changes, the LVM driver will correct the LTG size on the fly. That is, it will hold the I/O, modify the LTG size, and then resume.

The following command will enable variable LTG for a tmpvg volume group created prior to AIX 5L Version 5.3 at the next **varyonvg** and will set the logical track group size to the common max transfer size of the disks:

```
# chvg -L 0 tmpvg
```

If this command is not executed a volume group created prior to AIX 5L Version 5.3 will have the old LTG size in Version 5.3 and this LTG size will behave the same way it did in the prior release.

You can specify 128, 256, 512, or 1024 instead of 0 on the command line to indicate 128 KB, 256 KB, 512 KB, or 1024 KB LTG size, respectively. The value should be less than or equal to the maximum transfer size of all disks in the volume group. The default size is 128 kilobytes.

By default, volume groups in AIX 5L Version 5.3 are created with variable logical track group size. If you want to import it to a previous release of AIX, you first need to disable the variable LTG using the -I option for **mkvg** or **chvg** (then do a **varyoffvg** followed by **exportvg**), otherwise the **importvg** command on the previous release will fail.

The -L flag on the **mkvg** or **chvg** command is ignored for volume groups created in AIX 5L Version 5.3. A message will be printed that says **varyonvg** will determine LTG size based on the maximum transfer size of the disks at varyon time.

To obtain what the maximum supported LTG size of your hard disk is, you can use the **lquerypv** command with the -M flag. The output gives the LTG size in KB, as can be seen from the following lines:

```
# /usr/sbin/lquerypv -M hdisk0
256
```

The **lspv** command will display the same value as MAX REQUEST, as shown in Example 3-1.

Example 3-1 The value of LTG size is shown as MAX REQUEST

```
# lspv hdisk0
PHYSICAL VOLUME:    hdisk0                  VOLUME GROUP:     rootvg
PV IDENTIFIER:      000bc6fdbff92812 VG IDENTIFIER
000bc6fd00004c00000000fda
469279d
PV STATE:           active
STALE PARTITIONS:   0                       ALLOCATABLE:      yes
PP SIZE:            16 megabyte(s)          LOGICAL VOLUMES:  9
TOTAL PPs:          542 (8672 megabytes)    VG DESCRIPTORS:   2
FREE PPs:           431 (6896 megabytes)    HOT SPARE:        no
USED PPs:           111 (1776 megabytes)    MAX REQUEST:      256 kilobytes
FREE DISTRIBUTION:  108..76..30..108..109
USED DISTRIBUTION:  01..32..78..00..00
```

You can list the value of the LTG in use with the **lsvg** command shown in Example 3-2.

Example 3-2 The output shows the value of LTG in use.

```
# lsvg rootvg
VOLUME GROUP:       rootvg                  VG IDENTIFIER:
000bc6fd00004c00000000fda469279d
VG STATE:           active                  PP SIZE:        16 megabyte(s)
VG PERMISSION:      read/write              TOTAL PPs:      542 (8672
megabytes)
MAX LVs:            256                     FREE PPs:       431 (6896
megabytes)
```

```
LVs:                9                   USED PPs:       111 (1776
megabytes)
OPEN LVs:           8                   QUORUM:         2
TOTAL PVs:          1                   VG DESCRIPTORS: 2
STALE PVs:          0                   STALE PPs:      0
ACTIVE PVs:         1                   AUTO ON:        yes
MAX PPs per VG:     32512
MAX PPs per PV:     1016                MAX PVs:        32
LTG size (Dynamic): 256 kilobyte(s)     AUTO SYNC:      no
HOT SPARE:          no                  BB POLICY:      relocatable
```

Note: The LTG size for a volume group created in AIX 5L Version 5.3 will be displayed as Dynamic in the **lsvg** command output as shown in Example 3-2.

3.2 JFS2 enhancements

The following sections describe JFS2 enhancements found in AIX 5L Version 5.3.

3.2.1 Disk quotas support for JFS2

AIX 5L introduced the Enhanced Journaled File System (JFS2), which offers unprecedented scalability and performance characteristics. AIX 5L Version 5.3 further expands the JFS2 function and provides a mechanism to control usage of persistent storage by implementing disk usage quotas.

Persistent storage limits, from here on referred to as *quotas*, may be set for individual users or groups on a per file system basis. The quota system will issue a warning to the user when a particular quota is exceeded, but allow some extra space for current work. Remaining over quota beyond a specified grace period will result in further allocation attempts being denied until the total usage is reduced below the user's or group's quota.

To reduce quota information overhead and system administration time the JFS2 quota implementation introduces the concept of *Limits Classes,* which is explained in more detail in "Concepts and implementation specifics" on page 115. Also, in contrast to the JFS disk quota system, AIX 5L Version 5.3 offers extensive configuration support through a comprehensive set of new System Management Interface (SMIT) menus.

Packaging and installation

All executables that are required to use the AIX disk quota system for JFS and JFS2 file system types are included in the bos.sysmgt.quota fileset. This fileset is available by default after any AIX Base Operating System (BOS) installation.

The **lslpp** output given in the following example shows the file content of the bos.sysmgt.quota fileset in AIX 5L Version 5.3:

```
# lslpp -f bos.sysmgt.quota
  Fileset               File
  ----------------------------------------------------------------------------
Path: /usr/lib/objrepos
  bos.sysmgt.quota 5.3.0.0
                        /usr/sbin/repquota
                        /usr/sbin/edquota
                        /usr/sbin/quot
                        /usr/sbin/quotacheck
                        /usr/sbin/quotaoff -> /usr/sbin/quotaon
                        /usr/sbin/quotaon
                        /usr/sbin/j2edlimit
                        /usr/sbin/quota
```

To support the new and enhanced user-level commands, JFS2-specific commands were added to the quotactl() subroutine which is included in the AIX standard C Library (libc.a). The standard C library is installed by default on any AIX system as part of the bos.rte.libc fileset.

Additional changes had to be made to the **chfs** command to enable quotas on JFS2 file systems.

Concepts and implementation specifics

The disk quota system in AIX is based on the functional description documented in *UNIX System Manager's Manual (4.3 Berkeley Software Distribution): Disk Quotas in a UNIX Environment*. This document can be accessed through the following URL:

http://www.openbsd.org

The JFS-specific implementation in previous AIX releases further extends the function by the support of distinct quotas for users and groups and the optional enforcement of hard quota limits. The JFS-specific disk quota system is managed by command-line utilities.

The new JFS2-specific disk quota system implementation in AIX 5L Version 5.3 additionally introduces the concept of Limits Classes. Limits classes allow the

configuration of per file system default limits and offer comprehensive support through dedicated SMIT panels.

Limits Classes

In most installations that configure disk usage quotas there is little variation in the limits placed on individual users.

There may be a small number of conceptual classes where all members of one given class have the same individual limits (note that this is different from group quotas, where the total space used by everyone in that group is limited), but for the most part the traditional JFS-specific implementation duplicates usage limits information in every quota record.

Duplicating this information makes it unreasonable to implement any sort of default limits scheme, especially one that allows such defaults to be changed. To eliminate this problem the new JFS2-specific disk quota system introduces Limits Classes which define a set of hard and soft disk block and file allocation limits, and the grace periods before the soft limit becomes enforced as the hard limit.

System administration then becomes a process of defining Limits Classes (using different sets for user and group limits) and assigning users or groups to a particular class. Members of a Limits Class are bound by the limits defined by the class; a user or group that is not a member of a specific class is automatically a member of and bound by the limits of a per file system *Default Class*. The Default Class is always present as soon as the JFS2 file system has been enabled for quotas. The system administrator, however, can change the limit settings for any Default Class. Deleting an assignment to a user-defined Limits Class can be accomplished by assigning the user or group in question back to the Default Class. Changing the limits for an entire class is simply a matter of changing the limits in the Limits Class; there is no need to propagate those changes to every member of the class as in the traditional JFS-specific quota implementation.

To implement the Limits Class concept, the traditional limits information records are replaced by two new record types: a *Limits Record* which contains the hard and soft limits and grace period information, and a *Usage Record* which contains the usage and warning information for a particular user or group.

Each Usage Record simply references a Limits Record, and users with common limits defined all reference a single Limits Record, thereby reducing quotas information overhead. This method also drastically reduces system administration time. By predefining and designating a Limits Record to be the file system default, the administrator does not need to create any records at all for users who are subject to those default limits. If a user does not have a Usage

Record defined at either quotacheck or resource allocation time one is automatically created which references the default limits.

AIX 5L Version 5.3 quota user commands

The section “Packaging and installation” on page 115 gives an overview of all components which have to be present to configure the disk quota system for JFS and JFS2 files systems. The following paragraphs provide an outline of the quota user commands in AIX 5L Version 5.3.

The **chfs** command is used to enable quotas on JFS or JFS2 file systems. For JFS file systems you are allowed to define alternate names and locations for the quota files that are used to store the quota records. The default names are quota.user and quota.group, and they are created in the root of the JFS file system as soon as you run the **quotacheck** or the appropriate **edquota** command.

When a JFS2 file system is quota enabled by the use of the **chfs** command, the default quota files are automatically created in the root of the JFS2 file system and the system administrator cannot change their names and locations.

Note: Do not configure user-defined quota file names and locations for a quota-enabled JFS file system that coincide with the default quota file names and locations of a quota-enabled JFS2 file system. Also, be aware of the fact that the quota files only get created if no existing instances can be found.

The behavior of the **chfs** command has also changed significantly with respect to file system quota activation.

If you use **chfs** to configure a JFS file system for quotas only the file system stanza in /etc/filesystems receives an additional entry for the quota attribute. After that the **quotacheck** command has to be used to update the quota files with the current disk usage information. The **quotaon** command must follow immediately to initiate the concurrent tracking of disk usage and the enforcement of the limits. Also, you have to apply the **quotacheck** command and the **quotaon** command in sequence after a JFS quota enabled file system was mounted to ensure the accuracy and currency of the disk usage information and the enforcement of the limits.

If you use **chfs** to configure a JFS2 file system for quotas the **quotacheck** command will be invoked by default, the quota system becomes automatically active, and the defined quota limits are enforced. Likewise, if you mount a JFS2 quota-enabled file system, no **quotacheck** and no **quotaon** command is required to activate the quota system in a current and consistent state.

In the presence of mounted quota-enabled JFS2 file systems, the disk usage statistics are always maintained, the quota system is active, and the quota limits

will be applied. The **quotaon** and **quotaoff** commands are only required to assert control over quota limit enforcement.

The **edquota** command edits user and group quotas that pertain to JFS file systems only. If you use this command no quota-enabled JFS2 file system will be listed in the temporary file that is opened by the **edquota** command and contains the disk quota limit settings.

The new **j2edlimit** command manages quota Limits Classes for JFS2 file systems and as such can be considered as the JFS2 counterpart of the **edquota** command. Note the important distinction between these two commands: **edquota** manages quota information from a user point of view, whereas **j2edlimit** manages Limits Classes from a file system point of view.

The **j2edlimit** command enables the root user to perform the following tasks:

- Edit quota Limits Classes
- List quota Limits Classes
- Set an Existing Limits Class as the Default Limits Class
- Assign a user or group to a Limits Class

The **j2edlimit** command manages quota Limits Classes for one JFS2 file system at a time. In case you need to duplicate the identical Group Limits Class and User Limits Class definitions to another JFS2 file system you can use the following procedure:

1. Use the **cp** command to copy the relevant quota files from the root of the source to the root of the target JFS2 file system. If you merely control user quota only, the quota.user file needs to be copied. An analogous statement holds true for group quota control, which means that you have to copy the quota.group file only. You will have to copy both quota files for a full quota system implementation.
2. Run the **chfs** command against the target JFS2 file system. The **chfs** command does not recreate or overwrite the existing copy of the quota files, but will call **quotacheck** to update the quota files usage records while keeping the Limits Class definitions.

The remaining quota-related executables of the bos.sysmgt.quota file set, **repquotas**, **quotacheck**, **quotaoff**, **quotaon**, and **quota** examine the file system type and either execute the current code for a JFS-type file system or execute a JFS2-specific utility. Any JFS2-specific flags used when a JFS file system is specified result in a warning message and are ignored; when specified with commands that operate on multiple/all file systems, the extraneous flags are ignored silently.

The changes and additions made to the user-level commands are enabled by new JFS2-specific commands for the quotactl() subroutine. The quotactl() subroutine manipulates disk quotas and is included in the AIX Standard C Library (libc.a).

File system centric SMIT interfaces

Previous AIX releases use a user/group centric approach to manage quotas. Selected file systems are enabled for quotas by the **chfs** command but the quota limits are defined by the **edquota** command for individual users or groups.

Since the JFS2 quota support in AIX 5L Version 5.3 introduces the concept of per file system Limits Classes, a file system centric approach to manage quotas is potentially more intuitive.

In AIX 5L Version 5.3 the JFS2 quota support provides SMIT panels for the file system centric approach which directly map to JFS2 quota management commands. Table 3-2 provides an overview of the changes and additions to the SMIT interface in support for the JFS2 disk quota system.

Table 3-2 File system centric SMIT panels for JFS2 disk quota support

SMIT panel title	Fast path	Type	Status
Enhanced Journaled File System	jfs2	Menu	modified
Add an Enhanced Journaled File System	crjfs2std	Dialog	modified
Add a JFS2 on a previously defined logical volume	crjfs2lvstd	Dialog	modified
Change / Show Characteristics of a JFS2	chjfs2	Dialog	modified
Manage Quotas for an Enhanced Journaled File System	j2fsquotas	Menu	added
Enable / Disable Quota Management	j2enablequotas	Dialog	added
Stop / Restart Quota Limits Enforcement	j2enforcequotas	Dialog	added
List Quota Usage	j2repquota	Dialog	added
Recalculate Current Disk Block and File Usage Statistics	j2quotacheck	Dialog	added
Add a Limits Class	j2addlimit	Dialog	added
Change / Show Characteristics of a Limits Class	j2changelimit	Dialog	added
Make a Limits Class the Default for a File System	j2defaultlimit	Dialog	added

SMIT panel title	Fast path	Type	Status
Assign a User or Group to a Limits Class	j2assignlimit	Dialog	added
List Limits Classes for a File System	j2listlimits	Dialog	added
Remove a Limits Class	j2removelimit	Dialog	added

One SMIT menu and three dialogs have been modified to support quotas on JFS2 file systems. But the new SMIT menu Manage Quotas for an Enhanced Journaled File System can be considered as the most important addition to the SMIT interface since it provides a single point of control to handle JFS2 quota tasks. Figure 3-4 shows the actual appearance of this new SMIT menu.

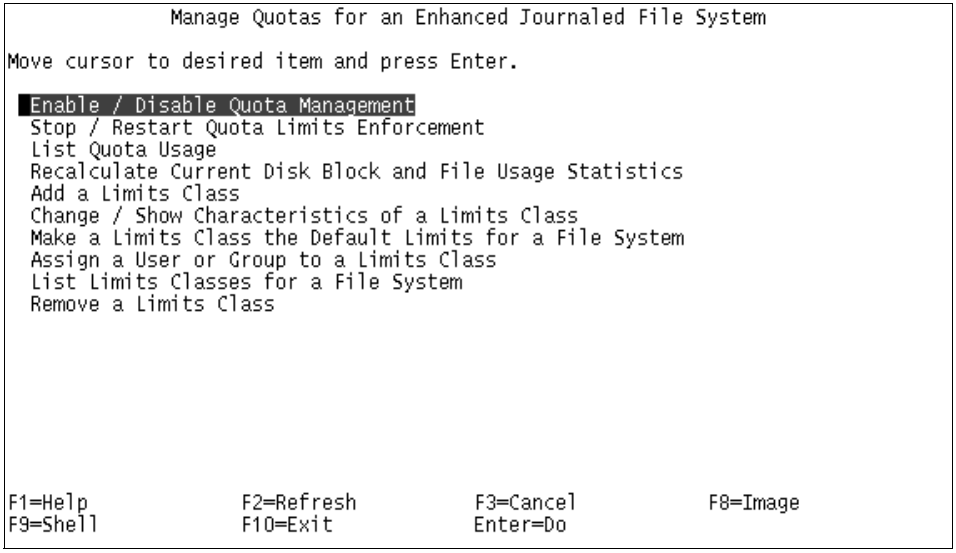

Figure 3-4 New SMIT menu: Manage Quotas for an Enhanced JFS

Setting up the Disk Quota System for JFS2

Depending on the specifics of a given system environment you may encounter the situation where it is required to configure the AIX disk quota system for a mixture of JFS and JFS2 file systems.

Consider the following scenario: The International Technical Support Organization (ITSO) is one of many departments of IBM that is concerned with technical support. Within the ITSO a number of different job roles exist. One of the ITSO employees holds the user ID aroell and works as an author on a given project.

The following steps provides you with some insight in the required procedure to establish a disk quota system on an IBM internal server for this particular scenario:

1. Log in with root authority.
2. Enable user and group quotas on the JFS2 /home file system and the mounted JFS /authors sample file system:

```
# chfs -a "quota = userquota,groupquota" /home
Initializing quota file /home/quota.user
Initializing quota file /home/quota.group
*** Checking user and group quotas for /dev/hd1 (/home)
root     fixed: inodes 0 -> 3   blocks 0 -> 64
bin      fixed: inodes 0 -> 1
guest    fixed: inodes 0 -> 1
aroell   fixed: inodes 0 -> 2   blocks 0 -> 4
sdutta   fixed: inodes 0 -> 2   blocks 0 -> 4
system   fixed: inodes 0 -> 3   blocks 0 -> 64
staff    fixed: inodes 0 -> 4   blocks 0 -> 8
bin      fixed: inodes 0 -> 1
usr      fixed: inodes 0 -> 1

# chfs -a "quota = userquota,groupquota" /authors
```

3. Use the **edquota** command to set the quota limits for the sample group ITSO (**edquota -g ITSO**) and the sample user aroell (**edquota -u aroell**) on the /authors JFS file system. The following example entries show the chosen quota limits:

```
Quotas for group ITSO:
/authors: blocks in use: 86324, limits (soft = 5242880, hard = 10485760)
        inodes in use: 167, limits (soft = 50000, hard = 100000)

Quotas for user aroell:
/authors: blocks in use: 86324, limits (soft = 51200, hard = 102400)
        inodes in use: 167, limits (soft = 500, hard = 1000)
```

   The block size is 1 KB and consequently the group ITSO has used about 84 MB of the maximum 5 GB of disk space. Of the maximum 50,000 files, users belonging to the ITSO group created just 167. The ITSO allows buffers of 5 GB of disk space and 50,000 files that can be allocated to temporary storage. The quotas for user aroell can be interpreted likewise.

4. Use the **j2edlimit -g /home** command to define a new *Group Limits Class* for the /home JFS2 file system. Within this example the new Group Limits Class defines the quota limits for all technical support departments within IBM. The + character in the ID field indicates that a new Group Limit Class shall be added. The related Group Limit Class ID will get automatically assigned by the system when you write your changes to the **j2edlimit** temporary file and leave the editing session. In our case the Group Limit

Class ID 1 is chosen. The following example entry lists the related quota limits:

```
Group Limits Classes for file system /home
Block Limits units: g=gigabyte, m=megabyte, or optional k=kilobyte
Grace Period units: d=days, h=hours, m=minutes, or s=seconds
Limits Class ID 0 is the default class.
Prepend '-' to ID to delete Limits Class.
Use '+' for ID to add new Limits Class.

       Block Limits               File Limits           Grace Period
ID     soft         hard          soft      hard        block    file

0      0            0             0         0           0        0
+      50g          1024g         50000     100000      4d       4d
```

5. Use the **j2edlimit** command to assign the ITSO sample group to the Group Limits Class ID 1:

   ```
   # j2edlimit -a 1 -g ITSO /home
   ```

   Because of the scope of the Group Limits Class ID 1 definition the system administrator could assign any other group of any given technical support department as required. This eliminates the necessity to configure and administer quota limits for each technical support department group individually. This is only a valid statement as long as the defined limits apply to every group in question.

6. Use the **j2edlimit -u /home** command to define a new *User Limits Class* for the /home JFS2 file system. Within this example the new User Limits Class defines the quota limits for all ITSO employees regardless of their specific job role. The related User Limit Class ID will get automatically assigned by the system when you write your changes to the **j2edlimit** temporary file and leave the editing session. In our case the User Limit Class ID 1 is chosen. The following example entry lists the related quota limits:

   ```
   User Limits Classes for file system /home
   Block Limits units: g=gigabyte, m=megabyte, or optional k=kilobyte
   Grace Period units: d=days, h=hours, m=minutes, or s=seconds
   Limits Class ID 0 is the default class.
   Prepend '-' to ID to delete Limits Class.
   Use '+' for ID to add new Limits Class.

          Block Limits               File Limits           Grace Period
   ID     soft         hard          soft      hard        block    file

   0      0            0             0         0           0        0
   +      51200        102400        500       1000        0        0
   ```

7. Use the **j2edlimit** command to assign the user aroell to the User Limits Class ID 1:

   ```
   # j2edlimit -a 1 -u aroell /home
   ```

   Because of the scope of the User Limits Class ID 1 definition the system administrator could assign any other user in the ITSO department as required. This eliminates the necessity to configure and administer quota limits for each ITSO employee individually. This is only a valid statement as long as the defined limits apply to every ITSO system user in question.

8. Enable the quota system with the **quotaon** command for all file systems:

   ```
   # quotaon -v -a
   ```

9. Use the **quotacheck** command to check the consistency of the quota files against actual disk usage:

   ```
   # quotacheck -v -a
   ```

10. Use the **repquota** command to view a summary of quotas and disk usage for all your file systems:

    ```
    # repquota -v -a
    ```

3.2.2 JFS2 file system shrink

Prior to Version 5.3 there is no way to shrink a file system dynamically while you are using it, although you can easily extend as needed. The procedure to shrink a file system was to create a new smaller version, copy the data, take the old version offline, then delete the old version. In Version 5.3 file system shrink is now available with JFS2.

Shrink a file system using SMIT

The easiest way to shrink a file system is through SMIT. Use SMIT chjfs2 fast path and choose the file system you want to shrink; you will see the menu shown in Figure 3-5 on page 124.

```
       Change / Show Characteristics of an Enhanced Journaled File System

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

                                                      [Entry Fields]
  File system name                                   /jfs2fs1
  NEW mount point                                    [/jfs2fs1]
  SIZE of file system
         Unit Size                                    512bytes                +
*        Number of units                              [4325376]                #
  Mount GROUP                                        []
  Mount AUTOMATICALLY at system restart?              no                      +
  PERMISSIONS                                         read/write              +
  Mount OPTIONS                                      []                       +
  Start Disk Accounting?                              no                      +
  Block Size (bytes)                                  4096
  Inline Log?                                         no
  Inline Log size (MBytes)                           [0]                       #
  Extended Attribute Format                          [v1]
  ENABLE Quota Management?                            no                      +

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

Figure 3-5 Change/Show Characteristics of an Enhanced Journaled File System

You can simply change the file system to your desired size in exactly the same way you extend it, but with the smaller number of units as shown in Figure 3-6 on page 126.

Shrink a file system using a command

User commands are available to shrink a file system. Use the **chfs** command with the following options:

```
chfs [ -n NodeName ] [ -m NewMountPoint ] [ -u MountGroup ]
   [ -A { yes | no } ] [ -p { ro | rw } ] [ -t { yes | no } ]
   [ -a Attribute=Value ] [ -d Attribute ] FileSystem
```

The enhanced options of the **chfs** command are discussed in the following sections.

-a size=NewSize

Specifies the size of the Enhanced Journaled File System in 512-byte blocks, megabytes or gigabytes. If NewSize has the M suffix, it is interpreted to be in Megabytes. If NewSize has a G suffix, it is interpreted to be in Gigabytes. If NewSize begins with a +, it is interpreted as a request to increase the file system size by the specified amount. If NewSize begins with a -, it is interpreted as a request to reduce the file system size by the specified amount.

If the specified size does not begin with a + or -, but it is greater or smaller than the file system current size, it is also a request to increase or reduce the file system size.

The inline log size remains unchanged if this file system new size is the same as the current file system size.

If the specified size is not evenly divisible by the physical partition size, it is rounded up to the closest number that is evenly divisible. The volume group in which the file system resides defines a maximum logical volume size and limits the file system size. When a request to reduce the file system size is successful, the logical volume should be equal to or smaller than the original LV size depending on the requested file system size.

-a logsize=LogSize

Specifies the size for an inline log in MB. Starting with Version 5.3, a + or - can be specified with LogSize. It is ignored if inLinelog not being used. It cannot be greater than 10% the size of the file system for an extending request, or smaller than 0.4% the size of the file system for a shrinking request.

Shrink an inline log

You can have an inline log in a Enhanced Journaled File system and possibly extend or shrink it. Here are the general guidelines:

- When new file system size is specified, but its inline log size is *not* specified, the new log size will be adjusted to be extended or shrunk proportionally, based on the specified extended/shrunk file system size. The log size increased or reduced should not more than 40% of the file system size increased or reduced.
- When a new file system size is not specified and there is an inline log, if a new logsize is specified the file system size might be changed to include the new log size.

Shrink a file system to minimum size

There is no command to show exactly how much a file system can be shrunk since the **df** command does not show the size of the metadata. In addition, the freed space reported by the **df** command is not necessarily the space that can be truncated by a shrink request due to file system fragmentation. A fragmented file system may not be shrunk if it does not have enough free space for an object to be moved out of the region to be truncated, and a shrink does not perform file system defragmentation. In this case, the **chfs** command should fail with the returned code 28. A way to work around this is shown in Figure 3-6 on page 126. Enter the **df** command to get information about the file system and look at the size of the Free field. That might be the maximum size (2324048 in this example)

you can shrink from the file system. Use the **chfs -a** command to shrink the file system and then check the final size with the **df** command. Remember that a file system can be extended or shrunk by a multiple of the physical partition size, which is 16 MB in this example.

```
# df
Filesystem    512-blocks      Free %Used    Iused %Iused Mounted on
/dev/hd4           65536     13240   80%     1644    11% /
/dev/hd2         2818048     29632   99%    26131     8% /usr
/dev/hd9var       131072     67656   49%      391     3% /var
/dev/hd3           65536     62992    4%       80     1% /tmp
/dev/hd1           32768     31640    4%       21     1% /home
/proc                  -         -    -        -     -  /proc
/dev/hd10opt       98304     21504   79%     1053     9% /opt
/dev/nimlv       6586368    153176   98%    24257     3% /export
/dev/hmcuser_lv     655360    634704    4%       18     1% /home/hmcuser
/dev/tem_lv      3276800   2500528   24%       18     1% /export/italien
/dev/jfs2lv1     4325376   2324048   47%        4     1% /jfs2fs1
#
# chfs -a size=-2324048 /jfs2fs1
Filesystem size changed to 2031616
#
# df
Filesystem    512-blocks      Free %Used    Iused %Iused Mounted on
/dev/hd4           65536     13240   80%     1644    11% /
/dev/hd2         2818048     29632   99%    26131     8% /usr
/dev/hd9var       131072     67648   49%      391     3% /var
/dev/hd3           65536     62992    4%       80     1% /tmp
/dev/hd1           32768     31640    4%       21     1% /home
/proc                  -         -    -        -     -  /proc
/dev/hd10opt       98304     21504   79%     1053     9% /opt
/dev/nimlv       6586368    153176   98%    24257     3% /export
/dev/hmcuser_lv     655360    634704    4%       18     1% /home/hmcuser
/dev/tem_lv      3276800   2500528   24%       18     1% /export/italien
/dev/jfs2lv1     2031616     30648   99%        4     1% /jfs2fs1
```

Figure 3-6 Shrink file system in command line

Note: Use the **df** command without -k, -m, -g, and the **chfs** command with a unit of 512-byte blocks to shrink a file system to minimum size in the previous example. Otherwise, the result can be different from what you expected.

Note that you cannot shrink a file system if the requested size is less than a physical partition size. If you attempt to reduce the size with less than a physical partition size, the request will be ignored, as shown in Figure 3-7 on page 127.

```
# df
Filesystem    512-blocks      Free %Used    Iused %Iused Mounted on
/dev/hd4           65536     13240   80%     1644    11% /
/dev/hd2         2818048     29632   99%    26131     8% /usr
/dev/hd9var       131072     67680   49%      391     3% /var
/dev/hd3           65536     62992    4%       80     1% /tmp
/dev/hd1           32768     31640    4%       21     1% /home
/proc                  -         -    -        -     -  /proc
/dev/hd10opt       98304     21504   79%     1053     9% /opt
/dev/nimlv       6586368    153176   98%    24257     3% /export
/dev/hmcuser_lv     65536     64704    4%       18     1% /home/hmcuser
/dev/tem_lv       3276800   2500528   24%       18     1% /export/italien
/dev/jfs2lv1       262144    261448    1%        4     1% /jfs2fs1
#
# chfs -a size=-1000 /jfs2fs1
# df
Filesystem    512-blocks      Free %Used    Iused %Iused Mounted on
/dev/hd4           65536     13240   80%     1644    11% /
/dev/hd2         2818048     29632   99%    26131     8% /usr
/dev/hd9var       131072     67680   49%      391     3% /var
/dev/hd3           65536     62992    4%       80     1% /tmp
/dev/hd1           32768     31640    4%       21     1% /home
/proc                  -         -    -        -     -  /proc
/dev/hd10opt       98304     21504   79%     1053     9% /opt
/dev/nimlv       6586368    153176   98%    24257     3% /export
/dev/hmcuser_lv     65536     64704    4%       18     1% /home/hmcuser
/dev/tem_lv       3276800   2500528   24%       18     1% /export/italien
/dev/jfs2lv1       262144    261448    1%        4     1% /jfs2fs1
# ^[21~_
```

Figure 3-7 File system unchanged: Attempt to reduce the size

Three internal steps to shrink file system

There are three steps to shrink a file system, internally provided by the command.

1. Validation to figure out if outside extents fit inside: The first step is to scan the allocation map (bMap) for allocated space outside a fence and to compare it to free space inside of the fence. If there is not enough free space, the command will fail.
2. Shrink the file system: Extents outside the fence are relocated to free spaces inside the fence.
3. Shrink LV: Finally, remove partitions in request.

Considerations

Table 3-3 provides a list of common questions and answers about this function.

Table 3-3 Things to know about file system shrink

When a file system is shrunk, will the logical volume that this file system resides on be automatically shrunk?	Yes
When a logical volume is created, and before the file system is defined for the logical volume, is there any way to shrink the logical volume only?	No
Can I shrink a standard JFS other than enhanced JFS (JFS2)?	No
Is it possible to shrink a an file system that has inline log in it?	Yes
Can I shrink an inline log while the size of the file system is unchanged?	Yes
During a shrink of the file system, are writes to the file system blocked?	Yes

Important: Shrinking a file system with snapshots is discouraged.

3.2.3 JFS2 logredo scalability

Logredo is the Journaled File System utility that applies all committed transactions recorded in the journal log since the most recent sync point. Its goal is to put the Journaled File System in the state it would have been in if all transactions had been applied, in order, by the Journaled File System itself.

The job of the logredo is accomplished in one pass over the log, reading backwards from log end to the first sync point encountered. This means that the log entries are read and processed in Last-In-First-Out (LIFO) order. In other words, the records logged latest in time are the first records processed.

AIX 5L Version 5.3 provides the following enhancements in the area of logredo to improve performance and to support large numbers of file systems:

- Support for minor numbers greater than 512 in a volume group
- Support for copy-on-write and cached updates to reduce I/O activity
- Support for shrink file system

Version 5.3 will allow a maximum of 4096 logical volumes in a volume group with the scalable volume group option. In addition, JFS2 logredo now can take advantage of copy-on-write and cached updates for performance reasons. Shrink file systems capability is included in Version 5.3 and is supported by the logredo utility.

Note: Logredo is called by the **fsck** and **mount** commands, and is not recommended to be run as a stand-alone utility unless completely necessary, and only by advanced users with a clear understanding of journaling in a file system.

3.2.4 JFS2 file system check scalability

The **fsck** command in AIX checks file system consistency and interactively repairs the file system if required. AIX 5L Version 5.3 enhanced the implementation of the helper which specifically performs the file system check for JFS2 file systems. The new code makes a better use of system resources and includes algorithms that improve the scalability and performance.

The **fsck** command, in particular, must read a large amount of data from the file system device. Access to this data and control of the memory used for buffering was greatly simplified by mapping the device that the file system resides on. As such the **fsck** command utilizes the new block device mapping capability introduced in AIX 5L Version 5.3, which is described in more detail in 2.5, “Block device mapping” on page 74.

3.2.5 JFS2 extended attributes Version 2 support

Extended attributes are an extension of the normal attributes of a file (such as size and mode). They are (name, value) pairs associated with a file or directory. The *name* of an attribute is a null-terminated string. The *value* is arbitrary data of any length. There are two types of extended attribute: extended attribute version 1 (EAv1) and extended attribute version 2 (EAv2). Starting with AIX 5L Version 5.3, EAv2 with JFS2 is now available. Table 3-4 summarizes the difference between EAv1 and EAv2. It should be noted that EAv2 is required to use an NFS4 ACL.

Table 3-4 Difference between EAv1 and EAv2

	EAv1	**EAv2**
No. of attributes	Max. 8 attributes	Architecturally unlimited
Size of attributes	4 KB per attribute	Max 16 TB per attribute
Name space	Only encoded name of 16 bits	String of max NAME_MAX length
User-defined attributes support	System-defined attributes only	Support both system-defined and user-defined attributes

Setting extended attribute version 2

If you created a file named report1 and want to set attributes to the file such as author, date, revision number, comments, and so on (SeonglulSon as author in this example), you can accomplish this with the **setea** command to set the value of an extended attribute and the **getea** command to read the value of an extended attribute, as shown in Example 3-3.

```
#setea -n Name { -v Value | -d | -f EAFile } FileName ...
#getea [-n Name] [-e RegExp] [-s] FileName
```

Example 3-3 Setting and reading extended attributes

```
#setea -n Author -v SeonglulSon report1
#getea report1
EAName: Author
EAValue:
SeonglulSon
```

In addition, you can specify a file name instead of value name with the **setea** command. Example 3-4 shows the command output.

Example 3-4 Configuring extended attributes with a file name

```
#getea report1
EAName: Author
EAValue:
SeonglulSon

#vi co-authors
Scott Vetter
Adrian Demeter
Armin Roell
Shiv Dutta
Seonglul Son

#setea -n Author -f co-authors report1
#getea report1
EAName: Author
EAValue:
Scott Vetter
Adrian Demeter
Armin Roell
Shiv Dutta
Seonglul Son
```

Compatibility

AIX 5L Version 5.3 continues to support EAv1 as the default format, and provides an option to create a file system with EAv2 and a runtime command to convert dynamically from EAv1 to EAv2 to create or access named attributes and advanced ACL. However, once a file system is created with EAv2 or conversion has been initiated, AIX 5L Version 5.2 cannot access the file system and attempting to mount will result in an EFORMAT error. The compatibility is summarized in Table 3-5.

Table 3-5 Compatibility of extended attributes

	JFS2 file system with EAv1	JFS2 file system with EAv2
AIX 5L Version 5.2 or earlier	No change	Mount not allowed. Other file system commands which attempt to read the formatted file system also not allowed. (backbyinode, for example)
AIX 5L Version 5.3 or later	Migration to EAv2 is not required. File system can be mounted with no changes. Other file system commands which attempt to read the formatted file system are also allowed.	New support

In order to change a JFS2 file system to EAv2, use the following command:

```
#chfs -a ea=v2 fs_name
```

3.2.6 JFS2 ACL support for NFS V4

Starting with AIX 5L Version 5.3, the Enhanced Journaled File system now supports ACLs for NFS version 4. This allows you to establish fine-grained access control for file system objects and support inheritance features. In order to take advantage of ACL support for NFS V4, you need to create the JFS2 file system with extended attribute format version 2 (EAv2). If you have file systems that have already been created with extended attribute format version 1(EAv1), you need to convert to EAv2 first, as shown in Figure 3-8 on page 132.

```
       Change / Show Characteristics of an Enhanced Journaled File System

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

[MORE...2]                                              [Entry Fields]
  SIZE of file system
         Unit Size                                      512bytes              +
*        Number of units                               [262144]               #
  Mount GROUP                                          []
  Mount AUTOMATICALLY at system restart?                no                    +
  PERMISSIONS                                           read/write            +
  Mount OPTIONS                                        []                     +
  Start Disk Accounting?                                no                    +
  Block Size (bytes)                                    4096
  Inline Log?                                           no
  Inline Log size (MBytes)                             [0]                     #
  Extended Attribute Format                            [v2]
  ENABLE Quota Management?                              no                    +
[BOTTOM]

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

Figure 3-8 SMIT menu for converting EAv1 to EAv2

NFS4 ACLs using the command line

The JFS2 file system included with AIX supports two ACL types from AIX 5L Version 5.3: AIXC and NFS4. The AIXC (AIX Classic) ACL type provides for the ACL behavior as defined on previous releases of AIX. This ACL type consists of the regular base mode bits and extended permissions. With extended permissions, you can permit or deny file access to specific individuals or groups without changing the base permissions. Example 3-5 shows the typical output of **acledit** before converting AIXC ACL to NFS4.

Example 3-5 Output of the **acledit** *command before converting to NFS4*

```
#acledit /fs2
*
* ACL_type   AIXC
*
attributes:
base permissions
    owner(root):  r-x
    group(system):  r-x
    others:  r-x
extended permissions
    disabled~
```

Now convert ACL with the **aclconvert** command. The usage of the **aclconvert** command and the output of the **acledit** command after converting ACL to NFS4 are shown in Example 3-6.

Note: Before you convert AIXC to NFS4 or NFS4 to AIXC ACL, it is always recommended to back up the original ACL or files. You can go back to AIXC from NFS4, or NFS4 to AIXC ACL, but the resulting permissions are not necessarily equivalent to the original ones.

Example 3-6 Output of the acledit command after converting to NFS4

```
# aclconvert -t NFS4 /fs2
# acledit /fs2
*
* ACL_type   NFS4
**
* Owner: root
* Group: system
*
s:(OWNER@):     d       wpDd
s:(OWNER@):     a       rRWxaAcCo
s:(GROUP@):     a       rx
s:(OWNER@):     d       rwpRWxDdos
s:(GROUP@):     d       rwpRWxDaAdcCos
s:(EVERYONE@):  a       rRxac
s:(EVERYONE@):  d       wpDd
```

If you use the **acledit -v** command instead of the **acledit** command with no flags, you will be presented a lot of useful explanations of access control entry (ACE) types, ACE access masks, and so on. Table 3-6 and Table 3-7 describe what the keys mean.

Table 3-6 ACE type

Key	Description
a	ACE type allows the access described in the access mask
d	ACE type is denied the access described in the access mask
l	Audit type ACE
u	Alarm type ACE

Table 3-7 ACE access mask

Key	Description
r	Permission to read the data for the file or permission to list the contents for a directory
w	Permission to modify the file's data or permission to add a new file to a directory
p	Permission to append data to a file or permission to create a subdirectory to a directory
R	Permission to read the named attributes of a file
W	Permission to write the named attributes of a file
x	Permission to execute a file
D	Permission to delete a file or directory within a directory
a	The ability to read basic attributes (non-ACLs) of a file (Read Base Attributes)
A	Permission to change basic attributes (non-ACLs) of a file
d	Permission to delete the file
c	Permission to read the ACL
C	Permission to write the ACL
o	Permission to change the owner

NFS4 ACLs using Web-based System Manager

You can apply NFS4 ACLs using the Web-based System Manager. It may be easier for you to understand since it is well organized with the GUI. Go to the Web-based System Manager and choose **File Systems** → **Overview and tasks**; you will find a new menu **Access control list** added in Version 5.3. It will lead you to configure NFS4 ACL as shown in Figure 3-9 on page 135.

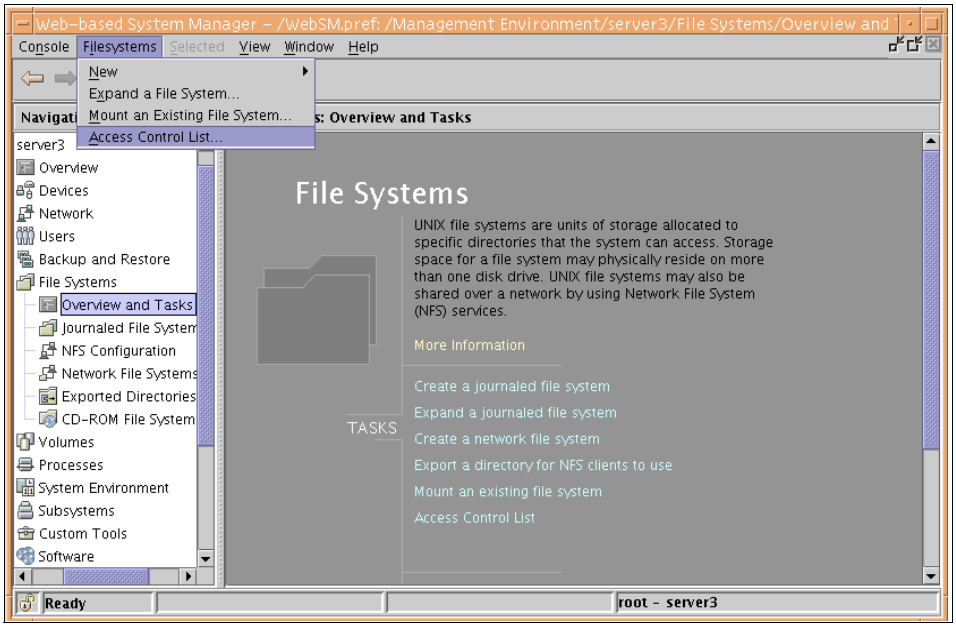

Figure 3-9 Configuring ACL through Web-based System Manager

You will be prompted to select a directory or file. This allows you to choose the file whose ACL the selected operation will be performed on, as shown in Figure 3-10.

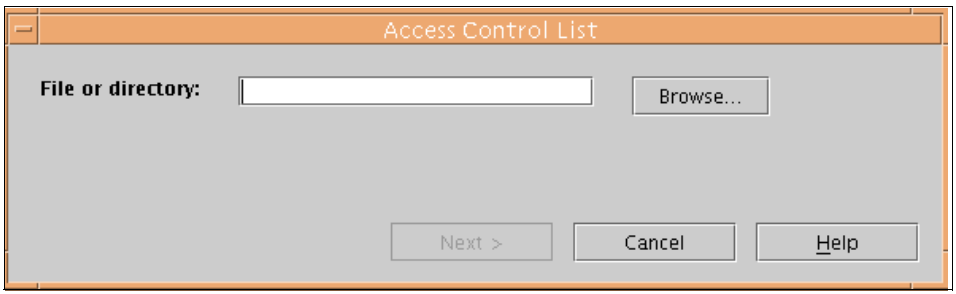

Figure 3-10 File Access Control List dialog panel

After you choose a file or directory, the second File Access Control panel is returned. Here you select the action you wish to perform, as shown in Figure 3-11 on page 136.

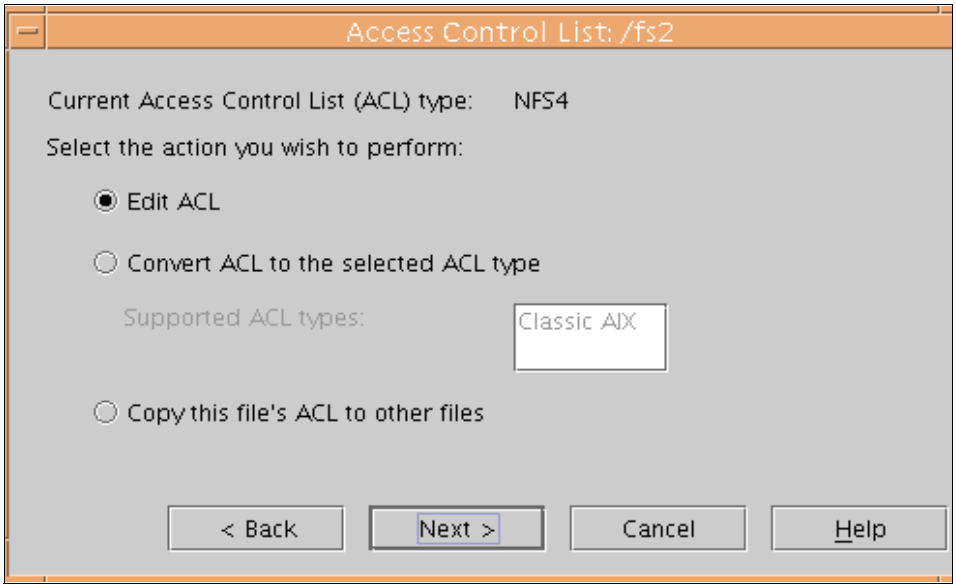

Figure 3-11 File ACL dialog: Second panel

In this panel you can choose to edit ACL or to convert ACL to the selected ACL type. Selecting Edit ACL will open the ACL edition dialog in the selected ACL type shown in Figure 3-12 on page 137. The following columns are displayed in this panel:

Identity Represents to whom this ACE applies. It is the rest of the string following the prefix of the identity string.

User type The type of user which will use the ACE. It can be one of user, group, or special. This is given by the prefix of the identity string.

ACE Type The type of the ACE entry. It is one of the following: allow, deny, alarm, or audit. This is given by the ACE type string.

Mask Indicates the various access rights defined in the ACE, expressed as a string of keys. Tips are added to those table cells that display in clear the meaning of the flags.

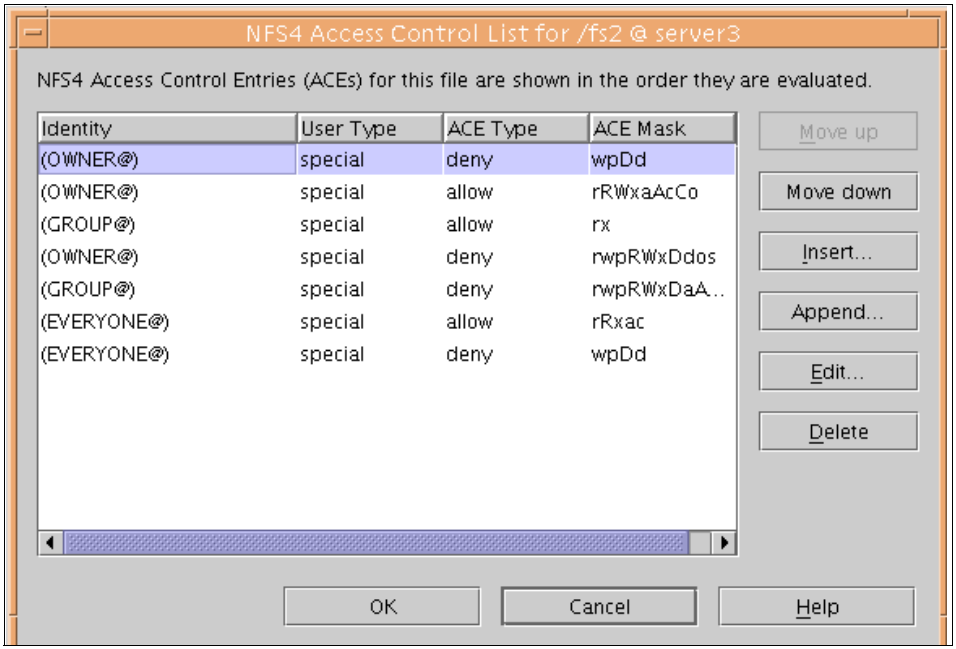

Figure 3-12 ACL edition dialog

If you choose Edit and click the Access Mask tab, you will see the panel shown in Figure 3-13 on page 138. These keys were defined in Table 3-7 on page 134.

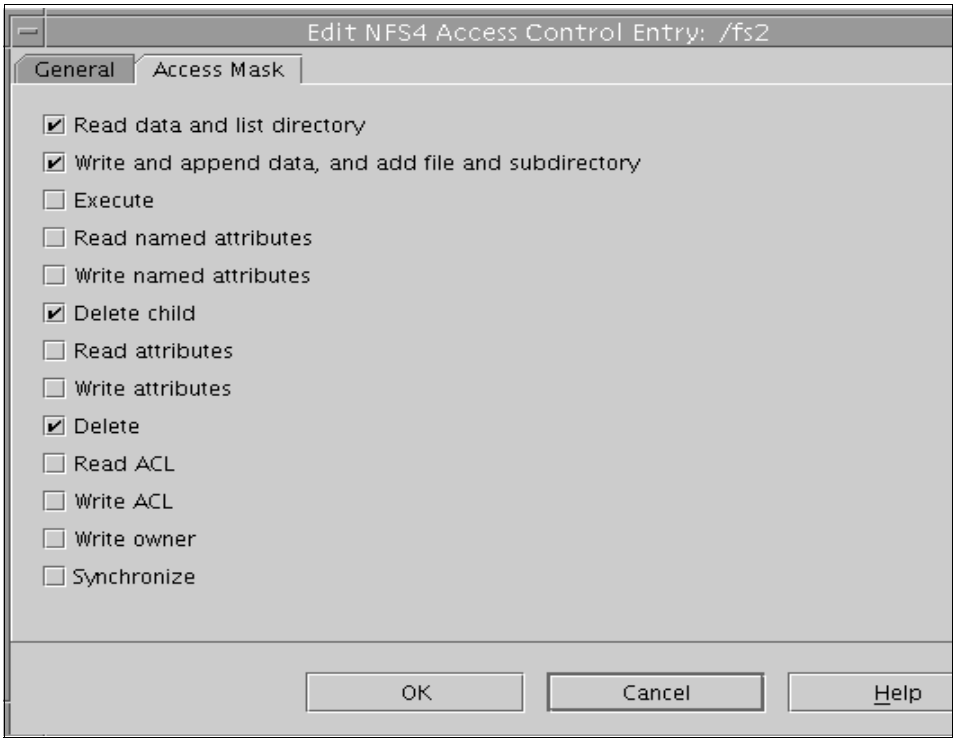

Figure 3-13 Add ACE dialog: Access Mask tab

3.2.7 ACL inheritance support

AIX classical ACL (AIXC) does not support inheritance; only NFSv4 ACL supports inheritance. When a new file or directory is created and the parent directory has an NFSv4 ACL, then an ACL will be established for the new file or directory. If an ACL is established for a child file or directory explicitly, it is decoupled from the parent's ACL. Any changes to the parent's ACL will not affect the child's ACL in this case. In an environment where users are not actively managing ACLs, all the ACLs of a directory's child files will likely have the same ACL. It makes sense to keep one copy of the ACL when the ACL data is common.

Configuring ACL with inheritance

If the following conditions are met for a certain file system, inheritance feature can be used:

1. It is a JFS2 file system with AIX 5L Version 5.3 or later.
2. Extended attribute version 2 (EAv2) is enabled with the JFS2 file system.

3. NFS4 ACL is applied.

When you configure ACL with the **acledit** command, there are fields to define different inheritance flags such as fi and di. Table 3-8 describes the inheritance flags.

Table 3-8 Keys for inheritance of NFS4 ACL

Key	Descriptions
fi	Can be placed on a directory and indicates that this ACE should be added to each new non-directory file created.
di	Can be placed on a directory and indicates that this ACE should be added to each new directory file created.
oi	Can be placed on a directory but does not apply to the directory, only to newly created files/directories as specified by the previous two flags.
ni	For child only, no inherit for grandchild.

Suppose that you have a directory named dir1 and want to inherit ACL to the sub-directories and files under the dir1 directory. Example 3-7 shows you how to accomplish this.

First, you convert to NFS4 ACL for the dir1 directory and edit the ACL with the **acledit** command. Modify ACL and add inheritance flags as needed. Then create a new file named file1 and verify file1 with the **aclget** command to see if the ACL of file1 is the same as that of dir1.

Example 3-7 Configuring ACL with inheritance

```
# aclconvert -tNFS4 dir1
# acledit dir1
*
* ACL_type   NFS4
*
*
* Owner: root
* Group: system
*
s:(OWNER@):     a       rwpRWxDaAdcCo   fidi
u:slson:        a       r               fidi

# touch dir1/file1
# aclget dir1/file1
*
* ACL_type   NFS4
*
*
```

```
* Owner: root
* Group: system
*
s:(OWNER@):     a       rwpRWxDaAdcCo   fidi
u:slson:        a       r               fidi
```

The inheritance as implemented in AIX 5L Version 5.3 is only valid for the newly created files and directories under the parent directory, not for the existing files and directories. if you want to inherit the ACL to all existing files and directories as well, then use the **aclget parent_directory | aclput -R directory** command.

Note: Before you convert AIXC to NFS4 or NFS4 to AIXC ACL, it is always recommended to back up the original ACL or files. You can go back to AIXC from NFS4 or NFS4 to AIXC ACL, but the resultant permissions are not necessarily equivalent to the original ones.

4

Reliability, availability, and serviceability

This chapter includes descriptions of the following enhancements for AIX 5L:

- Error logging, core files, and system dumps
  - Error log RAS
  - Enhancements for a large number of devices
  - Core file creation and compression
  - System dump enhancements
  - DVD support for system dumps
  - `snap` command enhancements
- Trace enhancements
  - Administrative control of the user trace buffers
  - Single thread trace

4.1 Error log RAS

Under very rare circumstances, such as powering off the system exactly while the errdemon is writing into the error log, the error log may become corrupted. In AIX 5L Version 5.3 there are minor modifications made to the errdemon to improve its robustness and to recover the error log file at its start.

When the errdemon starts, it checks for error log consistency. First it makes a backup copy of the existing error log file to /tmp/errlog.save and then it corrects the error log file, while preserving consistent error log entries.

The difference from the previous versions of AIX is that the errdemon used to reset the log file if it was corrupted, instead of repairing it.

If the error log copy to the /tmp/errlog.save fails, the CORRUPT_LOG and the LOG_COPY_FAILED or LOG_COPY_IO entry is written to the error log. These entries will be logged using the errlog() function. They will be logged along with other entries waiting in the buffer. Note that, in many corruption scenarios, the system can continue to write entries to the error log. The main reason the copy would fail is insufficient space in /tmp.

If the error log was copied and repaired, the CORRUPT_LOG error entry is written to the error log.

4.2 Enhancements for a large number of devices

For each device configured, an entry is made in the /dev directory. On systems with many devices, it may be possible for the system to run out of space in the root file system or to run out of inodes. Prior to Version 5.3, you would not know the cause of errors related to configuring a large number of devices since that error would not be reported. Starting with Version 5.3, the **cfgmgr** command now reports this error to the user. If an error is detected, the **cfgmgr** command determines if the problem is lack of space or inodes and sets a flag. If this occurs during runtime, the **cfgmgr** command displays an error message and exits. If this scenario occurs during boot, AIX makes an entry in the boot log to record the event and continues. Before the **cfgmgr** command exits, if the flag is set, it displays the error message.

Error message support

If you have an error message related to configuring a large number of devices, it appears similar to the following:

```
cfgmgr: 0514-624 Unable to define or configure one or more devices
        because the filesystem is full or is out of inodes
```

If a system runs into this problem when a user runs the **cfgmgr** command at the command line, the user will see the error message on the console. If the system runs into this problem during boot, the user can view the boot log with the **alog -ot boot** command.

Configuring a large number of devices

The number of devices that AIX can support varies from system to system, depending on several important factors. The following factors have an impact on the file systems that support the devices:

- Configuring a large number of devices requires storage of more information in the ODM device-configuration database. It can also require more device special files. As a result, more space and more inodes are required of the file system.
- Some devices require more space than others in the ODM device-configuration database. The number of special files or inodes used also varies from device to device. As a result, the amount of space and inodes required of the file system depends on the types of devices on the system.
- Multipath I/O (MPIO) devices require more space than non-MPIO devices because information is stored in the ODM for the device itself as well as for each path to the device. As a rough guideline, assume that each path takes up the space of one-fifth of a device. For example, an MPIO device with five paths will have the space equivalent to two non-MPIO devices.
- AIX includes both logical devices and physical devices in the ODM device-configuration database. Logical devices include volume groups, logical volumes, network interfaces, and so on. In some cases, the relationship between logical and physical devices can greatly affect the total number of devices supported. For example, if you define a volume group with two logical volumes for each physical disk that is attached to a system, this will result in four AIX devices for each disk. On the other hand, if you define a volume group with six logical volumes for each physical disk, there will be eight AIX devices for each disk. Therefore, only half as many disks could be attached.
- Changing device attributes from their default settings results in a larger ODM device-configuration database and could lead to fewer devices that can be supported.
- More devices means more real memory is required.

Two file systems are used by AIX to support devices:

- The RAM file system is used during boot in an environment that has no paging space and no disk file systems mounted. The size of the RAM file system is 25 percent of the system memory size up to a maximum of 128 MB.

One inode is allocated for every KB in the RAM file system. If the system memory size is 512 MB or larger, then the RAM file system will be at its maximum size of 128 MB with 131072 inodes. If either the amount of RAM file system space or number of inodes needed to support the attached devices exceeds what has been allocated to the RAM disk, the system might not boot. If this is the case, you must remove some of the devices.

- The space and inodes of the root file system (rootvg) on the disk can be increased as long as there are unallocated partitions in the rootvg. With the maximum RAM file system size, it is likely that up to 25,000 AIX devices could be configured. These numbers include both physical and logical devices. Depending on the various factors mentioned in this section, your system might be able to configure more or fewer devices than this number.

4.3 Core file creation and compression

Applications running on the AIX system may produce quite large core files.

The changes to the core file creation introduced in Version 5.3 are the following:

- Core file compression
- Enabling core file destination directory and core file naming

To check the core file creation settings or change them you can use two new commands:

lscore Checks the current or default settings

chcore Changes the core file creation settings

Example 4-1 shows how to use the commands to check and change the core file creation settings. The discussion that follows shows the steps you probably will use in your environment.

Example 4-1 Managing core file creation settings

```
# lscore
compression: off
path specification: off
corefile location: not set
naming specification: off
# mkdir /tmp/cores
# chcore -c on -p on -l /tmp/cores -n on

<logout and login>

# lscore
compression: on
```

```
path specification: on
corefile location: /tmp/cores
naming specification: on
# sleep 1000 &
[1]     360620
# kill -11 %1
#
[1] + Memory fault(coredump)   sleep 1000 &
# ls -l /tmp/cores
total 16
-rw-r--r--   1 root     system         5418 Jun 22 14:44 core.360620.22194424.Z
#
```

First we check the settings using the **lscore** command. In the example, we start from scratch and the displayed settings are the default ones.

Log in as the user you want to change the settings for. Use the **mkdir /tmp/corefiles** command to create the target directory for the core files and check that the directory is writable to the user. Use the **chcore -c on -p on -l /tmp/corefiles -n on** command to change the core file creation settings. The basic flags have the following meanings:

-c on/off/default Turns the compression on, off, or returns to default.

-p on/off/default Turns the use of destination directory on, off, or returns to default.

-l *directory* Specifies the target directory where core files are created. The directory must exist before you run the command and must be specified using fully qualified name.

-n on/off/default This flag sets the core file naming as if set by the CORE_NAMING environment variable.

After you make changes to the core file creation settings, users need to log out and log in again in order to apply the change.

The easiest way to test the new settings is to start any process, such as **sleep 1000 &** in the background and kill this process by the **kill -11 %1** command using signal SIGSEGV. The core file is created in the defined directory and its name is created from the process ID and the timestamp: `core.`*`PID.timestamp`*`.Z`. The compressed core file is compressed by the LZW (Lempel-Zev-Welch) algorithm and can be decompressed using the **uncompress** command if needed.

Note: If you uncompress the compressed core file and compress it back using the **compress** command, the file size will differ. This is because the in kernel core file compression stores some information in raw format. This is a correct behavior.

With the **chcore** command you can specify the user you want to change. The command can be run by any user, but only root can change settings for other users. After the settings are changed, an entry to /etc/security/user is added:

```
root:
        ...
        core_compress = on
        core_path = on
        core_naming = on
        core_pathname = /tmp/cores
```

If the -R *load_module* is used the core settings are stored in an external security repository, like LDAP, DCE, or GSA, instead of the default /etc/security/user file. The *load_module* has to be supplied by the external program.

If the core is generated by the **gencore** command, the core file naming and path may be omitted depending on the file name specified to the command. Specifying the full path name will omit the core file naming and path settings. For example, the **gencore *PID /path/filename*** command will omit the core file naming and path, but the **gencore *PID filename*** command will observe the core file naming and path settings.

4.4 System dump enhancements

System dumps are enhanced in AIX 5L Version 5.3 to provide more information about the status of the dumps. These enhancements include:

- Dump information to TTY
- Verbose flag to the **sysdumdev** command
- The **dumpfmt** command formatting

4.4.1 Dump information to TTY

During the creation of the system dump, additional information is displayed on the TTY about the progress of the system dump. Example 4-2 on page 147 shows a sample output to the TTY console.

Example 4-2 TTY output from system dump

```
# sysdumpstart -p
Preparing for AIX System Dump . . .
Dump Started .. Please wait for completion message
AIX Dump .. 23330816 bytes written - time elapsed is 47 secs
Dump Complete .. type=4, status=0x0, dump size:23356416 bytes
Rebooting . . .
```

At the time of writing, the kernel debugger and the 32-bit kernel must be enabled to see this function, and we tested the function only on the S1 port. However, this limitation may change in the future.

4.4.2 Verbose flag for the sysdumpdev command

Following a system crash there exist scenarios where a system dump may crash or fail without one byte of data written out to the dump device, for example power off or disk errors. For cases where a failed dump does not include the dump minimal table, it is useful to save some trace back information in the NVRAM. Starting with Version 5.3, the dump procedure is enhanced to use the NVRAM to store minimal dump information.

If the dump fails, use the **sysdumpdev -vL** command to check the reason for the failure. Example 4-3 and Example 4-4 on page 148 show samples of failed dumps.

Example 4-3 sysdumpdev -vL output in case of I/O error during dump

```
# sysdumpdev -Lv
0453-039

Device name:         /dev/hd6
Major device number: 10
Minor device number: 2
Size:                262144 bytes
Uncompressed Size:   4491225 bytes
Date/Time:           Fri Jun 25 16:04:19 CDT 2004
Dump status:         -4
I/O error
Dump copy filename: /var/adm/ras/vmcore.2.Z
End Date/Time:       Fri Jun 25 16:04:25 CDT 2004
Dump Duration:       6s
Traceback:
 iar:000d88bc        dmpnow+28
addr:000d88c8        dmpnow+34
addr:0037e21c        dmpioctl+23c
addr:002cde68        rdevioctl+c8
addr:003c0524        spec_ioctl+68
```

```
addr:002e032c        vnop_ioctl+24
Dump History:
 Cdt:                ldr
Entry:               ldr32l
Places:              jwrite
Leds Line 1:         0C2 0C1
Leds Line 2:         CDT: thrd
                     DUMP TABLE     3
                     CDT: ldr
                     5
                     5
                     5
                     CDT: thrd
```

In Example 4-3 we initiated a system dump using the **sysdumpstart -p** command and while the system dump was writing to the disk we pulled out the disk. The **sysdumpdev -Lv** command after a failed system dump shows us information that the I/O operation had failed and provides trace back information for analysis.

Example 4-4 sysdumpdev -vL output in case of power off during dump

```
root@Server4: sysdumpdev -Lv
0453-039

Device name:         /dev/hd6
Major device number: 10
Minor device number: 2
Size:                0 bytes
Uncompressed Size:   17717 bytes
Date/Time:           Fri Jun 25 16:16:38 CDT 2004
Dump status:         -3
dump crashed or did not start
End Date/Time:       Fri Jun 25 16:16:38 CDT 2004
Dump Duration:       0ns
Traceback:
 iar:000d88bc        dmpnow+28
addr:000d88c8        dmpnow+34
addr:0037e21c        dmpioctl+23c
addr:002cde68        rdevioctl+c8
addr:003c0524        spec_ioctl+68
addr:002e032c        vnop_ioctl+24
Dump History:
 Cdt:                dmp_minimal
Entry:               dbgtbls
Places:              jwrite
                     jwrite
                     jwrite
Leds Line 1:         0C2
```

```
Leds Line 2:
                    DUMP ON CPU 1
                    DUMP TABLE     0
                    CDT: dmp_minimal
                    DUMP TABLE     1
                    CDT: proc
```

In Example 4-4 we initiated the system dump and pressed the power button while the dump was running. In this case the **sysdumpdev** shows different dump status and dump history.

Tip: During the dump the LCD display shows progress codes and information. If you could not see the LCD you can get the codes from the **sysdumpdev -Lv** command output. See “Leds Line 1:” and “Leds Line 2:” sections of Example 4-3 or Example 4-4.

In the unlikely event a dump crashes, a second dump traceback is displayed instead of the dump history.

4.4.3 dmpfmt command changes

The **dmpfmt -c** command is extended to verify that certain dump components are present in the dump in addition to the dump integrity test. The **dmpfmt -c** command is executed by the system at boot time and its output is added to the DUMP_STATS error log entry.

4.5 DVD support for system dumps

This enhancement adds the possibility to:

- Copy system dumps to DVD media
- Use DVD as a primary or secondary dump device
- Enhance the **snap** command to handle system dumps on DVD

In order to support this function the target DVD device should be DVD-RAM or writable DVD. Remember to insert an empty writable DVD in the drive when using the **sysdumpdev** or **snap** commands, or when you require the dump to be copied to the DVD at boot time after a crash. If the DVD media is not present the commands will give error messages or will not recognize the device as suitable for system dump copy.

4.5.1 Copy system dumps to DVD media

Figure 4-1 shows the dialog you get at boot time. This dialog prompts you to select the target device to be used to receive the dump when there is insufficient space in the copy directory, usually in the /var file system.

```
               Copy a System Dump to Removable Media

    The system dump is 22884352 bytes and will be copied from /dev/hd6
    to media inserted into the device from the list below.

    Please make sure that you have sufficient blank, formatted
    media before you continue.

    Step One:  Insert blank media into the chosen device.
    Step Two:  Type the number for that device and press Enter.

           Device Type                   Path Name

     1  tape/scsi/scsd                   /dev/rmt0
>>>  2  cdrom/scsi/scsd                  /dev/cd0




                        +-----------------------------------------------------
    88  Help ?          |System Dump has been copied to /dev/rcd0.
    99  Exit            |Exit, then use the pax command to extract the dump.
                        |
>>> Choice[2]: 2
```

Figure 4-1 Copy system dump to media boot time dialog

Select the DVD device as the target device and the system will copy the dump to the DVD using the **pax** command format.

After the system dump is successfully copied to the DVD media, you can check or restore the dump from the system prompt using the **pax** command as follows:

```
# pax -v -f /dev/rcd0
PAX format archive
-rwx--x---  0 root     system   22884864 Jun 28 17:29 dump_file.Z
```

Setting the dump device to DVD

The **sysdumpdev** command as modified in AIX 5L Version 5.3 accepts DVD devices as the primary or secondary dump device. The following example shows how to set the primary dump device to /dev/cd0 and the secondary to /dev/cd1:

```
sysdumpdev -p /dev/cd0 -s /dev/cd1
```

If the system dump destination is a DVD, then the system dump should not exceed one DVD media size.

Using DVD for the snap command

The **snap** command is enhanced in that it enables the DVD device to be selected as source for the dump and that its -o flag can accept the DVD device as an output device. Use the **snap** command to generate its output to DVD as follows:

```
# snap -a -o /dev/cd0
...
The previous dump to paging space was not copied to disk at reboot.
You were given the opportunity to copy the dump to external media.
Would you like to restore the dump? (y/n)  y

Please enter external device(default /dev/rmt0):  /dev/cd0
Setting input device to /dev/cd0... done.
Please insert the media containing the dump into /dev/cd0 and press Enter
done.
...
Please remove dump media and replace with snap output media.
Press enter to continue
...
Copying information to /dev/cd0... Please wait... done.
...
```

The snap uses the **pax** format to copy the data to the DVD. You can check the DVD content using the **pax -v -f /dev/cd0** command.

4.6 snap command enhancements

The **snap** command already exists in the previous versions of AIX, but Version 5.3 extends its function in using external scripts, enabling the **snap** command to split up the output **pax** format file into smaller pieces or extending the collected data.

Extending snap to run external scripts

The scripts can either be parameters to the **snap** command, or in a case that the **all** option is specified, the scripts are expected to be in the /usr/lib/ras/snapscripts directory. The **snap** command can call the scripts in three different way:

- Specifying the script name in the /usr/lib/ras/snapscripts directory that the **snap** will call.
- Specifying the all keyword to the **snap** command. The **snap** command then calls all scripts in the /usr/lib/ras/snapscripts directory.

- Specifying the file:*filename* keyword with filename reference to the file containing the list of external scripts. The file must contain one script name per line.

First, prepare a sample script in the /usr/lib/ras/snapscripts directory and name it, for example, test_snap_script.ksh. We created just a short script that writes the environment to the log file.

```
# cat /usr/lib/ras/snapscripts/test_snap_script.ksh
#!/usr/bin/ksh
case $PASSNO in
  "1")  echo "4096" > $SCRIPTSIZE
        ;;
  "2")  set > $SCRIPTLOG
        ;;
  *)    echo "ERROR IN THE ENVIRONMENT"
esac
exit 0
```

Then we ran the **snap** command and checked the output of the script. The output is placed in the /tmp/ibmsupt/*scriptname* directory.

```
# snap test_snap_script.ksh
********Checking and initializing directory structure
Creating /tmp/ibmsupt directory tree... done.
Creating /tmp/ibmsupt/test_snap_script.ksh directory tree... done.
Creating /tmp/ibmsupt/testcase directory tree... done.
Creating /tmp/ibmsupt/other directory tree... done.
********Finished setting up directory /tmp/ibmsupt

Checking Space requirement for test_snap_script.ksh
Checking for enough free space in filesystem... done.

Gathering test_snap_script.ksh data
# ls -al /tmp/ibmsupt/test_snap_script.ksh
total 16
-rw-------   1 root     system            0 Jun 24 16:24
test_snap_script.ksh.err
-rw-------   1 root     system          900 Jun 24 16:24
test_snap_script.ksh.log
-rw-------   1 root     system            0 Jun 24 16:24
test_snap_script.ksh.out
-rw-------   1 root     system            5 Jun 24 16:24
test_snap_script.ksh.size
```

The system environment that is passed to the script from the **snap** command contains the following variables:

SNAPDIR	The destination directory for all **snap** command output.
PASSNO	Pass number of the script. The **snap** command calls the third party script twice and tells the script the phase (1 or 2) it is calling the script.
SCRIPTSIZE	The path to a file where the script is expected to write the estimated size during the first pass.
SCRIPTLOG	The path to an output log file to which the script is expected to write.

Note: The **snap** command calls the third party script in two phases. In the first phase the script is expected to prepare for the second run and calculate the estimated size of the output data. The second phase should gather the necessary data and store it in the $SNAPDIR directory.

The `snapsplit` command

The **snapsplit** command is introduced in Version 5.3. The command splits the snap.pax.Z file into smaller files or joins together split snap files. The command expects to be run from the /tmp/ibmsupt directory, where the snap.pax.Z file resides. In Example 4-5 we show the use of the command to split the snap into 4 MB files, and the command to rejoin the split files.

Example 4-5 snapsplit command

```
# snapsplit -s 4
 was split successfully ...
#ls -al
...
-rw-------   1 root     system      7179388 Jun 25 13:03 snap.pax.Z
-rw-r--r--   1 root     system      4194304 Jun 25 14:31
snap.sq1test1.062504143158.pax.Zaa
-rw-r--r--   1 root     system      2985084 Jun 25 14:31
snap.sq1test1.062504143158.pax.Zab
...
# snapsplit -u -T 062504143158
Restoring . . .
 was successfully restored ...
# ls -al
-rw-------   1 root     system      7179388 Jun 25 13:03 snap.pax.Z
-rw-r--r--   1 root     system      7179388 Jun 25 14:39
snap.sq1test1.062504143158.pax.Z
-rw-r--r--   1 root     system      4194304 Jun 25 14:31
snap.sq1test1.062504143158.pax.Zaa
```

```
-rw-r--r--   1 root     system      2985084 Jun 25 14:31
snap.sq1test1.062504143158.pax.Zab
```

The **snapsplit** command created the snap.*hostname.timestamp.pax.Zxx* files of size 4 MB or less. The **snapsplit -u** command rejoined the files to snap.hostname.timestamp.pax.Z file. You can take the timestamp for the -T flag from the name of the split files.

The -T or -h flags available with the **snapsplit** command enable you to handle snap dumps from different systems taken at different times. The -f flag enables you to handle renamed snap files.

Splitting the snap output file with the snap command

The size of the **snap** output file can be a problem. There is a new flag, -O *megabytes*, introduced in Version 5.3 that enables a split of the **snap** output file. The **snap** command calls the **snapsplit** command. You can use the flag as follows to split the large snap output into smaller 4 MB files:

```
# snap -a -c -O 4
```

Additional data collected by snap

In Version 5.3, the **snap** command collects additional information about the system, or the output is reorganized. For example, **lslpp -La, sysdumpdev -Lv, lsdev -Ccscsi** is collected to lsdev.scsi; **lsdev -Ccproc** is collected to the processor.log; **bindprocessor -q** is collected to the processor.log; and **lsvpd** is collected to lsvpd.out.

4.7 Administrative control of the user trace buffers

In previous versions of AIX, the trace buffer size for a regular user is restricted to a maximum of 1 MB. This was done to prevent a malicious user using up all the memory to be as pinned and consequently cause failures on the system. Also, in the early days of computing, this restriction was necessary because memory was at a premium.

This size threshold is working out to be a limitation for some applications, such as MQ Series, to monitor their messaging system, and consequently affects their serviceability.

Version 5.3 allows the users belonging to the system group to set the buffer threshold for the user trace buffers. The threshold is set for the system. To do this, the **trcctl** command is introduced and a menu is added to SMIT. The settings are stored in the SWservAt ODM class.

The following example shows how to use the **trcctl** command. First we check the current settings:

```
# trcctl -l

Default Buffer Size: 262144
Default Log File Size: 2621440
Default Log File: /var/adm/ras/trcfile
Non-Root User Buffer Size Maximum: 1048576
```

We change the log file size to 40 MB using the **-L** flag, the maximum buffer size that a non-root user can specify to 20 MB using the **-N** flag, the path and file name of the trace file to /tmp/tracefile using the **-o** flag and the default trace buffer size that the trace command uses to 1 MB with the **-T** flag.

```
# trcctl -L 40M -N 20M -o /tmp/tracefile -T 1M
```

We check the new settings using the **-l** flag.

```
# trcctl -l

Default Buffer Size: 1048576
Default Log File Size: 41943040
Default Log File: /tmp/tracefile
Non-Root User Buffer Size Maximum: 20971520
```

Finally, we show how to return the settings back to default using the **-r** flag. The defaults differ if you use 32-bit kernel and 64-bit kernel. Our example shows the case of the 64-bit kernel.

```
# trcctl -r
# trcctl -l

Default Buffer Size: 262144
Default Log File Size: 2621440
Default Log File: /var/adm/ras/trcfile
Non-Root User Buffer Size Maximum: 1048576
```

To ease the use of the **trcctl** command the SMIT is updated with a new menu Manage Trace under the **smit trace** fastpath, followed by two sub menus that change the settings or restore the default. Figure 4-2 on page 156 shows the **smit cngtrace** menu.

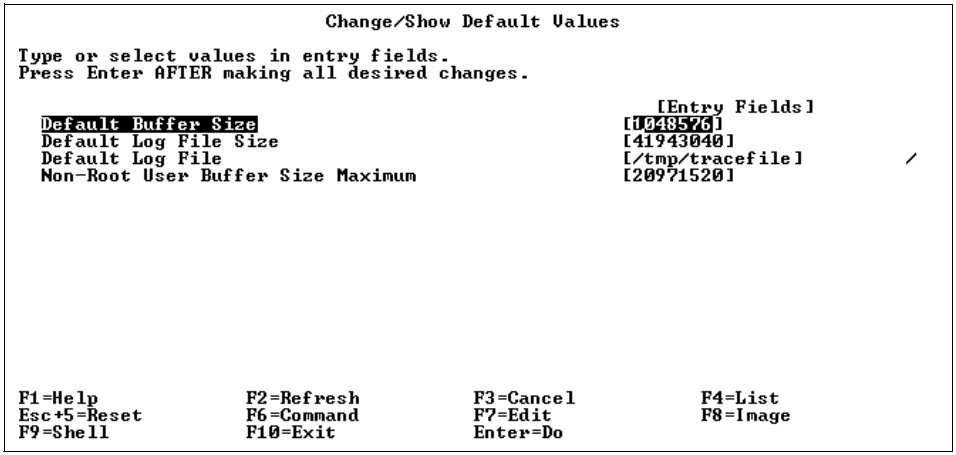

Figure 4-2 smit cngtrace SMIT menu

4.8 Single thread trace

In versions prior to Version 5.3, the AIX system trace traced the entire system. The **trace** command in Version 5.3 is enhanced with new flags. These flags enable the trace to run only for specified processes, threads, or programs. This will save space in the trace file and also helps to focus on just the area you are interested in.

The new flags introduced to the **trace** command have the following meanings:

-A Sets process trace for the listed processes following the flag.

-t Sets threads to be traced.

-I Specifies that interrupt-level events are to be traced.

-P Specifies if the trace is propagated to child processes or threads.

-x Specifies the trace is to be run for the specified program. After the program ends, the trace ends.

-X Specifies the trace is to be run for the specified program. After the program ends, the trace continues to run.

To trace one specific command, its child processes and threads, start the trace command as follows:

```
# trace -J fact -x /ad01/fop_test -P p -a
# trcrpt -O exec=y -O pid=y -O tid=y -O svc=y -O timestamp=1
...
```

In this example the trace is started for the /ad01/fop_test program and the file activities are traced. The trace stops immediately after the program ends. Then the trace report can be generated using the **trcrpt** command.

If you started the trace with the **-X** flag instead of the **-x** flag, the trace does not stop when the traced program ends. Do not forget to stop the trace after you are done.

The following is another example where we start a trace that traces two processes, inetd and sendmail:

```
# ps -ef | grep inetd
    root 131266 225456   0   Jun 22      -  0:00 /usr/sbin/inetd
# ps -ef | grep sendmail
    root 184494 225456   0   Jun 22      -  0:00 sendmail: accepting
connections
# trace -J fact -A '131266 184494' -P p -a
# trcstop
# trcrpt -O exec=y -O pid=y -O tid=y -O svc=y -O timestamp=1
...
```

In this example we check for the PID of the two processes we intend to trace and start the trace for those two PIDs. In this case, the trace is not stopped automatically when the process stops and we need to use the **trcstop** command to stop it. Use the **trcrpt** command to generate the trace report.

When you want to generate thread-specific traces using the **-t** flag, you need to supply the TID additionally to the **-t** flag. You can get the TID from, for example, the **ps -efm -o THREAD** command.

Using the **-I** flag you can add additional information to the trace containing the interrupts.

In addition to the new flags of the **trace** command, the **smit trcstart** menu is updated by items reflecting the new flags. See Figure 4-3 on page 158 for a sample **smit trcstart** menu.

```
                                START Trace

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

[TOP]                                                   [Entry Fields]
  EVENT GROUPS to trace                                 []                          +
  ADDITIONAL event IDs to trace                         []                          +
  Event Groups to EXCLUDE from trace                    []                          +
  Event IDs to EXCLUDE from trace                       []                          +
  Process IDs to Trace                                  []                          +
  Program to Trace                                      []
  Propagate Tracing to                                  [new processes and thr>     +
  Trace MODE                                            [alternate]                 +
  STOP when log file full?                              [no]                        +
  LOG FILE                                              [/var/adm/ras/trcfile]
  SAVE PREVIOUS log file?                               [no]                        +
  Omit PS/NM/LOCK HEADER to log file?                   [yes]                       +
  Omit DATE-SYSTEM HEADER to log file?                  [no]                        +
[MORE...4]

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

Figure 4-3 smit trcstart SMIT menu

5

System management

AIX 5L provides many enhancements in the area of system management and utilities. This chapter discusses these enhancements. Topics include:

- InfoCenter for AIX 5L Version 5.3
- Multiple desktop selection from BOS menus
- Erasing hard drive during BOS install
- Service Update Management Assistant
- Long user and group name support
- Dynamic reconfiguration usability
- Paging space garbage collection
- Dynamic support for large page pools
- Interim Fix Management
- List installed filesets by bundle
- Configuration file modification surveillance
- DVD backup using the mkdvd command
- NIM security
- High Available NIM (HA NIM)
- General NIM enhancements

5.1 InfoCenter for AIX 5L Version 5.3

Starting with AIX 5L Version 5.3, IBM @server pSeries and AIX documentation will be available in one of two information centers: the IBM @server pSeries and AIX Information Center on the Web, and the AIX Information Center on the documentation CD. Figure 5-1 shows the AIX Information Center on the documentation CD.

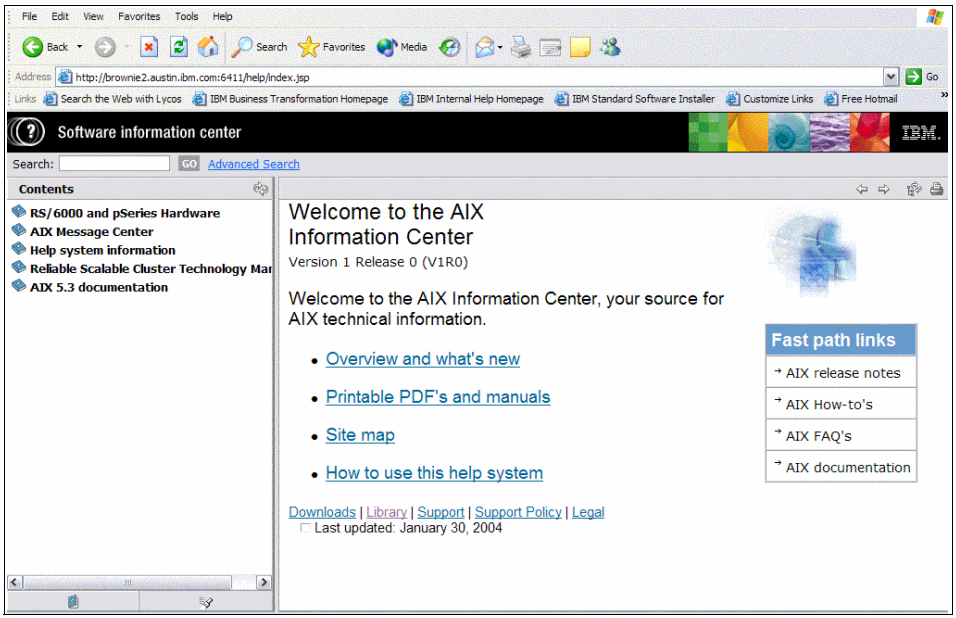

Figure 5-1 AIX Information Center on the documentation CD

The AIX and pSeries Information Center is more than a portal to documentation. From this Web site, you can access the following tools and resources:

`http://publib16.boulder.ibm.com/pseries/en_US/infocenter/base/`

- A message database that shows what error messages mean and, in many cases, how you can recover. This database also provides information for LED codes and error identifiers.
- How-to tips with step-by-step instructions for completing system administrator and user tasks.
- FAQs for quick answers to common questions.
- The entire AIX software documentation library for Version 5.1, Version 5.2, and Version 5.3. Each publication is available in PDF format, and abstracts are provided for books for Version 5.2 and Version 5.3.

- Centralized information previously located throughout the library and easier access to information about some new AIX functions:
  - A new selection in the navigation bar centralizes all partitioning information, including planning, installation, and implementation information for partitioned-system operations.
  - A new Advanced Accounting publication is available. Advanced Accounting offers increased flexibility, allowing users to customize it to meet their needs. Advanced Accounting allows for the collection of data that is not necessarily associated with a user, but may be associated with specific, user-defined projects. Interval Accounting is now possible, so that users can collect accounting data from long-running jobs while the job is running.
  - A new Partition Load Manager for AIX Guide and Reference provides experienced system administrators with information about how to perform such tasks as installing, configuring, and managing Partition Load Manager for AIX. This guide also provides the administrator with reference information about commands and files that are used to run and manage Partition Load Manager for AIX.
  - Links to the entire pSeries and p5 hardware documentation library.
  - A resources page that links users to other IBM and non-IBM Web sites proven useful to system administrators, application developers, and users.
  - Links to related documentation from IBM, including white papers, IBM Redbooks, and technical reports on topics such as RS/6000, SP, and HACMP for AIX. Release Notes and readme files are also available through the information center.
  - Several new videos are available for customer-installable features and customer-replaceable parts.
  - A new application, the AIX Information Center, will be available for installation beginning with AIX 5L Version 5.3. The information center will provide navigation and search capabilities for all installed AIX 5L Version 5.3 publications. The information center will be included on the AIX Documentation CD. It can be installed and used on a local system or installed on a documentation server for intranet use. The information center is powered by Eclipse technology.

5.2 Multiple desktop selection from BOS menus

Prior to Version 5.3, you can choose only one of three desktops during BOS install (CDE, GNOME, or KDE). If you need any other desktops, you can add them after rebooting the system. Starting with Version 5.3, multiple desktop

selection from the BOS install menus is available. The available choices are CDE, GNOME, and KDE.

Changes to the BOS menus

The Install Options menu remains the same. The Install More Software menu is enhanced with two additional prompts:

If CDE is the desktop selected from the Install Options menu, prompts would appear similar to the following:

```
4. GNOME Desktop (Toolbox for Linux Application).....No
5. KDE Desktop (Toolbox for Linux Application).....No
```

If GNOME is the selected desktop:

```
4. CDE Desktop (Volume 2).....No
5. KDE Desktop (Toolbox for Linux Application).....No
```

If KDE is the selected desktop:

```
4. CDE Desktop (Volume 2).....No
5. GNOME Desktop (Toolbox for Linux Application).....No
```

Each prompt to install an additional desktop defaults to *No*. If no desktop is chosen (Desktop = "NONE" option on the Install Options menu), the prompts for an additional desktop are not displayed. If the INSTALL_TYPE = CC_EVAL or the CONSOLE is not /dev/lft0, the prompts for an additional desktop are not displayed.

Login screens

The existing implementations of the three desktops (CDE, GNOME, or KDE) each provide an interface to allow selection of another desktop manager.

- From the GNOME login screen, you can select Session and access a menu of possible sessions. CDE and KDE are displayed on the session menu if they are available.
- From the KDE login screen, you can select CDE or GNOME as well as KDE.
- From the CDE login screen, you can select the option button and then select Session to get a menu of session types that includes GNOME and KDE if they are installed.

5.3 Erasing hard drive during BOS install

AIX 5L Version 5.3 now provides hard disk erasure of the installation drives at install time through the BOS install menus and through the bosinst variables.

When you choose to erase hard drives, the erasure process will be internally performed by the **diag** command.

Changes to BOS menus

The following changes were made to the BOS install menus.

- For the installation case

  If you are doing an overwrite install and you select the disks to install to, then you will see a new option 55 More Disk Options which will take you to the Erasure Options for Disks menu. It will in turn continue on from there normally.

- For the erase and not install case (the maintenance case)

  The maintenance menu will have option 4 Erase Disks, which will take the user to the Select Disk(s) That You Want to Erase menu. Continuation from there will take the user to the Erasure Options for Disks menu. It will be the last menu, and as such will then run the erase and exit.

New erasure menu

The new erasure menu is shown in Example 5-1.

Example 5-1 New erasure menu

```
                Erasure Options for Disks
Select the number of times the disk(s) will be erased, and select the
corresponding pattern to use for each disk erasure.  If the number of patterns
to write is 0 then no disk erasure will occur.  This will be a time consuming
process.  Either type 0 and press Enter to continue with the current settings,
or type the number of the setting you want to change and press Enter.

    1  Number of patterns to write..... 0
    2  Pattern #1..... 00
    3  Pattern #2..... ff
    4  Pattern #3..... a5
    5  Pattern #4..... 5a
    6  Pattern #5..... 00
    7  Pattern #6..... ff
    8  Pattern #7..... a5
    9  Pattern #8..... 5a

>>> 0  Continue with choices indicated above
   88  Help ?
   99  Previous Menu
>>> Choice[0]:
```

New disk selection menu

Example 5-2 shows a new disk selection menu very similar to the one where users select the disks to install to. This is used when the user is selecting to erase the drives but not to install them.

Example 5-2 New disk selection menu

```
              Select Disk(s) That You Want to Erase
Type one or more numbers for the disk(s) to be used for erasure and press
Enter.  To cancel a choice, type the corresponding number and press Enter. At
least one disk must be selected.  The current choices are indicated by >>>.

        Name     Location Code       Size    VG Status     Bootable    Maps
>>>  1  hdisk0    10-80-00-0,0       4303      rootvg          Yes       No
     2  hdisk1    10-80-00-0,4       4303      none            Yes       No

>>> 0  Continue with choices indicated above

   66  Devices not known to Base Operating System Installation

   77  Display More Disk Information

   88  Help ?

   99  Previous Menu

>>> Choice[0]:
```

New and changed bosinst.data entries

No changes are needed to the target disk stanza. These stanzas will be used to also designate the drive to erase. There is a way to designate a set to install and a different set to erase. Table 5-1 lists the new and changed bosinst entries that will be under the control_flow stanza.

Table 5-1 New and changed bosinst.data entries

Stanza	Description
ERASE_PATTERNS	Specifies the patterns to write to the chosen hard drives. The value for this field is a comma separated list of the patterns to use for each erasure of the drives. A valid pattern is a hexadecimal value from 0 to ffffffff. The number of patterns specified must be equal to or greater than the number of iterations specified in ERASE_ITERATIONS. If ERASE_ITERATIONS is 0 then this field is ignored. Ex: If ERASE_ITERATIONS = 3 then a valid entry for this field could be ERASE_PATTERNS = 00, ff, 0a0a0a0a.

Stanza	Description
ERASE_ITERATIONS	Specifies the number of times to erase the chosen hard drives before the installation occurs. This field is only valid when the INSTALL_METHOD field is set to overwrite or erase_only. The choices for this field are the numbers from 0 to 8. If the field is set to 0 then no erasure of the hard drives will occur. The default is 0.
INSTALL_METHOD	A new value besides overwrite, migrate, and preserve will be added. It will be erase_only, which will only erase the drives and not do an installation.

5.4 Service Update Management Assistant

AIX 5L development is highly focused on implementing tools and functions which help to fulfill IBM's Autonomic Computing strategy. If you apply the Autonomic Computing paradigm to the area of software maintenance you envision an environment where most, if not all, tasks related to fix and maintenance level management are automated and no longer require human intervention.

In a first approach, AIX 5L Version 5.2 provided proactive capabilities through the **compare_report** command. This command and its SMIT interface allow the comparison of installed software or fix repositories to a list of available fixes from the IBM support Web site, enabling system administrators to develop a proactive fix strategy.

AIX 5L Version 5.3 advances one step further and introduces automatic download, scheduling, and notification capabilities through the new Service Update Management Assistant (SUMA) tool. SUMA is fully integrated into the AIX Base Operating System and supports scheduled and unattended task-based download of Authorized Program Analysis Reports (APARs), Program Temporary Fixes (PTFs), and recommended maintenance levels (MLs). SUMA can also be configured to periodically check the availability of specific new fixes and entire maintenance levels, so that the time spent on such system administration tasks is reduced. The SUMA implementation allows for multiple concurrent downloads to optimize performance and has no dependency on any Web browser.

5.4.1 Packaging and installation

The Service Update Management Assistant is available by default after any AIX 5L Version 5.3 operating system installation. All SUMA modules and the **suma** command are contained in the bos.suma fileset. SUMA is implemented in the

Perl programming language, so the Perl library extensions fileset perl.libext and the Perl runtime environment fileset perl.rte are prerequisites to bos.suma.

The perl.libext fileset holds additional Perl modules which allow access to AIX-specific function from native Perl code. The provided extensions yield a significant performance improvement over system calls.

SUMA directly benefits from enhancements to the Perl runtime environment in AIX 5L Version 5.3. The new base AIX Perl distribution (perl.rte) adds nine new Perl modules which enable programmatic Internet access and provide the ability to parse and generate XML code.

The **lslpp -p bos.suma** command can be used to verify the requisites for the bos.suma fileset:

```
# lslpp -p bos.suma
  Fileset               Requisites
  ----------------------------------------------------------------------------
Path: /usr/lib/objrepos
  bos.suma 5.3.0.0      *prereq bos.rte 5.1.0.0
                        *prereq perl.rte 5.8.2.0
                        *prereq perl.libext 2.1.0.0

Path: /etc/objrepos
  bos.suma 5.3.0.0      *prereq bos.rte 5.1.0.0
                        *prereq perl.rte 5.8.2.0
                        *prereq perl.libext 2.1.0.0
```

Use the **lslpp -f bos.suma** command to examine the list of directories and files which are required to install the SUMA feature.

```
# lslpp -f bos.suma
  Fileset               File
  ----------------------------------------------------------------------------
Path: /usr/lib/objrepos
  bos.suma 5.3.0.0      /usr/suma/lib/SUMA/FixInventory.pm
                        /usr/suma/bin/suma_fixinv
                        /usr/suma/lib/SUMA/Policy.pm
                        /usr/suma/lib/SUMA/Download.pm
                        /usr/suma/lib/msg.map
                        /usr/suma/bin/sm_suma
                        /usr/suma/lib
                        /usr/suma/lib/SUMA/Scheduler.pm
                        /usr/suma/lib/SUMA/DBMStanzaDir.pm
                        /usr/suma/lib/SUMA
                        /usr/suma/lib/SUMA/SoftwareInventory.pm
                        /usr/suma/bin/suma_mgdb
```

```
                          /usr/suma/bin
                          /usr/suma/lib/SUMA/NotifyCache.pm
                          /usr/suma/bin/suma_swinv
                          /usr/suma/bin/suma
                          /usr/suma
                          /usr/sbin/suma -> /usr/suma/bin/suma
                          /usr/suma/lib/SUMA/PolicyFactory.pm
                          /usr/suma/lib/SUMA/StanzaDB.pm
                          /usr/suma/lib/SUMA/GConfig.pm
                          /usr/suma/lib/SUMA/Util.pm
                          /usr/suma/lib/SUMA/Messenger.pm

Path: /etc/objrepos
  bos.suma 5.3.0.0        /var/suma/tmp
                          /var/suma
                          /var/suma/data
```

5.4.2 Functional description

The Service Update Management Assistant is implemented as a task-oriented utility which supports a comprehensive set of features:

- Automated task-based retrieval of the following fix types and categories:
  - Specific APAR
  - Specific PTF
  - Latest critical PTFs
  - Latest security PTFs
  - All latest PTFs
  - Specific fileset
  - Specific maintenance level
- Task setup supported by SMIT (fast path suma) or command line interface (**/usr/sbin/suma**).
- Customizable task defaults to expedite setup of SUMA tasks.
- Tunable number of concurrent downloads.
- Unattended task activation at flexible intervals through scheduling module (utilizing the **cron** command).
- Three different task actions to initiated download preview, actual code download, or combined download and fix repository cleanup (utilizing the **lppmgr** command).

- Download filtering against local software repository, maintenance level, **lslpp** command output file and installed software on the local host, or a NIM client
- Summary statistics show number of downloaded, failed, and skipped update images
- Support for FTP, HTTP, or HTTPS transfer protocols and proxy servers. (HTTPS requires OpenSSL to be installed from AIX Toolbox for Linux Applications.)
- E-mail notification of update availability and task completion.
- Messaging function providing six verbosity levels (Off, Error, Warning, Information, Verbose, and Debug) that may be uniquely set for sending information to the screen, log file, or e-mail address.

5.4.3 Concepts and implementation specifics

The **suma** command can be considered as the central component of the SUMA implementation; it provides control over all SUMA-related tasks and operations. Therefore we also refer to the **suma** command as the SUMA Controller. The **suma** command Perl script is the main interface among the various SUMA modules and allows for initiation of all SUMA functions. The SUMA Controller does the following:

- Receives task information either from SMIT or directly from the command line and distributes this information to the task, notification, and scheduler modules.
- Handles calls from SMIT or the command line to manage tasks or notifications.
- Handles all cron daemon events when called by cron and initiates the established task.
- Reports error conditions and logs operations performed.

The SUMA controller utilizes certain SUMA modules to execute SUMA operations and functions. Refer to Figure 5-2 for a graphical illustration of the SUMA modules and their interaction and dependencies.

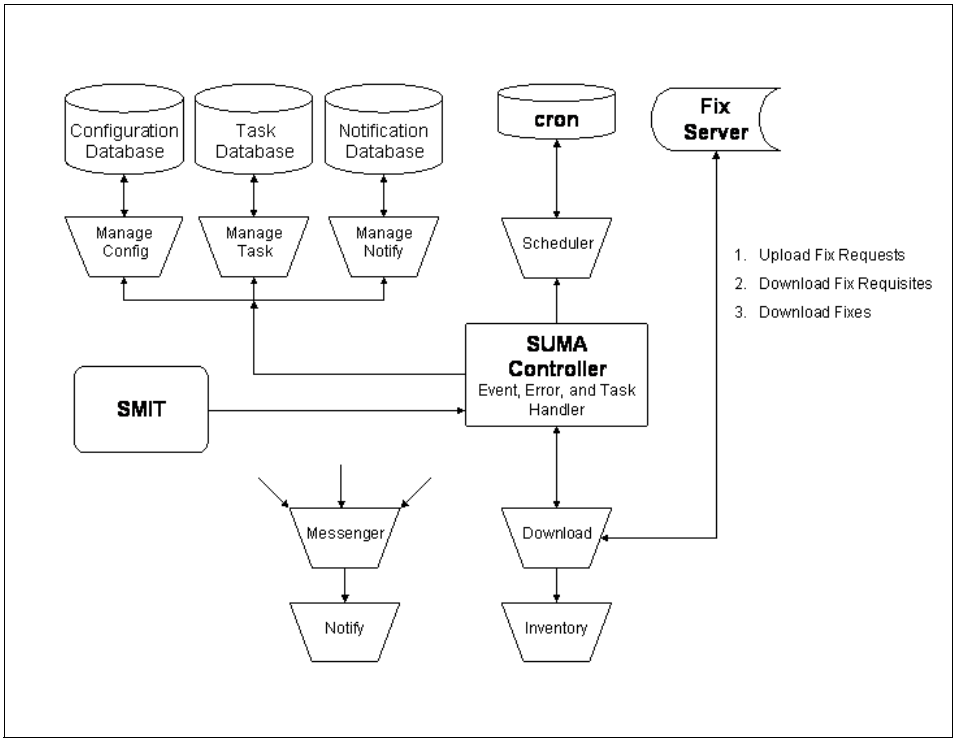

Figure 5-2 Service Update Management Assistant: Control flow diagram

The SUMA modules supply the following services and functions:

Download module

The download module provides functions related to network activities and is solely responsible for communicating with the IBM ℯserver pSeries support server. This communication manifests itself in two different transaction types. In the first, a list of filesets is requested from the fix server based on the SUMA task data passed to the download module. The second consists solely of downloading the requested files from the IBM ℯserver support server.

Manage configuration module

The manage configuration module represents a utility class containing global configuration data and general-purpose methods. These methods allow for the validation of field names and field values since this information is predefined, meaning that there is a known set of supported global configuration fields and their corresponding supported values. This module provides the interface to the global configuration database file.

Messenger module

The Messenger module provides messaging, logging, and notification capability. Messages will be logged (or displayed) when their specified verbosity level is not greater than the threshold defined by the SUMA global configuration.

The log files themselves will be no larger than a known size (by default, 1 MB), as defined by the SUMA global configuration facility. When the maximum size is reached, a backup of the file will be created, and a new log file started, initially containing the last few lines of the previous file. Backup files are always created in the same directory as the current log file. Therefore, minimum free space for log files should be kept in mind.

There are two log files which are located in the /var/adm/ras/ directory. The log file /var/adm/ras/suma.log contains any messages that pertain to SUMA Controller operations. The other log file, /var/adm/ras/suma_dl.log tracks the download history of SUMA download operations and contains only entries of the form DateStamp:FileName. The download history file is appended when a new file is downloaded. The two logs are treated the same with respect to maximum size and creation/definition.

The messenger module relies on contact information (e-mail addresses) from the notification database file which is managed by the notify module.

Notify module

The notify module manages the file which holds the contact information for SUMA event notifications. This database stores a list of e-mail addresses for use by SMIT when populating the list of notification addresses as part of SUMA task configuration.

Task module

SUMA makes use of the task module to create, retrieve, view, modify, and delete SUMA tasks. All SUMA task-related information is stored in a dedicated and private task database file.

Scheduler module

The scheduler module is responsible for handling scheduling of SUMA task execution and interacts with the AIX **cron** daemon and the files in the /var/spool/cron/crontabs directory.

Inventory module

The inventory module returns the software inventory (installed or in a repository) of the local system (localhost) or a NIM client. It covers all software that is in the installp, RPM, or ISMP packaging format.

If the system specified to the module is not local then the system must be a NIM client of the local system.

Utility and database modules

Other modules supply private utilities for SUMA code and utilities for handling the stanza-style SUMA databases.

5.4.4 Command line interface

The **suma** command provides task-related SUMA control functions. The command can be used to perform the following operations on a SUMA task:

- Create
- Edit
- List
- Schedule
- Unschedule
- Delete

The specified operation will be performed on the task represented by a unique task identifier (TaskID). For the create or edit cases on a SUMA task, if the TaskID is not specified, the create operation will be assumed, and a unique TaskID will be generated. The **suma -l** command displays all SUMA tasks if the TaskID is not specified. The **suma -c** command entered without any additional flag will list the SUMA global configuration settings. The usage information of the **suma** command is shown in the following:

```
# suma -?
Usage:
Create, Edit, or Schedule a SUMA task.
        suma { { [-x][-w] } | -s CronSched } [ -a Field=Value ]... [ TaskID ]

List SUMA tasks.
        suma -l [ TaskID ]...
List or Edit the default SUMA task.
        suma -D [ -a Field=Value ]...
List or Edit the SUMA global configuration settings.
        suma -c [ -a Field=Value ]...
Unschedule a SUMA task.
        suma -u TaskID
Delete a SUMA task.
        suma -d TaskID
```

5.4.5 SMIT user interface

All Service Update Management Assistant related tasks and functions are supported by SMIT menus and panels. The new main SUMA menu shown in Figure 5-3 can be directly accessed through the SMIT fast path suma. Alternatively, you can select the new Service Update Management Assistant (SUMA) option in either the Software Maintenance and Utilities or the Software Service Management menu. Both menus are listed under the Software Installation and Maintenance option of the SMIT top-level menu.

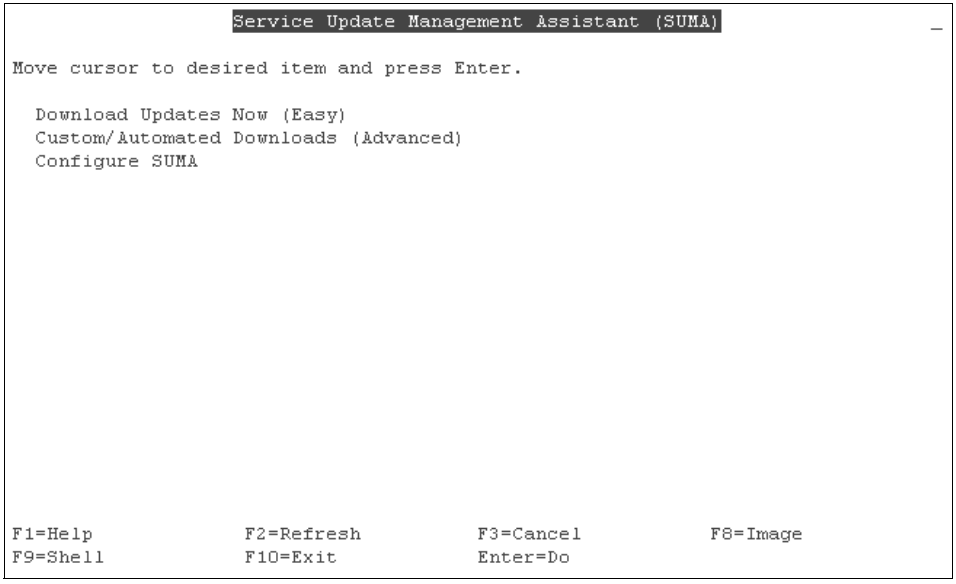

Figure 5-3 New SMIT menu: Service Update Management Assistant

A good starting point for a SUMA novice is potentially the Download Updates Now (Easy) SMIT menu. Some common tasks are readily accessible through this menu of quick-and-easy-options. The options are streamlined for usability, but consequently provide limited access to the full set of SUMA's capabilities. The following selections are offered:

- Download by APAR Number
- Download by Fix Type
- Download Maintenance Level
- Download All Latest Fixes
- Download by Fileset Name

Using the download options of this menu and closely examining the SMIT command output will give some insight into the way SUMA retrieves updates

from the IBM @server support Web site. Use the logging facility of SMIT to study the output in detail.

An experienced SUMA user can directly proceed to use the advanced options under the Custom/Automated Downloads (Advanced) menu. These advanced options allow the system administrator to exert greater control over SUMA's function, and expose many more capabilities not available through the Download Updates Now menu. The user will be required to possess a deeper understanding of SUMA and the options available. The options under that menu allow management of highly-customizable tasks, which can be scheduled for later or repeating execution, and tailored to fit the exact needs of the system environment. The following options are accessible through the advanced menu:

- Create a New Task
- View/Change an Existing Task
- Remove a Task
- View All Tasks

Figure 5-4 on page 174 shows the new SMIT panel to create a new SUMA task.

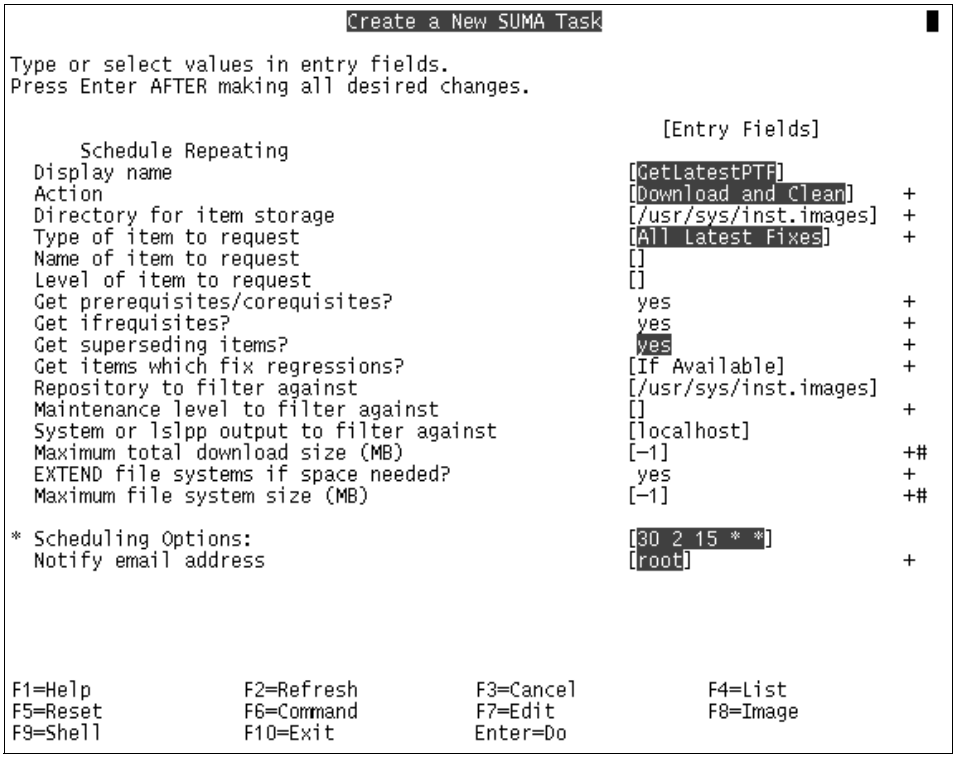

Figure 5-4 SMIT panel: Create a New SUMA Task

The third option under the main SUMA SMIT menu allows the system administrator to customize the SUMA configuration and task defaults. The Configure SUMA SMIT menu provides control over configuration options whose settings affect all of the task-oriented menus previously described. Three options are available for selection:

- Base Configuration
  Settings such as verbosity level, **lppmgr** flags, Internet protocols, log file maximum size, and download time out can be configured here.
- Task Defaults
  This panel allows the system administrator to set up a basic task profile whose settings will affect both the easy and advanced actions in the previously described menus. The options under the Download Update Now (Easy) menu have these default settings fixed, under the covers, and not configurable within the panels themselves. In the advanced task panels, these settings appear as defaults when a new task is being created.

- Saved E-mail Addresses
  The underlying panels allow the system administrator to view and delete entries of notification e-mail addresses.

Table 5-2 provides a complete overview of all available SUMA-related SMIT menus and panels.

Table 5-2 SUMA SMIT menus and panels

SMIT menu or panel title	Fast path	Type
Service Update Management Assistant (SUMA)	suma	Menu
Download Updates Now (Easy)	suma_easy	Menu
Download by APAR Number	suma_easy_apar	Panel
Download by Fix Type	suma_easy_fixtype	Panel
Download Maintenance Level	suma_easy_ml	Panel
Download by Fileset Name	suma_easy_fileset	Panel
Custom/Automated Downloads (Advanced)	suma_task	Menu
Select an Action to Perform	suma_task_new	Panel
View/Change an Existing SUMA Task	suma_task_edit	Panel
Remove a SUMA Task	suma_task_delete	Panel
View/Change SUMA Task Defaults	suma_task_defaults	Panel
Configure SUMA	suma_config	Menu
Base Configuration	suma_config_base	Panel
View/Change SUMA Task Defaults	suma_task_defaults	Panel
Saved Email Addresses	suma_notify	Menu
Remove a Saved Email Address	suma_notify_remove	Panel

5.5 Long user and group name support

AIX 5L Version 5.3 provides support for long user and group names. Prior to Version 5.3, there is a limit of eight characters for user and group names. Now the user and group name length is increased from 8 to 255. Longer than eight character user or group name capability can be enabled on the system by changing the system-wide configuration parameter using the SMIT panels. Note that the change needs to be done after careful analysis and consideration. Be

sure to determine the right size before setting the size. It is extremely difficult to reduce the size of the user or group names once increased because there will be existing users with longer names who might not be able to log in if the limits are reduced. A size should be chosen that it meets the requirements of your particular site (across multiple systems and platforms).

Verify and change user name limit

Before changing the user name limit, you may want to verify your user name limit. Enter **`smit chgsys`** to view the new attribute related to user name limit as highlighted in Figure 5-5.

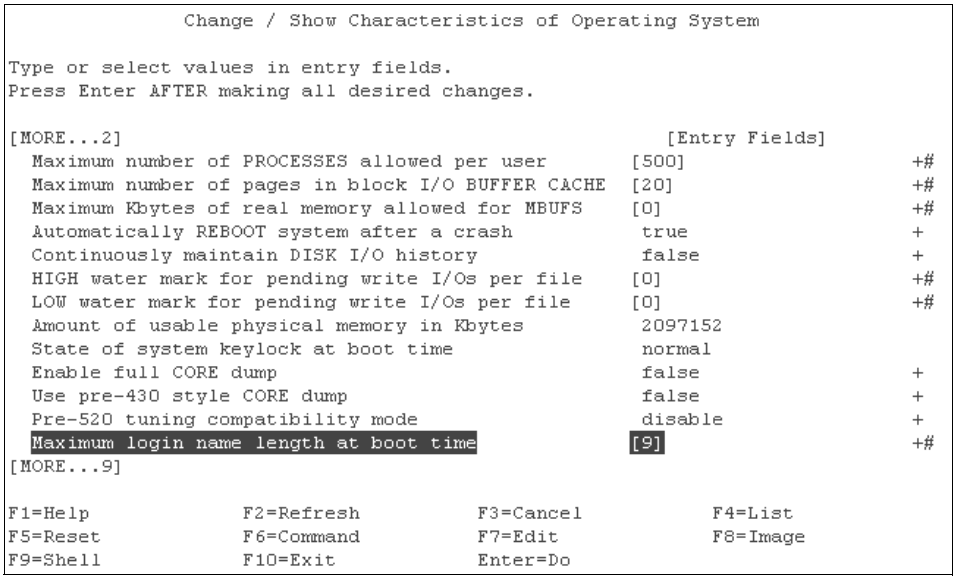

Figure 5-5 Change/Show Characteristics of Operating System

Notice that the Maximum login name length at boot time attribute is set to 9, but actually it is an 8-character user name limit since it has a terminating NULL character. You will see an error message if you try to create a user with a name longer than 8 characters in length before you change the default. Modify the value in this field to your desired number. You must reboot the system for changes to take effect.

Example 5-3 on page 177 shows the steps used to verify and change the user name limit through the command line if you wish to change the user name limit to 20 characters. It should be noted that the current user name limit (output of the **`getconf`** command) and the limit defined in ODM (output of the **`lsattr`** command) could be different before you reboot the system to make the changes effective.

Example 5-3 Verify and change user name limit through the command line

```
# mkuser slson1234567890
3004-694 Error adding "slson1234567890" : Name is too long.

# lsattr -El sys0 -a max_logname
max_logname 9 Maximum login name length at boot time True

# chdev -l sys0 -a max_logname=21
sys0 changed
# lsattr -El sys0 -a max_logname
max_logname 21 Maximum login name length at boot time True

# getconf LOGIN_NAME_MAX
9
# shutdown -Fr
```

After you reboot the machine, you are ready to make a user name with more than 8 characters. Create a user named *slson1234567890* and set a password for the user.

Example 5-4 Create a long user name

```
# lsattr -El sys0 -a max_logname
max_logname 21 Maximum login name length at boot time True
# getconf LOGIN_NAME_MAX
21

# mkuser slson1234567890
# passwd slson1234567890
Changing password for "slson1234567890"
slson1234567890's New password:
Enter the new password again:
#
```

Considerations

If you decrease the limit to the original value while you have long user names defined, the command will not tell you which user names are too long. After you reboot the system, if there are longer names used than the new lower limit allows, those user names cannot be used for login. This is illustrated in Example 5-5.

Example 5-5 Cannot log in with longer user name because of the new lower limit

```
# chdev -l sys0 -a max_logname=9
sys0 changed
# lsattr -El sys0 -a max_logname
max_logname 9 Maximum login name length at boot time True
```

```
# getconf LOGIN_NAME_MAX
21

#shutdown -Fr
# login slson1234567890
slson1234567890's Password:
[compat]: 3004-616 User "slson123" does not exist.

*******************************************************************************
*                                                                             *
*  Welcome to AIX Version 5.3!                                                *
*                                                                             *
*                                                                             *
*  Please see the README file in /usr/lpp/bos for information pertinent to    *
*  this release of the AIX Operating System.                                  *
*                                                                             *
*******************************************************************************

3004-614 Unable to change directory to "".
        You are in "/home/guest" instead.
$
```

5.6 Dynamic reconfiguration usability

Dynamic reconfiguration (DR) implemented with AIX 5L Version 5.2 allows physical resources such as memory and processors to be added to or deleted from a logical partition (LPAR) without rebooting the partition. To enhance the DR usability the new **cpupstat** command was added in AIX 5L Version 5.3 to provide information to the user on configurations preventing DR operations from being successful.

A processor dynamic reconfiguration remove request can fail for a variety of reasons. The most common of these is that the resource is busy due to an active processor binding. The operating system cannot ignore processor bindings and carry on DR operations or applications might not continue to operate properly. To ensure that this does not occur, the system administrator needs to release the binding, establish a new one, or terminate the application. The specific process or threads that are impacted is a function of the type of binding that is used.

There are several different commands, subroutines, and methods which can be used to establish a processor binding:

attachrset command	Attaches a resource set (rset) to a process
binprocessor command	Binds or unbinds the kernel threads of a process to a processor

bindprocessor() subroutine	Binds kernel threads to a processor when used by an application
WLM software partitions	Workload Manager (WLM) enforced binding to processor resource sets based on WLM class attributes
wlm_set() subroutine	Sets or queries the Workload Manager (WLM) state from within an application and can be used to request that WLM takes the resource set bindings into account

AIX 5L Version 5.3 offers the new **cpupstat** command to analyze the system for processor bindings and detect configurations that could cause a CPU dynamic reconfiguration operation to fail. The **cpupstat** command is part of the bos.rte.commands fileset which is always installed on an AIX system. The command can be executed by any user.

If you enter the command without any additional parameters the following usage message is displayed:

```
# cpupstat
0560-001 Usage : cpupstat [-v] -i LogicalCPU.
```

The command requires the index of a logical CPU ID as input; the optional -v flag provides verbose output when specified.

The command will run through three consecutive stages:

1. Check all the WLM class rsets for rsets with only one single active CPU matching the passed in logical CPU ID. A count of the number of classes with such an rset is printed. If the verbose option is given, the names of the classes will be printed as well.
2. Check all the kernel registry rsets for rsets with only one single active CPU matching the passed in logical CPU ID. A count of the number of processes with attachments to such rsets will be printed to the user. If the verbose option is given, the process IDs will be printed as well.
3. A count of the **bindprocessor** command attachments for the highest numbered bind ID will be printed for the user. If the verbose option is given, the process IDs will be printed as well.

The highest bind ID will always be removed if a processor DR removal operation succeeds. Consequently the count of bindprocessor attachments for the highest numbered bind ID will be determined regardless of the specified index of a logical CPU.

The **cpupstat** command can be invoked directly from the command line but it will also be executed after a failed CPU DR removal, with the argument being the logical CPU that failed to be removed. Thus all the necessary information is provided to permit removal of the processor bind related inhibitors to the DR operation.

5.7 Paging space garbage collection

Ever increasing demands in resource capacity, scalability, and flexibility require modifications and enhancements to core AIX functions, algorithms, and services. This general observation also applies for AIX paging space allocation policy and management.

AIX Version 4.3.2 introduced the deferred paging space allocation policy, which ensures that no disk block is allocated for paging until it becomes necessary to pageout a given memory page. This allocation policy was a modification and enhancement to the existing late allocation policy which initiated a disk block allocation once a memory page was requested and accessed, regardless of the need to pageout that frame. The early paging space allocation policy offers an alternative to the deferred paging space allocation policy. If the former policy is chosen the paging space is allocated when memory is requested – even before the frame is accessed or scheduled for pageout. The deferred allocation policy reduces the amount of required paging space substantially.

Independent of the allocation policy in use, the paging space slots were only released by process termination or by the disclaim() system call. The free() system call only releases allocated paging space if the MALLOCDISCLAIM environment variable is set to true.

The later behavior may no longer be desirable for modern applications and systems capable of dynamic resource optimization. The following two scenarios illustrate this statement:

- Consider the class of applications used in a large real-memory environment and which are long running in nature. An application of that class potentially runs in memory; however, burst activity occasionally will generate some paging allocation. These pages eventually get re-paged in, and with the Virtual Memory Manager's (VMM) allocation policy of the previous AIX releases, retain their disk blocks. Thus they may stay in memory for the life span of the related processes, but use up paging space needlessly. This class of applications may literally be running for months, and cannot tolerate down time. But if, over time, an application runs out of paging space, the operating system may be forced to kill the application to maintain system integrity.

- Consider a similar environment to the one just described. A long-lived application runs in a logical partition; however, periodic dynamic reconfiguration memory removes perform load balancing and send memory to another partition for a relatively short amount of time. That memory may be returned when the partition that was the target of the DR memory operation is done with its work. Frequent small DR operations may take place as partitions are spawned to perform one single task, then exit. The result of this memory remove can and frequently will be paging space allocation. If memory is subsequently re-added, these disk blocks are not freed, and this can cause pressure on paging space, in spite of having a reasonable amount of main memory.

AIX 5L Version 5.3 addresses these issues and introduces enhanced paging space management algorithms to reclaim, or garbage collect (GC) paging space as needed.

Version 5.3 implements two paging space garbage collection (PSGC) methods for deferred allocation working segments:

- Garbage collect paging space on re-pagein
- Garbage collect paging space scrubbing for in-memory frames

The following modes and environments are supported:

- LPAR and SMP mode
- 32-bit and 64-bit multi-processor kernel
- All currently supported hardware platforms
- Deferred allocation working storage segments only

5.7.1 Garbage collection on re-pagein

AIX 5L Version 5.3 introduces a dynamic decision-making mechanism, whereby pagein will make a determination automatically, whether or not to free the disk block (at the end of pagein) based upon whether paging space is low or not.

This garbage collection method is also referred to as the re-pagein disk block free mechanism. This mechanism only applies to the deferred page space allocation policy. Normally, the deferred page space allocation policy will, upon paging a page back in from disk, retain its disk block, and leave the page unmodified in memory. If the page does not get modified then, when it is selected for replacement by the VMM it does not have to be written to disk again. This is a performance advantage for pages that are not modified after being paged back in from disk. The re-pagein disk block free mechanism allows the system administrator to specify that the operating system should free a page's disk block after pagein. The page will be left modified in memory. If the page gets selected

for replacement by the VMM, it will have a new disk block allocated for it, and it will be written to disk. There are two advantages to this:

- Paging space disk blocks are conserved. Pages in memory do not have unnecessary disk space committed to them.
- If such pages are selected for replacement by the VMM, new disk blocks will be allocated for them. In the event that the Virtual Memory Manager is paging out numerous pages at once, it is quite possible that these disk blocks may be allocated adjacent or nearby on the paging device. This can improve the performance of writing out the pages. If they had retained their original disk locations, and had been modified after pagein, then if they get paged out along with other pages, the disk blocks are likely to be scattered across the paging devices.

This re-pagein mechanism is enabled by default on the operating system when the system becomes low on free paging space disk blocks. At this point, by default, pages that are paged in due to a modification will have their disk blocks freed. A modification could be in the form of a bzero(), for example. Since the page is being modified, there is no significant performance cost to freeing the disk block because, if this page is selected for replacement by the Virtual Memory Manager later on, it would have to be written to disk anyway.

The enhanced **vmo** command provides all tuning and control functions for the re-pagein garbage collection. It gives the system administrator the ability to alter the thresholds of free disk blocks at which the re-pagein disk block free mechanism starts and stops. The mechanism may also be disabled or always enabled. Additionally, the ability to free disk blocks on pageins due to read accesses to a page may be specified.

The **vmo** command allows configuration of the following re-pagein GC parameters:

npsrpgmin Low paging space threshold when re-pagein GC starts.

npsrpgmax High paging space threshold when re-pagein GC stops.

rpgclean Controls if re-pagein GC will be active only for store operations (page modify) or for store as well as load (page read/access) operations.

rpgcontrol Controls if re-pagein GC will be disabled, always enabled, or only enabled if paging space utilization is within the limits of npsrpgmin and npsrpgmax. The scope and behavior of rpgcontrol is influenced by rpgclean.

By default, AIX 5L Version 5.3 will activate the re-pagein garbage collection only if paging space utilization is within the limits of npsrpgmin and npsrpgmax and it will only free paging space disk blocks on pagein of pages that are being modified (load operation).

The thresholds for GC activation are just above the traditional paging space warning thresholds. Note that there is a static initial default value (Init) for the thresholds, but, they are adjusted as paging spaces are added, or grown (Grow), and shrunk (Shrink) as they are removed.

The following list provides an overview of how the new paging space tuning parameters are determined and how they depend on the traditional paging space thresholds for npskill and npswarn.

npskill	Init(128 psdb) Grow(MAX(npskill, numpsblks/128)) Shrink(MAX(128 psdb, numpsblks/128))
npswarn	Init(512 psdb) Grow(MAX(npswarn, npskill*4)) Shrink(MAX(512 psdb ,npskill*4))
npsrpgmin	Init(768 psdb) Grow(MAX(npsrpgmin, npswarn+(npswarn/2))) Shrink(MAX(768 psdb, npswarn+(npswarn/2)))
npsrpgmax	Init(1024 psdb) Grow(MAX(npsrpgmax, npswarn*2)) Shrink(MAX(1024 psdb, npswarn*2))

The preceding examples used the following definitions:

psdb	Paging space disk blocks (4096 bytes)
numpsblks	Number of paging space blocks

In the example, 768 blocks are 3 MB, while 1024 are 4 MB. The default npswarn value is 2 MB (512 blocks), but it is adjusted when a paging space grows or is added, to where it attempts to be (npskill*4). Note, as shown, that npsrpgmin and npsrpgmax will be adjusted in lockstep automatically with npswarn and npskill upon adding and removing paging space.

Refer to “VMM tuning parameters for PSGC” on page 185 for additional details about the **vmo** command GC controls and thresholds.

Early paging space allocation segments (PSALLOC=early) reserve all of their paging space disk blocks when they are created, or marked for early allocation. If the system frees an allocated disk block in an early allocation segment, the reserved amount of paging space consumption will not be reduced and the global system paging space count does not get updated. Because of this behavior, early allocation segments are ignored by the re-pagein GC algorithm.

5.7.2 GC paging space scrubbing for in-memory frames

The re-paging disk block free mechanism is supplemented by a more powerful but less granular method. Instead of, or in addition to the re-pagein disk block free mechanism, already discussed in “Garbage collection on re-pagein” on page 181, a system administrator might want to try and reclaim paging space disk blocks for pages that are already in memory.

When using the deferred page space allocation policy, pages that are paged in from paging space retain their paging space disk block while they are in memory. If the page does not get modified, then when it is selected for replacement by the Virtual Memory Manager, it does not have to be written to disk again. This is a performance advantage for pages that are not modified after being paged back in from disk. Over time, however, after many pageins, it is possible that there could be many pages in memory that are consuming paging space disk blocks. This could cause paging space to become low. To address this problem, the system administrator, through the **vmo** command, can enable a mechanism to free paging space disk blocks from pages in memory that have them. This is referred to as garbage collection, or scrubbing of these disk blocks.

The **vmo** parameters scrub, scrubclean, npsscrubmin and npsscrubmax control the garbage collection paging space scrubbing activity as follows:

npsscrubmin — Low paging space threshold for GC paging scrubbing to start. The npsscrubmin will default to npsrpgmin.

npsscrubmax — High paging space threshold for GC paging space scrubbing to stop. The npsscrubmax threshold will default to npsrpgmax

scrubclean — Controls if GC paging space scrubbing will free disk blocks for unmodified pages as well as modified. By default, only paging space disk blocks for modified pages in memory are freed.

scrub — Controls if GC paging space scrubbing will be enabled or disabled. By default paging space scrubbing is disabled. If it is enabled, scrubbing of in-memory paging space disk blocks will be activated when the number of system free paging space blocks is below npsscrubmin, and continues until above npsscrubmax.

Note that there is a static initial default value (Init) for the thresholds, but, they are adjusted as paging spaces are added, or grown (Grow), and shrunk (Shrink) as they are removed.

The following list provides an overview of how the new paging space scrubbing tuning parameter defaults are computed:

npsscrubmin Init(768 psdb)
Grow(MAX(npsscrubmin, npsrpgmin))
Shrink(MAX(1024 psdb, npsrpgmin))

npsscrubmax Init(1024 psdb)
Grow(MAX(npsscrubmax,npsrpgmax))
Shrink(MAX(1024 psdb, npsrpgmax))

The preceding examples used the following definition:

db Paging space disk blocks (4096 bytes))

If the GC paging space scrubbing is enabled, an internal timer will trigger a kernel service to start the garbage collection process every 60 seconds when the number of system free paging space blocks are within the limits of the lower and upper scrubbing thresholds.

5.7.3 VMM tuning parameters for PSGC

In support for the new re-pagein disk free mechanism and the garbage collection paging space scrubbing, eight new controls and tuning parameters can be configured by the **vmo** command. Descriptions, the default values, and the value ranges for the new parameters are listed in this section. The traditional npskill and the npswarn parameters are also listed for completeness and for your reference. Consult the AIX product documentation for a detailed and full description of the **vmo** command.

npskill Specifies the number of free paging-space pages at which the operating system begins killing processes.
Default: MAX(64, number of paging space pages/128).
Range: 0 to total number of paging space pages on the system.

npswarn Specifies the number of free paging-space pages at which the operating system begins sending the SIGDANGER signal to processes.
Default: MAX(512, 4*npskill)
Range: 0 to total number of paging space pages on the system.

npsrpgmax Specifies the number of free paging space blocks at which the operating system stops freeing disk blocks on pagein of deferred page space allocation policy pages.
Default: MAX(1024, npswarn*2).
Range: 0 to total number of paging space blocks in the system.

npsrpgmin Specifies the number of free paging space blocks at which the operating system starts freeing disk blocks on pagein of deferred

page space allocation policy pages.
Default: MAX(768, npswarn+(npswarn/2)).
Range: 0 to total number of paging space blocks in the system.

npsscrubmax — Specifies the number of free paging space blocks at which the operating system stops scrubbing in-memory pages to free disk blocks from deferred page space allocation policy pages.
Default: MAX(1024, npsrpgmax).
Range: 0 to total number of paging space blocks in the system.

npsscrubmin — Specifies the number of free paging space blocks at which the operating system starts scrubbing in-memory pages to free disk blocks from deferred page space allocation policy pages.
Default: MAX(768, npsrpgmin).
Range: 0 to total number of paging space blocks in the system.

rpgclean — Enables or Disables freeing paging space disk blocks of deferred page space allocation policy pages on read accesses to them.
Default: 0, free paging space disk blocks only on pagein of pages that are being modified (write).
Range: 0-1
1, Free paging space disk blocks on pagein of a page being modified (write) or accessed (read).

rpgcontrol — Enables or disables freeing of paging space disk blocks at pagein of deferred page space allocation policy pages.
Default: 2, always enables freeing of paging space disk blocks on pagein, regardless of thresholds. The value of rpgclean will determine the scope of this setting. That means, by default, just write accesses are always processed and read accesses will only be additionally processed if rpgclean is 1.
Range: 0-3

0 - Disables freeing of paging space disk blocks on pagein.
1 - Enables freeing of paging space disk blocks when the number of system free paging space blocks is below npsrpgmin, and continues until above npsrpgmax. The value of rpgclean will determine the scope of this setting. That means, by default, just write accesses are always processed and read accesses will only be additionally processed if rpgclean is 1.
3 - Always enables freeing of paging space disk blocks on pagein of pages that are being modified (write). If rpgclean is set to 1, re-pagein GC will additionally free paging space disk blocks of pages that were only accessed (read), but the GC activity will only take place when the number of system free paging space blocks is within the threshold limits. Therefore option 3 can be considered to be a combination of options 1 and 2.

scrub	Enables or disables freeing of paging space disk blocks from pages in memory for deferred page space allocation policy pages. Default: 0, Disables scrubbing completely. Range: 0-1, 1 enables scrubbing of in memory paging space disk blocks when the number of system free paging space blocks is below npsscrubmin, and continues until above npsscrubmax.
scrubclean	Enables or Disables freeing paging space disk blocks of deferred page space allocation policy pages in memory that are not modified. Default: 0, Free paging space disk blocks only for modified pages in memory. Range: 0-1, 1, Free paging space disk blocks for modified or unmodified pages.

5.8 Dynamic support for large page pools

The AIX support for large page architecture has been available since AIX 5L Version 5.1. A white paper about this topic is on the Web at:

http://www-1.ibm.com/servers/aix/whitepapers/large_page.html

AIX 5L Version 5.3 allows dynamic, runtime resizing of the large pools without the need for a system reboot. In previous versions large pools could be changed only through system reboot.

The large pages are changed by the **vmo** command changing the lgpg_size and the lgpg_regions attribute. Example 5-6 on page 188 shows the procedure to change the large page pools dynamically. In the example we first check the lgpg_regions and lgpg_size attributes of the VMM and then set them to two regions of 16 MB size, which totals to 32 MB large pool size. You can check the allocated large pages (alp) and the free large pages (flp) with the **vmstat -l** command.

Example 5-6 lgpg command example

```
# vmo -o lgpg_regions -o lgpg_size
lgpg_size = 0
lgpg_regions = 0
# vmo -o lgpg_regions=2 -o lgpg_size=16777216
Setting lgpg_size to 16777216
Setting lgpg_regions to 2
# vmstat -l
System configuration: lcpu=4 mem=512MB ent=0

kthr    memory              page              faults              cpu            large-page
----- ----------- ------------------------ ------------ ----------------------- -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa    pc    ec   alp   flp
 1  1 65587 45851   0   0   0   0    0   0  14  273  75  0  0 99  0  0.01   1.1     0     2
```

Setting the lgpg_regions and lgpg_size back to zero disables use of the large page pools.

There are two different conversion operations when changing the large page segment size:

- The first conversion operation creates large pages from 4 KB pages when increasing the large page pool. If there are no free pages, the VMM migrates the 4 KB memory area to the 16 MB large page area.
- The second conversion operation creates 4 KB pages from large pages when decreasing the large page pool. When converting from large pages to 4 KB pages, only large pages that are not in-use (free large pages) can be converted into 4 KB pages.

In case of an error, the conversion stops. The already converted pages will remain converted.

The dynamic large page pool function is only available on machines that support DLPAR. There are some environments where these machines still cannot use DLPAR because of a DLPAR unsafe kernel extension.

5.9 Interim Fix Management

Establishing a unified framework for systems management with a common terminology across various IBM @server products is one of the long-term strategies which drives IBM's ongoing convergence effort. As a consequence of this strategy, AIX Emergency Fix Management has been renamed to *Interim Fix Management* in AIX 5L Version 5.3.

Interim fixes serve only as a temporary solution to an identified software problem to bridge the time until the program temporary fix (PTF) will have been developed and successfully passed regression test. Thus, the term *interim fix* and the term `emergency fix` are equivalent and can be used interchangeably.

To accommodate the name change, AIX 5L Version 5.3 introduces two new hard links to the **emgr** and the **epkg** command which will be established during the installation of the bos.rte.install fileset. The links introduce the **fixmgr** command as a synonym for **emgr** and the **fixpkg** command as a synonym for **epkg**. You can use the **lslpp** command to identify the new members of the command family:

```
# lslpp -f bos.rte.install | grep /usr/sbin/fix
                        /usr/sbin/fixmgr -> /usr/sbin/emgr
                        /usr/sbin/fixpkg -> /usr/sbin/epkg
```

No functional changes were implemented in conjunction with the fix management renaming.

5.10 List installed filesets by bundle

AIX supports the concept of software bundles. A software bundle comprises a group of software packages that support a particular environment or major functional complex and therefore are supposed to be installed together as one unit. The software packages are listed in the associated bundle definition file whose file name has to end with the .bnd extension. AIX provides a set of predefined bundles in the /usr/sys/inst.data/sys_bundles directory. User defined bundles should be placed in the /usr/sys/inst.data/user_bundles directory because SMIT and the Web-based System Manager will look by default in this directory to populate relevant selection lists.

In AIX releases prior to AIX 5L Version 5.3 no one-step facility existed to enable a system administrator to list installable software packages based on the contents of a bundle file. Version 5.3 adds a new feature to the **lslpp** command that allows the administrator to view the state of the installable software packages listed in a bundle file.

lslpp command line interface changes

The **lslpp** command displays information about installed filesets or fileset updates. The enhanced command implementation provides the new -b flag which has to be immediately followed by the value of the File parameter. The following paragraphs show the syntax statement of the **lslpp** command manual page and describe the new flag and parameter in more detail.

```
...
lslpp Command

Purpose
   Lists installed software products.

Syntax

   lslpp { -d | -E | -f | -h | -i | -l | -L | -p } ] [ -a] [ -c] [ -J ] [-q ]
   [ -I ] [ -O { [ r ] [ s ] [ u ] } ] [ [ FilesetName ... | FixID ... | -b
   File | all ]

   lslpp -w [ -c ] [ -q ] [ -O { [ r ] [ s ] [ u ] } ] [ FileName ... | all ]

   lslpp -L -c [ -v]

   lslpp -S [A|O]

   lslpp -e
...
```

The FilesetName and the FixID parameter have been present in AIX releases prior to Version 5.3. The FilesetName parameter is the name of a software product or package. The FixID (also known as PTF or program temporary fix ID) parameter specifies the identifier of an update to a formatted fileset. The File parameter is new to AIX and specifies a bundle file to use as a fileset list for input.

-b File Specifies a bundle file to search for fileset names. The filesets listed in the bundle are then listed as if they had been specified explicitly as FilesetName parameters. To mimic installp behavior, the installp image names are automatically wildcarded. For example, a bundle file entry of I:bos.abc will behave as if bos.abc* was specified as a FilesetName parameter.

If the file does not reside in one of the known bundle locations, the full path and file name, including extension, must be specified. /usr/sys/inst.data/sys_bundles/ and /usr/sys/inst.data/user_bundles/ are the known locations for bundle files.

The **`lslpp -l -b App-Dev`** command provided in the following example shows the state of all filesets which belong to the predefined application development system bundle /usr/sys/inst.data/sys_bundles/App-Dev.bnd. Three filesets are listed as not installed at the end of the output. This fact may indicate that the App-Dev bundle was never used in an install operation.

```
# lslpp -l -b App-Dev
  Fileset                      Level  State      Description
  ----------------------------------------------------------------------------
Path: /usr/lib/objrepos
  bos.adt.base               5.3.0.0  COMMITTED  Base Application Development
                                                 Toolkit
  bos.adt.include            5.3.0.0  COMMITTED  Base Application Development
                                                 Include Files
  bos.adt.lib                5.3.0.0  COMMITTED  Base Application Development
                                                 Libraries
  bos.net.tcp.adt            5.3.0.0  COMMITTED  TCP/IP Application Toolkit
  bos.perf.diag_tool         5.3.0.0  COMMITTED  Performance Diagnostic Tool
  bos.perf.libperfstat       5.3.0.0  COMMITTED  Performance Statistics Library
                                                 Interface
  bos.perf.perfstat          5.3.0.0  COMMITTED  Performance Statistics
                                                 Interface
  bos.perf.proctools         5.3.0.0  COMMITTED  Proc Filesystem Tools
  bos.perf.tools             5.3.0.0  COMMITTED  Base Performance Tools
  bos.perf.tune              5.3.0.0  COMMITTED  Performance Tuning Support
  xlC.cpp                    6.0.0.0  COMMITTED  C for AIX Preprocessor

Path: /etc/objrepos
  bos.perf.diag_tool         5.3.0.0  COMMITTED  Performance Diagnostic Tool
  bos.perf.libperfstat       5.3.0.0  COMMITTED  Performance Statistics Library
                                                 Interface
  bos.perf.perfstat          5.3.0.0  COMMITTED  Performance Statistics
                                                 Interface
  bos.perf.tools             5.3.0.0  COMMITTED  Base Performance Tools
  bos.perf.tune              5.3.0.0  COMMITTED  Performance Tuning Support
lslpp: 0504-132  Fileset bos.loc.adt* not installed.
lslpp: 0504-132  Fileset bos.net.nfs.adt* not installed.
lslpp: 0504-132  Fileset Java14.debug* not installed.
```

SMIT interface support

AIX 5L Version 5.3 provides the SMIT dialog List Installed Software by Bundle in support of the new lslpp feature. You can access this menu directly by the use of the SMIT fast path list_installed_sw_bnd or along the following path through the SMIT menu hierarchy: **smit** → **Software Installation and Maintenance** → **List Software and Related Information** → **List Installed Software and Related Information** → **List Installed Software by Bundle**.

5.11 Configuration file modification surveillance

The AIX 5L Version 5.3 enhanced **geninstall** command provides an easy way for system administrators to see what modifications have been made to any of the configuration files listed in the /etc/check_config.files file. When an installation or update operation changes these files, the differences between the old and new files are recorded in the /var/adm/ras/config.diff output file. If the /etc/check_config.files file requests that the old file be saved, the old file can be found in the /var/adm/config directory.

Version 5.3 ships a default /etc/check_config.files file which can be modified by the system administrator at any time. The **cat /etc/check_config.files** command output that follows shows an example of its contents:

```
# cat /etc/check_config.files
# Watched File List
# d = delete old file; s = save old file
# Old files are saved to /var/adm/config
# This file may be edited
d /etc/check_config.files
d /etc/csh.cshrc
d /etc/csh.login
d /etc/diskpartitions
d /etc/environment
d /etc/filesystems
d /etc/group
s /etc/inittab
d /etc/motd
d /etc/passwd
d /etc/profile
d /etc/rc
d /etc/services
d /etc/shells
d /etc/swapspaces
d /etc/vfs
```

The first few lines of this example check_config.files file are informational comments that are followed by a two column list. All comment lines have to begin with the # character and blank lines are ignored. The first column of any other line has to start with a letter d or s in either upper or lower case. The second column holds the full path to the configuration file being monitored during the install or update process.

If an s (either case) is specified, the old (pre-installation) version of the file will be saved at /var/adm/config, retaining its prior path information. (For example: /etc/inittab would be saved to /var/adm/config/etc/inittab). If a saved version of a

configuration file already exists in the /var/adm/config directory, the existing version will be renamed by appending a process ID which is related to the current **geninstall** command process.

If a d (either case) is specified, the pre-installation version of the file will not be saved in /var/adm/config.

If neither d nor s is specified, that is, only the configuration file name is given, the file will be treated as if d was specified in the first column.

Upon completion of the **geninstall** command execution, the new configuration files will be compared against those saved in /var/adm/config/. If any of the files have changed during the course of installation, a message will be printed in the installation summary section stating which file (or files) changed, along with a concluding message, stating that one or more files have changed. The message also gives the location of the difference log /var/adm/ras/config.diff. The messages are either displayed at the system console or are captured by the smit.log if the installation process was started by SMIT.

An excerpt of a sample smit.log file that was written during the installation of LDAP server support on an AIX 5L Version 5.3 system follows.

```
...

Finished processing all filesets.  (Total time:  5 mins 23 secs).

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

Installation Summary
--------------------
Name                        Level           Part        Event       Result
-------------------------------------------------------------------------------
db2_08_01.pext              8.1.1.16        USR         APPLY       SUCCESS
db2_08_01.msg.en_US.iso8859 8.1.1.16        USR         APPLY       SUCCESS
db2_08_01.jhlp.en_US.iso885 8.1.1.16        USR         APPLY       SUCCESS
db2_08_01.db2.engn          8.1.1.16        USR         APPLY       SUCCESS
db2_08_01.das               8.1.1.16        USR         APPLY       SUCCESS
db2_08_01.cs.rte            8.1.1.16        USR         APPLY       SUCCESS
db2_08_01.conv              8.1.1.16        USR         APPLY       SUCCESS
db2_08_01.conn              8.1.1.16        USR         APPLY       SUCCESS
db2_08_01.cnvucs            8.1.1.16        USR         APPLY       SUCCESS
ldap.msg.en_US              5.2.0.0         USR         APPLY       SUCCESS
ldap.html.en_US.man         5.2.0.0         USR         APPLY       SUCCESS
ldap.html.en_US.config      5.2.0.0         USR         APPLY       SUCCESS
ldap.server.java            5.2.0.0         USR         APPLY       SUCCESS
ldap.server.rte             5.2.0.0         USR         APPLY       SUCCESS
```

```
ldap.server.com             5.2.0.0         USR         APPLY       SUCCESS
ldap.server.cfg             5.2.0.0         USR         APPLY       SUCCESS
ldap.server.com             5.2.0.0         ROOT        APPLY       SUCCESS
ldap.server.cfg             5.2.0.0         ROOT        APPLY       SUCCESS
db2_08_01.essg              8.1.1.16        USR         APPLY       SUCCESS

File /etc/group has been modified.
File /etc/inittab has been modified.
File /etc/passwd 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 content of the related /var/adm/ras/config.diff log file is shown in the following example. Note that the /etc/inittab file was only saved to /var/adm/config because the related entry in the /etc/check_config.files begins with the letter s in the first column. The modifications to the configuration files are analyzed and reported by the **diff** command.

```
# cat /var/adm/ras/config.diff
================================================================================
        /etc/group
        The original file was not saved.
2c2
< staff:!:1:ipsec,aroell
---
> staff:!:1:ipsec,aroell,ldap
20a21
> ldap:!:201:ldap
================================================================================
        /etc/inittab
        The original file was saved to /var/adm/config/etc/inittab.
70a71,72
> fmc:2:respawn:/usr/opt/db2_08_01/bin/db2fmcd #DB2 Fault Monitor Coordinator
> ibmdir:2:once:/usr/ldap/sbin/rc.ibmdir > /dev/null 2>&1
================================================================================
        /etc/passwd
        The original file was not saved.
15a16
> ldap:*:202:1::/home/ldap:/usr/bin/ksh
```

Should a /var/adm/ras/config.diff file already exist when beginning an install operation, the old config.diff file will be renamed, with the current **geninstall** command process ID suffixed to the name. This will keep the most current version with a known file name that never changes. Any old files are named for the process that moved them. This is particularly important to longer installs,

such as during migration, when the **installp** command might be called multiple times.

5.12 DVD backup using the mkdvd command

System administrators can use the **mkcd** command of previous AIX releases to create a multi-volume CD or DVD from a **mksysb** or **savevg** backup image. In order to create a DVD ISO9660 backup it is required to specify the **-L** flag of the **mkcd** command to ensure that the image will be in the correct format. To prevent the command from failing because a single flag was omitted on the command line, AIX 5L Version 5.3 provides the new **mkdvd** command. The **mkdvd** command is actually a hard link to the /usr/sbin/mkcd script, but the script has been enhanced to verify the command name which was used to call it. If the command name was **mkdvd** then the internal DVD_FLAG will be set and the image format is automatically adjusted to be suitable for DVDs.

5.13 NIM security

In previous versions of AIX, NIM used **rsh** and **rcmd** commands to perform remote execution of commands on clients. These r-commands were a potential security exposure.

AIX 5L Version 5.3 is enhanced by the **nimsh** environment that is part of the bos.sysmgt.nim.client fileset. It allows the following two remote execution environments:

- NIM service handler for client communication - basic **nimsh**
- NIM cryptographic authentication - OpenSSL

While the basic **nimsh** is an easy to use solution with sufficient security, the OpenSSL provides additional up-to-date cryptographic security. In the following sections we explain both approaches to NIM security.

The original **rsh** or **rcmd** command environments are still supported in Version 5.3 because of compatibility and ease of use reasons.

The nimsh daemon is controlled by the AIX system resource controller. You can check the subsystem with the **lssrc -s nimsh** command. The recommended way to start the nimsh subsystem is by the **nimclient -C** command.

5.13.1 NIM service handler for client communication - NIMSH

The nimsh subsystem is a restricted shell environment that allows only NIM method execution. This increases the security of remote access compared to the **rsh** or **rcmd** command environments.

When the NIM server intends to run a method on the NIM client, it connects to the nimsh subsystem and sends authentication information and the requested method in a similar way as the rcmd() service. When the nimsh authenticates the incoming request, it checks that the request is valid and performs the requested method. If the authentication fails or the method is not registered, the nimsh denies the execution of the request. The use of the service ports and the authentication is explained in the following sections.

Service ports

The client daemon has two ports registered with the Internet Assigned Numbers Authority (IANA). Part of the /etc/services shows the port numbers here:

```
nimsh                3901/tcp            # NIM Service Handler
nimsh                3901/udp            # NIM Service Handler
nimaux               3902/tcp            # NIMsh Auxiliary Port
nimaux               3902/udp            # NIMsh Auxiliary Port
```

The primary nimsh port is used to listen for the incoming service requests. The NIM master selects the port 1023 or the next available lower port to connect to the nimsh (3901) port in the NIM client. Once the service request is accepted, the primary port is used for the stdin and stdout (file descriptor 0 and 1). During the authentication process the NIM master selects another available port that is 1022 and delivers the port number information to the NIM client. The client opens an auxiliary connection from port nimaux (3902) to the port of the NIM master as received from the master. The secondary nimaux port is opened for the stderr (file descriptor 2).

Enabling secondary port

The following procedure describes how to configure existing standalone clients to use the nimsh communication protocol with a secondary port option enabled. By default, nimsh uses a reserved port for returning stderr output during command execution. The default setting allows administrators to specify a specific port for opening behind a firewall, but it can cause performance issues when several connections are attempted in a short amount of time.

When TCP connections are closed, the closing sockets enter TIME_WAIT state. The length of time for this state may last up to 240 seconds depending on system settings. The secondary port option allows you to specify any specific range of ports to cycle through during NIMSH operation.

For firewalls, administrators might want to open a specific range on the firewall, and then for each machine on the internal network, ensure that the port range on the machine coincides with the open range on the firewall. When changing the NIMSH secondary port, you should choose a range of ports outside of the range used for system services. Try using ports 49152 through 65535.

To configure existing standalone clients to use the NIMSH communication protocol with a secondary port range, complete the following steps:

1. Use the **smitty nim_config_services** fast path on the NIM client.
2. Select nimsh as the Communication Protocol used by client.
3. Specify a start value for the secondary port number.
4. Specify an increment value for the secondary port range.

To configure existing standalone clients to use the NIMSH communication protocol with a secondary port range from the command line, complete the following steps:

1. Edit the /etc/environment file. Add the variable `NIM_SECONDARY_PORT=60000:5` to use ports 60000 - 60005 within NIMSH.
2. Use the desired **nimclient** command option to restart the nimsh daemon.

Authentication process

Service requests of the nimsh are handled in a similar fashion to rcmd(). The communicating host (NIM server) builds packets with the following data for authentication:

- Host name of NIM client
- CPUID of NIM client
- CPUID of NIM master
- Return port for secondary (stderr) connection
- Query flag

If the Query flag is set to 1 the nimshd daemon treats the incoming request as a client discovery for information. The following data is returned:

- Default host name obtained from inet0
- Default route obtained from inet0
- Network address obtained from host name
- Network interface obtained from host name

If the query flag is not set, then a request for service (NIM operation) is pushed by the NIM master. The nimshd daemon validates the method request as follows:

1. Verify the host name of the NIM master is the recognized master host name to the client.
2. Check the client CPUID passed in the authentication data. It should match the client's machine ID.
3. Check the master CPUID passed in the authentication data. It should match the master's machine ID stored in the memory. It is read into the memory from the /etc/niminfo file and the **mktcpip -S *primary_nim_interface*** command outputs.
4. Verify that the operation passed in the method is a method in the path /usr/lpp/bos.sysmgt/nim/methods.
5. Check for cryptographic authentication settings.

For additional security, nimsh supports push disablement. When push disablement is set, nimsh does not process any NIM operation controlled by the NIM master. Use the **nimclient** command to enable or disable the push from master.

- You can check the authentication procedure using the **nimquery** command on the master, while at the same time watching the nimsh.log on the client. First start the nimsh subsystem on the client and check that the /etc/niminfo and the **mktcpip** command values are read and logged to the log file:

```
# nimclient -C
0513-059 The nimsh Subsystem has been started. Subsystem PID is 422002.
# cat /var/adm/ras/nimsh.log
Tue Jul 13 17:23:26 2004        /usr/sbin/nimsh: NIM Service Handler
started from SRC
Tue Jul 13 17:23:26 2004        no environment value for NIM_SECONDARY_PORT
Tue Jul 13 17:23:26 2004        value for route is net,-hopcount,0,,0 and
gateway is 9.3.5.41
Tue Jul 13 17:23:26 2004        value for hostname is server3
Tue Jul 13 17:23:26 2004        value for netaddr is 9.3.5.196
Tue Jul 13 17:23:26 2004        value for netif is en0
Tue Jul 13 17:23:26 2004        value for netmask is 255.255.255.0
Tue Jul 13 17:23:26 2004        obtained master's hostname:
NIM_MASTER_HOSTNAME=server4

Tue Jul 13 17:23:26 2004        obtained master's id:
NIM_MASTERID=000C91AD4C00

Tue Jul 13 17:23:26 2004        obtained master's hostname:
NIM_MASTER_HOSTNAME_LIST="server4 server2"
```

```
Tue Jul 13 17:23:26 2004        value for machine id is 0000316A4C00

# mktcpip -S en0
#host:addr:mask:_rawname:nameserv:domain:gateway:cost:activedgd:type:start
server3:9.3.5.196:255.255.255.0:en0:::9.3.5.41:0:no:N/A:no
```

- Next, start a query from the server to the client:

```
s4 # nimquery -ahost=server3
host:server3:addr:9.3.5.196:mask:255.255.255.0:gtwy:9.3.5.41:_pif:en0:_ssl:
no:_psh:no:_res:no:asyn:no:
s4 # nimquery -ahost=server3 -p
==================
NIMSH Client Values
==================
hostname    =  server3
IP address  =  9.3.5.196
subnetmask  =  255.255.255.0
gateway     =  9.3.5.41
interface   =  en0
====================
NIMSH Client Options
====================
SSL Enabled =  no
Push Denied =  no
Restricted Shell =  no
```

- Finally, check the nimsh.log on the client:

```
# cat /var/adm/ras/nimsh.log
...
Tue Jul 13 17:27:56 2004        file descriptor is 13
Tue Jul 13 17:27:56 2004        file descriptor is : 13
Tue Jul 13 17:27:56 2004        family is : 2
Tue Jul 13 17:27:56 2004        source port is : 1023
Tue Jul 13 17:27:56 2004        source addr is : 9.3.5.197
Tue Jul 13 17:27:56 2004        source hostname is : server4
Tue Jul 13 17:27:56 2004        getting 2nd port
Tue Jul 13 17:27:56 2004        count equals 0
Tue Jul 13 17:27:56 2004         got stderr port 0
Tue Jul 13 17:27:56 2004        success: we got 1st write query is 1
Tue Jul 13 17:27:56 2004        success: we got 2nd write local id is
000000000000
Tue Jul 13 17:27:56 2004        success: we got 3rd write remote id is
000C91AD4C00
Tue Jul 13 17:27:56 2004        success: we got 4th write command is
Tue Jul 13 17:27:56 2004        sending ack to client
Tue Jul 13 17:27:56 2004        buffer value is host:server3:
```

```
Tue Jul 13 17:27:56 2004        buffer value is addr:9.3.5.196:
Tue Jul 13 17:27:56 2004        buffer value is mask:255.255.255.0:
Tue Jul 13 17:27:56 2004        buffer value is gtwy:9.3.5.41:
Tue Jul 13 17:27:56 2004        buffer value is _pif:en0:
Tue Jul 13 17:27:56 2004        buffer value is _ssl:no:
Tue Jul 13 17:27:56 2004        buffer value is _psh:no:
Tue Jul 13 17:27:56 2004        buffer value is _res:no:
Tue Jul 13 17:27:56 2004        buffer value is asyn:no:
Tue Jul 13 17:27:56 2004        sending ack to client
Tue Jul 13 17:28:18 2004        file descriptor is 13
Tue Jul 13 17:28:18 2004        file descriptor is : 13
Tue Jul 13 17:28:18 2004        family is : 2
Tue Jul 13 17:28:18 2004        source port is : 1023
Tue Jul 13 17:28:18 2004        source addr is : 9.3.5.197
Tue Jul 13 17:28:18 2004        source hostname is : server4
Tue Jul 13 17:28:18 2004        getting 2nd port
Tue Jul 13 17:28:18 2004        count equals 0
Tue Jul 13 17:28:18 2004         got stderr port 0
Tue Jul 13 17:28:18 2004        success: we got 1st write query is 1
Tue Jul 13 17:28:18 2004        success: we got 2nd write local id is
000000000000
Tue Jul 13 17:28:18 2004        success: we got 3rd write remote id is
000C91AD4C00
Tue Jul 13 17:28:18 2004        success: we got 4th write command is
Tue Jul 13 17:28:18 2004        sending ack to client
Tue Jul 13 17:28:18 2004        buffer value is host:server3:
Tue Jul 13 17:28:18 2004        buffer value is addr:9.3.5.196:
Tue Jul 13 17:28:18 2004        buffer value is mask:255.255.255.0:
Tue Jul 13 17:28:18 2004        buffer value is gtwy:9.3.5.41:
Tue Jul 13 17:28:18 2004        buffer value is _pif:en0:
Tue Jul 13 17:28:18 2004        buffer value is _ssl:no:
Tue Jul 13 17:28:18 2004        buffer value is _psh:no:
Tue Jul 13 17:28:18 2004        buffer value is _res:no:
Tue Jul 13 17:28:18 2004        buffer value is asyn:no:
Tue Jul 13 17:28:18 2004        sending ack to client
```

When comparing the nimquery and the nimsh.log files you can follow the authentication procedure.

Note: The /etc/niminfo is read only at the nimsh startup. If you make changes to this file, the nimsh needs to be restarted.

Configure nimsh

In order to get the nimsh running, the NIM server must be configured, for example through **smit nim_config_env**, and the NIM client should be defined to the NIM server, for example through **smit nim_mkmac** on the NIM master. The

NIM client must be configured into the NIM environment using the **smit niminit** or the **niminit** command on the NIM client. We recommend using SMIT instead of the regular NIM commands because the NIM performs additional checks of the NIM environment. The default settings will create the client definitions to use the **rsh** or **rcmd** commands.

If you want to use the nimsh environment, you have to log in to both the client and the server and define the nimsh for each client.

On the NIM client run the **smit nim_config_services** command and define the nimsh for the communication protocol used by the client. See the highlighted line in Figure 5-6 for an example of **smit nim_config_services** SMIT screen.

```
                    Configure Client Communication Services

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

                                                      [Entry Fields]
* Communication Protocol used by client               [nimsh]                 +

  NIM Service Handler Options
*   Enable Cryptographic Authentication              [disable]               +
      for client communication?

    Install Secure Socket Layer Software (SSLv3)?    [no]                    +
    Install Secure Socket Layer Software (SSLv3)?    [/dev/cd0]               /
                  -OR-
      lpp_source which contains RPM package          []                      +

    Alternate Port Range for Secondary Connections
       (reserved values will be used if left blank)
      Secondary Port Number                          []                       #
      Port Increment Range                           []                      +#



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

Figure 5-6 smit nim_config_services

This SMIT task will update the NIM configuration including the NIM master and it will start the nimshd subsystem. The changes are written to the /etc/niminfo and the /etc/environment files.

To enable the **nimsh** communications, you can alternatively run the **niminit** command on every NIM client as follows:

```
# . /etc/niminfo
# mv /etc/niminfo /etc/niminfo.bak
# niminit -aname=server2 -amaster=server4 -aconnect=nimsh
```

```
nimsh:2:wait:/usr/bin/startsrc -g nimclient >/dev/console 2>&1
0513-044 The nimsh Subsystem was requested to stop.
0513-059 The nimsh Subsystem has been started. Subsystem PID is 442484.
# nimclient -C
0513-059 The nimsh Subsystem has been started. Subsystem PID is 417820.
```

You can start the already configured **nimsh** by running the **nimclient -C** command (note the uppercase -C) on every client. The **nimclient -C** command will not update the /etc/niminfo and the /etc/environment files.

On the NIM server run the **smit nim_chmac** command and check that the Communication protocol used by client has changed to nimsh. See the highlighted line in Figure 5-7 for an example of the **smit nim_chmac** SMIT screen.

```
                    Change/Show Characteristics of a Machine

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

                                                      [Entry Fields]
  Machine Name                                       [server2]
* Hardware Platform Type                             [chrp]                    +
* Kernel to use for Network Boot                     [mp]                      +
  Machine Type                                        standalone
  Network Install Machine State                       currently running
  Network Install Control State                       ready for a NIM opera>
  Primary Network Install Interface
    Network Name                                      network1
    Host Name                                        [server2]
    Network Adapter Hardware Address                 [0]
    Network Adapter Logical Device Name              [ent]
    Cable Type                                        bnc                      +
    Network Speed Setting                            []                        +
    Network Duplex Setting                           []                        +
  IPL ROM Emulation Device                           []                        +/
  CPU Id                                             [0001810F4C00]
  Communication Protocol used by client              [nimsh]                   +
  Comments                                           []

  Force                                               no                       +


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

Figure 5-7 smit nim_chmac

The SMIT menu will update the NIM configuration on the NIM master. Now, you can check the nimsh environment, for example with the **nim -o lslpp server2** command.

To change the client communications for the server2 client on the NIM server from the command line, you can alternatively run the nim command as follows:

```
# nim -o change -aconnect=nimsh server2
```

This command produces the same result as the SMIT task shown in Figure 5-7.

Once the nimshd subsystem is started, it produces log entries in the /var/adm/ras/nimsh.log file. The log is used only for debug purposes.

BOS install and nimsh

You can select the nimsh also for the BOS install operation. At the time of writing there was no SMIT menu to do this. Use the **nim** command as follows:

```
nim -o bos_inst -a connect=nimsh -a source=rte -a spot=spot530 \
 -a lpp_source=aix530 server3
```

5.13.2 NIM cryptographic authentication - OpenSSL

AIX 5L Version 5.3 introduces a cryptographic extension for the NIM. This cryptographic extension is based on the OpenSSL environment as it is delivered on the Linux Toolbox for AIX media. Although you can use other sources for the OpenSSL software, we do not recommend them because the automatic NIM tools are prepared for the appropriate version of OpenSSL as installed from the Linux Toolbox for AIX media.

Note: You will need to install the OpenSSL package from the Linux Toolbox for AIX media directly from the media, or copy it to the lpp_source NIM resource for later installation by NIM tools.

Install OpenSSL

First you need to install the OpenSSL from the Linux Toolbox for AIX package, or copy it to the lpp_source NIM resource. We recommend copying the OpenSSL RPM package from the CD media to the lpp_source NIM resource. Once the OpenSSL RPM package is in the lpp_source, you can easily install the OpenSSL in the NIM network directly from the SMIT screens for NIM.

The minimal tested version of the OpenSSL is 9.6g; version 9.7 should also work fine. In our examples we are using OpenSSL version 9.6m. It is important to have the same or at least compatible versions installed on the server and the clients.

Prepare a file system for /ssl_nimsh

The SSL configuration requires an exchange directory for encryption keys. By default, NIM uses the /ssl_nimsh directory that resides in the / (root) file system.

We recommend creating a file system named /ssl_nimsh, of minimal size (for example 16 MB), and mounting the file system.

Enable SSL for NIM server

You have the choice to create the SSL keys and configuration manually or run the NIM configuration utilities from the **smit nim_ssl** SMIT screen. The utility calls programs from the /usr/samples/nim/ssl directory along with the basic NIM commands. After the utilities finish, the necessary keys are created in the /ssl_nimsh directory. See Figure 5-8 for the SMIT screen that configures the NIM to use the SSL environment.

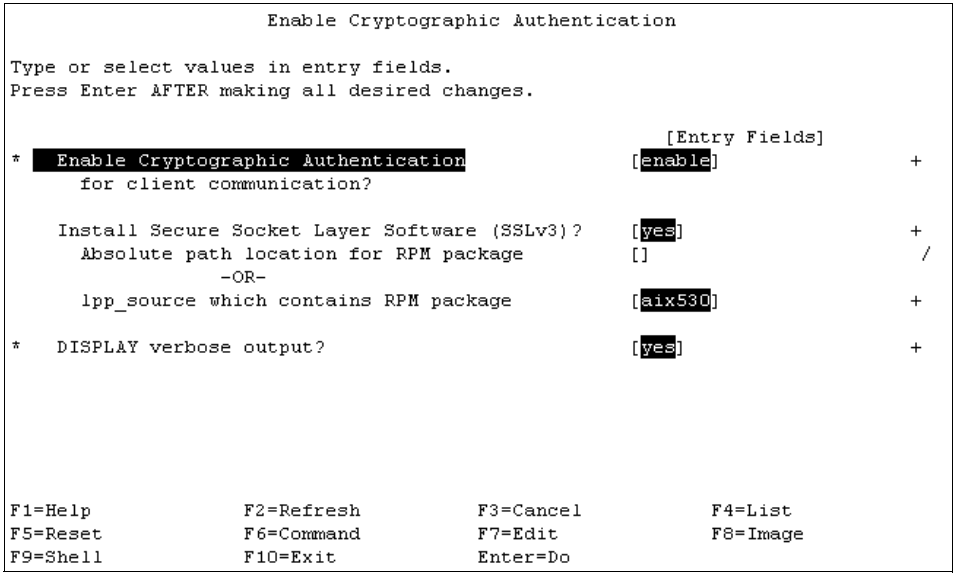

Figure 5-8 smit nim_ssl

As shown in Figure 5-8, setting the Enable Cryptographic Authentication to enable the utility will set up the SSL environment, including creating keys, for the NIM master. If you did not install the SSL directly from the CD before, but you copied the RPM packages to the lpp_source NIM resource, you can set the Install Secure Socket Layer Software (SSLv3) to yes and the lpp_source which contains RPM packages to the lpp_source you use. If your installation of the OpenSSL has failed or is not done you will receive error messages from the NIM configuration utility.

Enable SSL for NIM client

After the NIM server is configured for SSL you have to enable the clients to use SSL for NIM. This is done using the **smit nim_config_services** SMIT screen. See Figure 5-9 on page 205 for the SMIT screen.

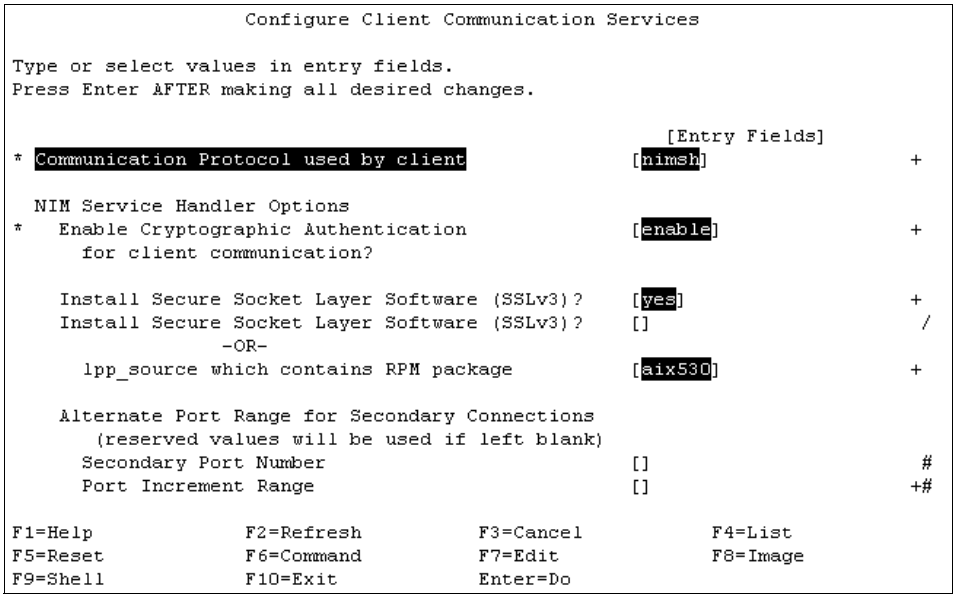

Figure 5-9 smit nim_config_services for SSL

Set the highlighted item Communication Protocol used by client to `nimsh` as the nimsh environment is responsible for the NIM communications. Set Enable Cryptographic Authentication to `enable`. By selecting this item you will switch the client to SSL communications for NIM.

The clients receive the encryption keys through the TFTP protocol. The keys from the /ssl_nimsh directory are copied to the /tftpboot directory during the key exchange protocol. The /tftpboot directory is used for the TFTP protocol as called by the NIM.

Check the function

Check that the client is switched to the nimsh (secure) communication protocol using the **`lsnim -l server2`** command on the NIM master:

```
# lsnim -l server2
server2:
   class          = machines
   type           = standalone
   connect        = nimsh (secure)
   platform       = chrp
   netboot_kernel = mp
   if1            = network1 server2 0
   cable_type1    = bnc
   Cstate         = ready for a NIM operation
```

```
prev_state     = customization is being performed
Mstate         = currently running
cpuid          = 0001810F4C00
Cstate_result  = success
```

You can test the communication of the secure nimsh, for example with the **nim -o lslpp server2** command.

Certificate viewing file

The following are examples from a **certview** certificate viewing script for OpenSSL certificates. The script is located in the /usr/samples/nim/ssl directory.

The script is provided to help users view hash, issuer, subject, and other certificate information available using the **openssl** command. The script can be modified based on user need or preference.

To print out all readable values for certificates:

```
# certview certificate_names
```

To print out the hash value for certificates:

```
# certview -h certificate_names
```

To print out the issuer value for certificates:

```
# certview -i certificate_name
```

To print out the subject value for certificates:

```
# certview -s certificate_name
```

To print out the subject, issuer, and enddate values for certificates:

```
# certview -I certificate_name
```

Certificate password loading file

The following are examples from a **certpasswd** certificate password loading file for NIM OpenSSL certificates. The file is located in the /usr/samples/nim/ssl directory. The file is provided to help users store a desired password for decrypting the NIM master's client key. The password provided must match the password used to encrypt the NIM master's client key during NIM SSL configuration.

To load the encrypted key's password in the NIM environment:

```
#  certpasswd
```

To unload the encrypted key's password from the NIM environment:

```
#  certpasswd -u
```

Note: Only the NIM master's client key may be password encrypted.

To password encrypt the NIM master's client key, complete the following steps:

1. On the NIM master, edit the /ssl_nimsh/configs/client.cnf config file.
2. Locate the encrypt_key variable and change the value to `yes`.
3. Add the `output_password` variable underneath encrypt_key and specify the password. To provide the password within the config file, if output_password is not provided, you will be prompted for the password during key generation.
4. Enter the following command:

   ```
   # make -f /usr/samples/nim/ssl/SSL_Makefile.mk client
   ```

5. On each SSL client, copy the new server.pem file using the **nimclient -c** command.
6. Load the password into the NIM environment using the **certpasswd** command. When using password encrypted keys, NIM commands may fail with the following error if the correct password is not loaded:

   ```
   0042-157 nconn: unable to access the "clientkey.pem" file
   ```

Once loaded, the password specified will be used for client key decrypting until the user unloads the password.

5.14 High Available NIM (HA NIM)

The most significant single point of failure in a NIM environment is the NIM master. AIX 5L Version 5.3 introduces a way to define a backup NIM master, takeover to the backup master, and then failback to the primary master. This helps to create more reliable NIM environments.

Figure 5-10 on page 208 shows a simple functional diagram of the high available NIM architecture.

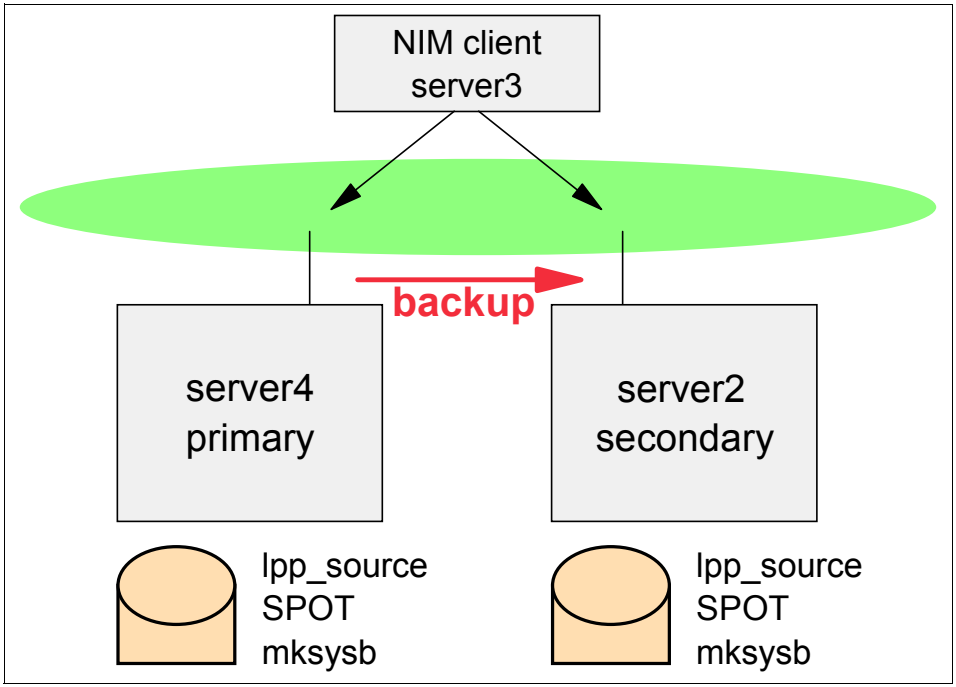

Figure 5-10 High availability NIM

The primary NIM master, server4 is installed in the regular way. It has the regular installation resources created. The secondary or backup server is server2, which needs to have the NIM master filesets installed. The NIM master has to run a backup operation regularly to synchronize the configurations from the primary to the backup server. At the time of writing only rsh communications are supported.

The takeover is done by an administrator command from the backup server. The backup server updates the configuration where possible and is enabled to run operations on the objects. When a fallback is done, the administrator has to re-synchronize the configurations since it may have changed on the backup server.

Figure 5-11 on page 209 shows a new menu added in AIX 5L Version 5.3 to manage the HA NIM environment. You can get this menu by running the **`smit nim_altmstr`** command. The menu is available on the NIM primary master and also on the NIM backup server.

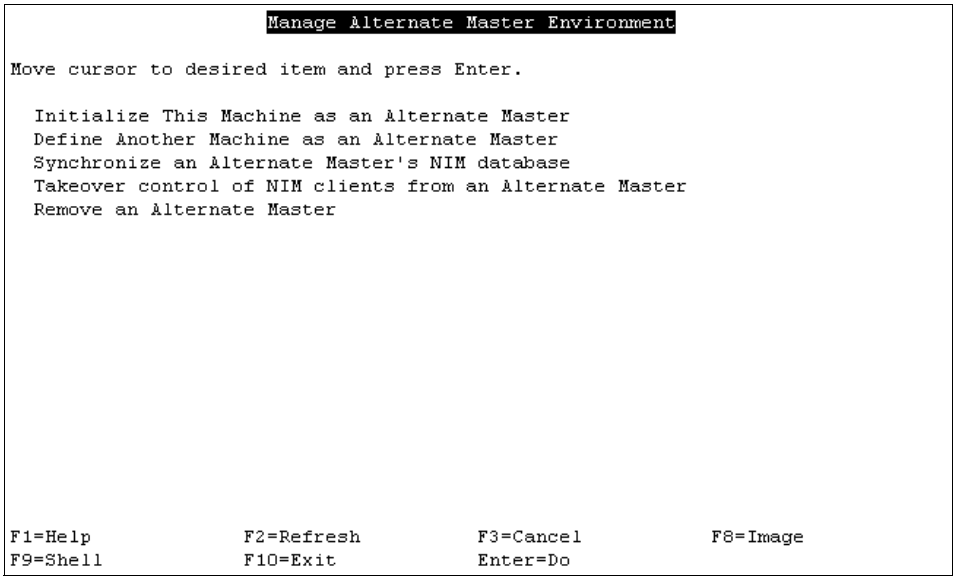

Figure 5-11 smit nim_altmstr

The following sections describe how to configure the HA NIM environment.

Install NIM master filesets

On the backup server install the NIM master and SPOT filesets using regular AIX procedures. Check them using the **lslpp -L** command as shown in the following example.

```
s2 # lslpp -L "bos.sysmgt.nim*"
  Fileset                      Level  State  Type  Description (Uninstaller)
  ----------------------------------------------------------------------------
  bos.sysmgt.nim.client      5.3.0.0    C     F    Network Install Manager -
                                                   Client Tools
  bos.sysmgt.nim.master      5.3.0.0    C     F    Network Install Manager -
                                                   Master Tools
  bos.sysmgt.nim.spot        5.3.0.0    C     F    Network Install Manager -
SPOT
```

Define the backup master to the primary master

Figure 5-12 on page 210 shows the SMIT screen that defines the backup NIM server to the primary NIM server. Run the **smit nim_mkaltmstr** command on the primary NIM server to receive this screen. The highlighted item shows that the definition is related to alternate_master, that is, the backup NIM server.

```
                                Define a Machine

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

                                                      [Entry Fields]
* NIM Machine Name                                   [server2]
* Machine Type                                       [alternate_master]       +
* Hardware Platform Type                             [chrp]                   +
  Kernel to use for Network Boot                     [mp]                     +
  Communication Protocol used by client              []                       +
  Primary Network Install Interface
*   Cable Type                                        bnc                     +
    Network Speed Setting                            []                       +
    Network Duplex Setting                           []                       +
*   NIM Network                                       network1
*   Host Name                                         server2
    Network Adapter Hardware Address                 [0]
    Network Adapter Logical Device Name              []
  IPL ROM Emulation Device                           []                      +/
  CPU Id                                             []
  Machine Group                                      []                       +
  Comments                                           []


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

Figure 5-12 smit nim_mkaltmstr

You can get the same result by running the **nim** command as follows:

```
nim -o define -t alternate_master -aplatform=chrp -aif1='network1 server2 0' \
 -anetboot_kernel=mp server2
```

The SMIT menu will create the necessary definitions into the NIM database of the backup NIM server.

Initialize the backup master

Figure 5-13 on page 211 shows the SMIT screen that allows you to initialize the secondary NIM server. The core command that is run from this menu is the **niminit -a is_alternate=yes ...** command with additional attributes.

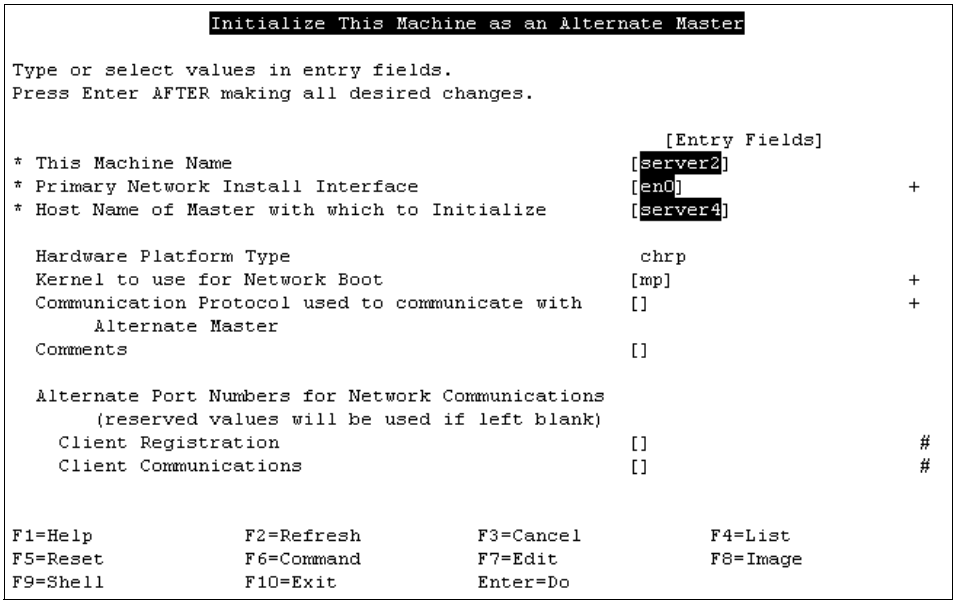

Figure 5-13 smit niminit_altmstr

You can get the same results as displayed in the SMIT screen in Figure 5-13 by running the **niminit** command as follows:

```
# niminit -ais_alternate=yes -aname=server2 -amaster=server4 -apif_name=en0 \
 -aplatform=chrp
```

At this time, the relation between the backup NIM server and the primary NIM master is established. Note that you will need to synchronize the resources and the configuration from the primary NIM master to the secondary NIM server.

Tip: Initialize also the reverse relation, when the primary master is also a backup server. Do it in the same way as you initialized the backup server here.

Perform the synchronization

Before performing the synchronization from the primary master to the backup server, you should create the basic NIM objects, like lpp_source and SPOT, manually on the backup server in the same fashion as on the primary server.

The synchronization procedure makes a backup of the NIM database on the primary master and copies it to the backup server. The backup is then restored on the backup server and recovers the NIM database using the **nim_master_recover** utility.

Figure 5-14 shows the SMIT interface for the NIM synchronization. Start this screen by running the **smit nim_mac_op_sync_hdr** command.

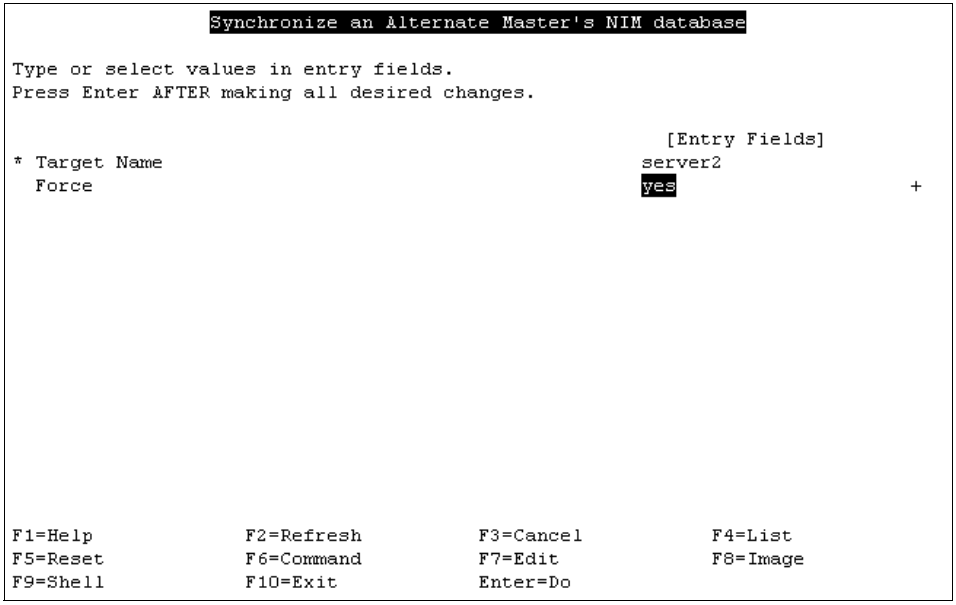

Figure 5-14 smit nim_mac_op_sync_hdr

Alternatively, you can use the **nim** command that is more simple in this case:

```
nim -o sync -aforce=yes server2
```

When the NIM objects are updated, NIM stores the last modification time in the NIM database. This attribute is checked when the backups are performed.

Tip: The synchronization is initiated from the machine where you are logged in to the target machines. If you log in to the backup server and start the synchronization from there you will overwrite your primary server.

Note: You should consider scheduling regular NIM synchronizations.

Takeover

When there is a need to activate the secondary NIM server to function as the primary NIM master, you need to use the **smit nim_mac_op_takeover_hdr** SMIT screen to do it. See Figure 5-15 on page 213 for the SMIT screen.

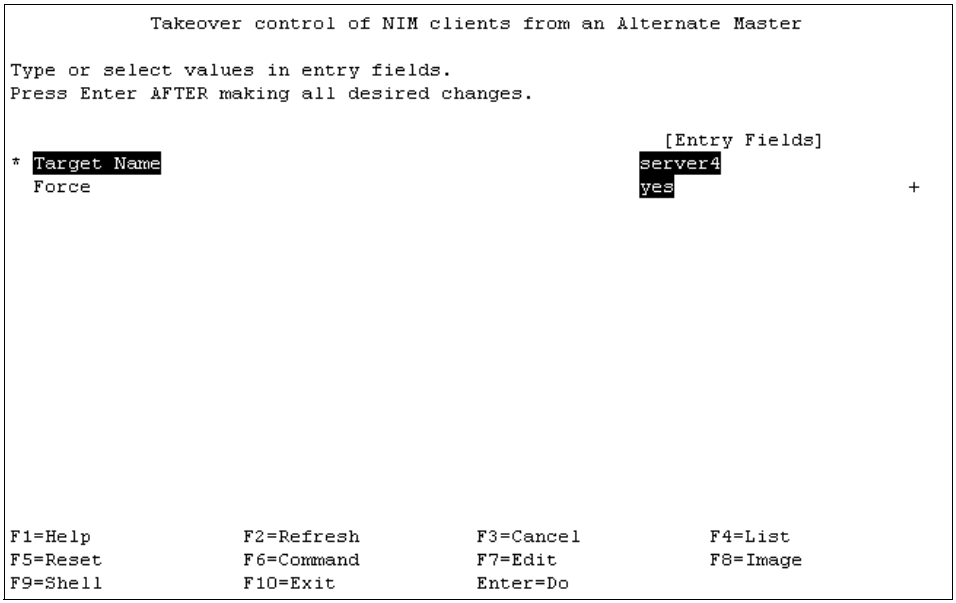

Figure 5-15 smit nim_mac_op_takeover_hdr

Alternatively, you can use the **nim** command that is more simple in this case:

```
# nim -o takeover -aforce=yes server4
```

This will attempt to contact the primary NIM master and update it. If the contact is not successful the backup NIM server updates its database with the sync_required information for the primary master. The backup server attempts to contact the clients and switch the controlling master. If the client is unreachable, the backup NIM server updates its database with the sync_required information for each unreachable client.

The backup server then updates its NIM database and is ready to run operations as the NIM master.

Fallback

When the original primary NIM master is prepared to reverse takeover the operations from the backup can be redirected back to the primary in the same case as in the takeover from the primary to the secondary server.

Now, log in to the primary server and use the **smit nim_mac_op_takeover_hdr** SMIT menu or run the **nim -o takeover -aforce=yes server2** command.

5.15 General NIM enhancements

AIX 5L Version 5.3 provides the following enhancements to the Network Install Manager (NIM):

- Detailed output when creating a NIM lpp_source resource
- Creating SPOT resource from a mksysb
- Restore SPOT copy function
- Adjustments in NIM to process multiple CD media
- NIM interface to change network attributes

We describe these enhancements in the following sections.

5.15.1 Detailed output when creating a NIM lpp_source resource

Before Version 5.3, the NIM lpp_source creation did not give any information about the progress of the procedure. AIX 5L Version 5.3 extends the NIM lpp_source creation through verbose output. If the verbose output is set the procedure that creates the lpp_source calls the **bffcreate -v** or **gencopy -bv** commands with the verbose flag turned on.

There is a new show_progress attribute introduced to the nim command that administrators can modify to turn the verbose attribute on or off. The default value is to have the show_progress attribute turned on. An example of creating the lpp_source with showing the progress output follows:

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

The SMIT menus for creating the lpp_source are updated to enable the user to set the show_progress option.

5.15.2 Creating a SPOT resource from a mksysb

Before Version 5.3 the NIM SPOT resource could be created from an existing lpp_source resource or from the installation media. The disadvantage of such generic SPOT is the amount of disk space it consumes and the long creation time.

In AIX 5L Version 5.3 the SPOT can be created either from an installation source (lpp_source or install media) or from a **mksysb** resource. The **nim** command allows a **mksysb** resource to be entered as the source attribute when creating a SPOT resource.

```
#nim -o define -t spot ... -a source=mksysb_resource spot_name
```

Create a mksysb NIM resource before you create the SPOT. You will need to substitute the reference to a NIM **mksysb** resource if you want to create the SPOT from a **mksysb**. An example that creates the SPOT from a **mksysb** follows:

```
# mksysb -i /export/images/aix530v1.mksysb
...
# nim -o define -t mksysb -aserver=master \
 -alocation=/export/images/aix530v1.mksysb mksysb530v1
...
# nim -o define -t spot -aserver=master -alocation=/export/spot530 \
 -asource=mksysb530v1 spot530
...
```

The SMIT menus for creating the lpp_source are updated to enable the user to set the show_progress option.

The SPOT is created from the **mksysb** using the /usr/lib/bootpkg/bootpkg_list file that lists only the necessary files for the SPOT. This results in creating a quite small SPOT, that is approximately 50 MB in size. When the SPOT is created from the lpp_source, the only way to create a universal SPOT is to run the **installp** command for all devices, which produces a large SPOT whch is about 400 MB.

5.15.3 Restore SPOT copy function

In AIX 5L Version 5.3, the restore SPOT copy function was reintroduced. The user can select the SPOT residing on the NIM server to install BOS on the clients. The **nim** and the **nimclient** commands are modified to accept the SPOT as source. An example for the **nim** command follows:

```
# nim -o bos_inst -a source=spot -a spot=spot530 -a lpp_source=aix530 \
 -a no_client_boot=yes server3
```

The command will allocate the SPOT resource named spot530, along with other resources, and the client can be rebooted to perform the BOS installation from the SPOT.

Several SMIT menus related to BOS installation are updated and can accept the SPOT as the source for the BOS installation.

5.15.4 Adjustments in NIM to process multiple CD media

The BOS installation of previous versions of AIX was limited to one CD media. Future releases of AIX BOS installation may require more that a single CD media

for the BOS installation. An enhancement to the NIM procedures to handle the installation CD media was done so that NIM handles multiple installation media.

When NIM copies the installation files from the CD media to disk, it uses the **gencopy** command. The **gencopy** command uses **bffcreate** to package the files from the media. The **gencopy** command accepts a subset of flags used for bffcreate specified after the -b flag, and then forwards these options to the **bffcreate** command.

The default behavior of the **bffcreate** command is to handle multiple media. The -S flag used in **bffcreate** suppresses multi-volume handling of the input source.

When creating the lpp_source with the **nim** command or through SMIT, you can set the creation to process or not to process multiple volumes by using the -a multiple_volumes=yes attribute to the **nim** command. An example of the nim command follows:

```
nim -o define -t lpp_source -aserver=master -alocation=/export/aix530 \
 -asource=/dev/cd0 -amulti_volume=yes
```

5.15.5 NIM interface to change network attributes

In AIX 5L Version 5.2 there was already a **nim_master_recover** tool that could recover the NIM master configuration. AIX 5L Version 5.3 enhances the **nim_master_recover** tool so that it can recover the NIM master after the network attributes change. This helps the administrators to more easily change the IP address of the NIM master. In addition there is a new SMIT menu that addresses the NIM network changes. See Figure 5-16 for the **smit nim_ch_if1** screen.

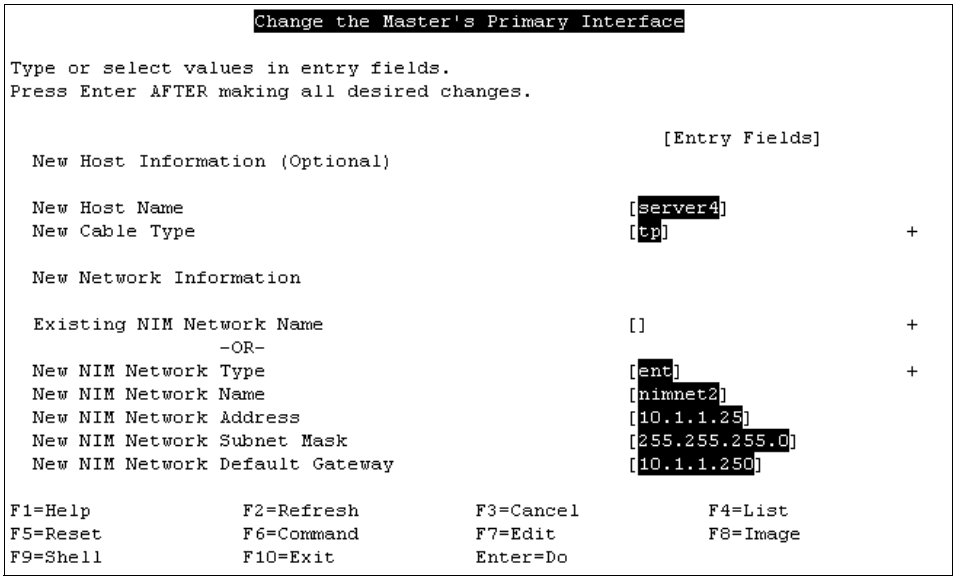

Figure 5-16 smit nim_ch_if1

The correct procedure to change the network attributes in a NIM environment is as follows:

1. Add the new hostname and IP address to the /etc/hosts, DNS, or other name resolving service.
2. Update the /.rhosts on the clients if using it.
3. Run the **smit nim_ch_if1** SMIT menu that changes the NIM configuration.
4. Change the TCP/IP configuration in AIX.
5. Check the environment.
6. Delete old NIM definitions.

Changing the NIM master network configuration through the SMIT menu in Figure 5-16 on page 217 will do the following:

- The current NIM database is first saved by the m_backup_db method.
- The NIM network resources are updated in the NIM database.
- The NIM master's primary interface (if1) is updated to include the new network, hostname, and adapter information.
- The NIM client /etc/niminfo files are updated on the clients.

If the procedure fails on a client for any unexpected reason, such as incorrectly updated DNS or /.rhosts, you will need to re-initialize the client configuration using the **niminit** command.

5.15.6 VIPA and EtherChannel support in NIM adapters

In AIX 5L Version 5.3 the **nimadapter** command is extended to accept definitions of VIPA and EtherChannel secondary adapters.

5.15.7 EZ NIM enhancements

EZ NIM is an easy to understand user interface to the NIM environment. EZ NIM does the following:

- Creates an lpp_source resource when setting up the master environment.
- Adds nimsh and cryptographic authentication when setting up the master and reinstalling the clients.
- Adds an rte option to be selected for client reinstall.
- Adds install options of overwrite, preservation, and migration for client reinstall.
- Adds new options for viewing the NIM environment.

We do not show examples of SMIT screens here because after you run the **smit eznim** fast path, you get into the easy to understand, self-explanatory menus of the EZ NIM.

5.16 Alternate Disk Installation migration

Alternate Disk Installation Migration (ADM) is an Alternate Disk Installation option available whenever the version or release values for an AIX release change. These are the V and R values in the VRMF (VersionReleaseMaintenanceFix), which is 5.3.0.0 for this release of AIX 5L. ADM allows the users to migrate their systems *on the side* without requiring downtime for installation. ADM is supported at 4.3 and higher. ADM is currently only available for migration to 5.1 or 5.2. This function is now added to AIX 5L Version 5.3.

Alternate Disk Installation migration can be done using SMIT with or without NIM.

5.16.1 Alternate Disk Migration without NIM

This method is a two-step process. First you clone the rootvg and then update the operating system using the new release CD.

As shown in Figure 5-17, you select the Alternate Disk Installation option from the Software Installation and Maintenance menu in SMIT.

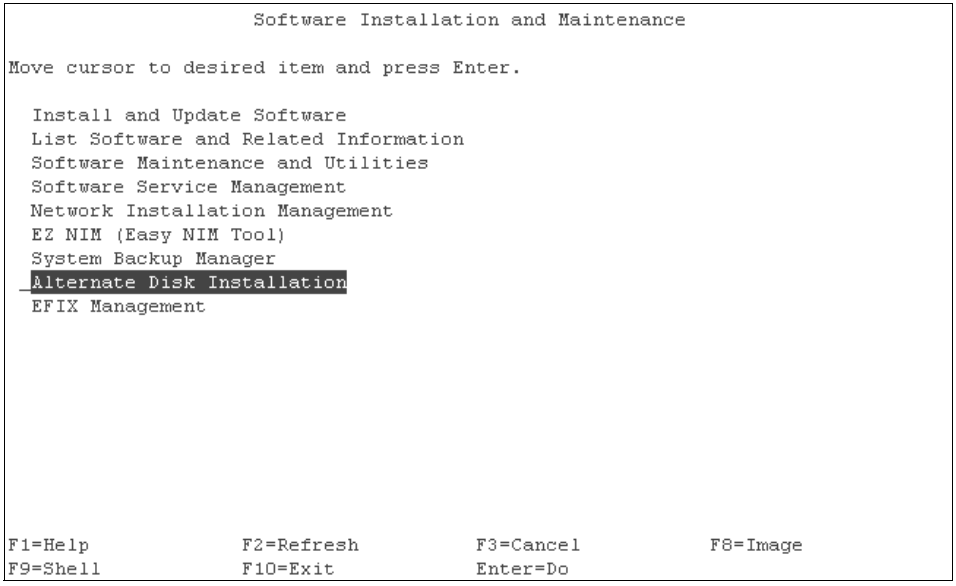

Figure 5-17 SMIT Software Installation and Maintenance panel

The next step is to select the Clone the rootvg to an Alternate Disk option as shown in Figure 5-18.

On the Clone the rootvg to an Alternate Disk panel, use update_all in the Bundle to install field and press Enter as shown on Figure 5-19.

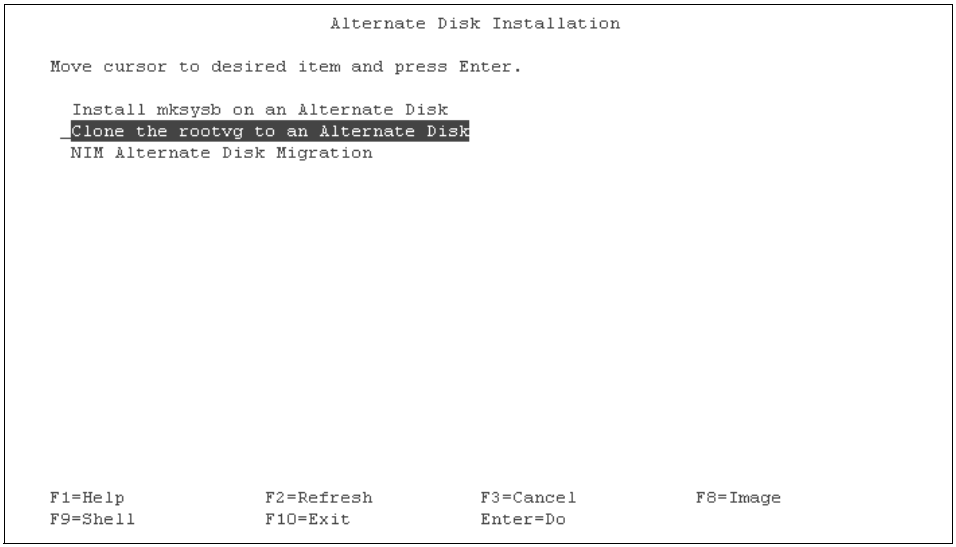

Figure 5-18 SMIT Alternate Disk Installation panel

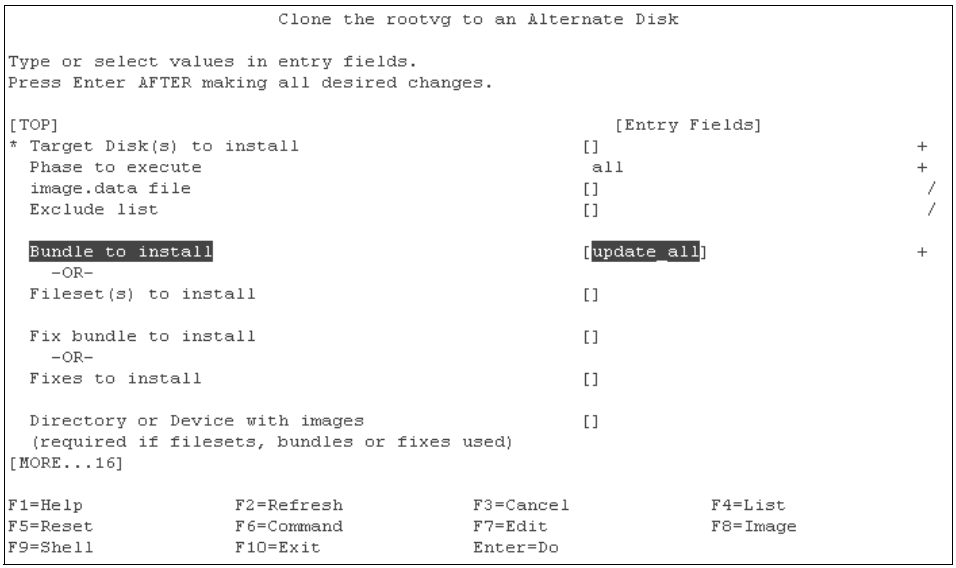

Figure 5-19 Clone the rootvg to an Alternate Disk panel

5.16.2 Alternate Disk Migration with NIM

This process is similar to the previous one except you select NIM Alternate Disk Migration in the panel shown in Figure 5-18 on page 220, and you receive the NIM Alternate Disk Migration panel shown in Figure 5-20.

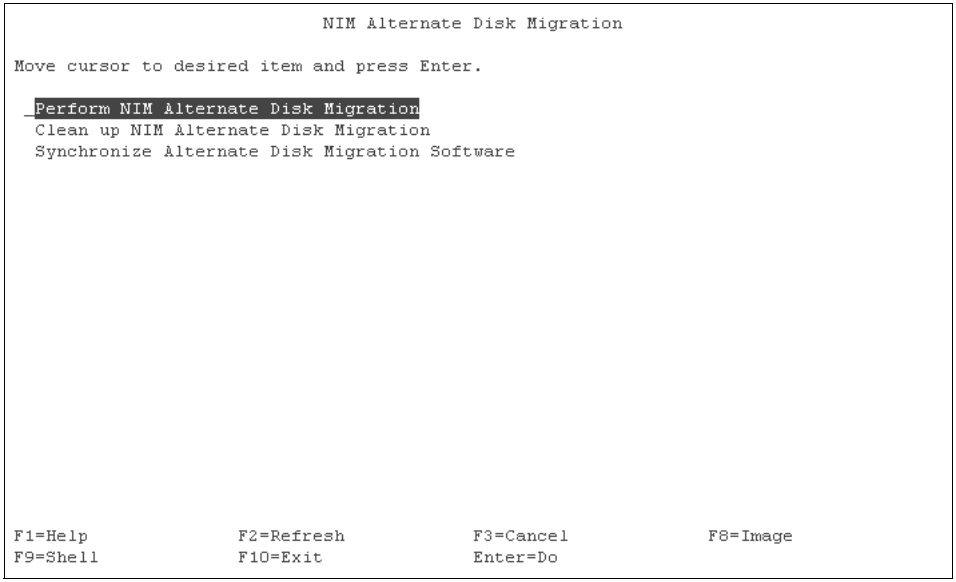

Figure 5-20 NIM Alternate Disk Migration panel

Selecting Perform NIM Alternate Disk Migration brings you to the Perform NIM Alternate Disk Migration panel (Figure 5-21 on page 222) where you fill in appropriate values for the fields.

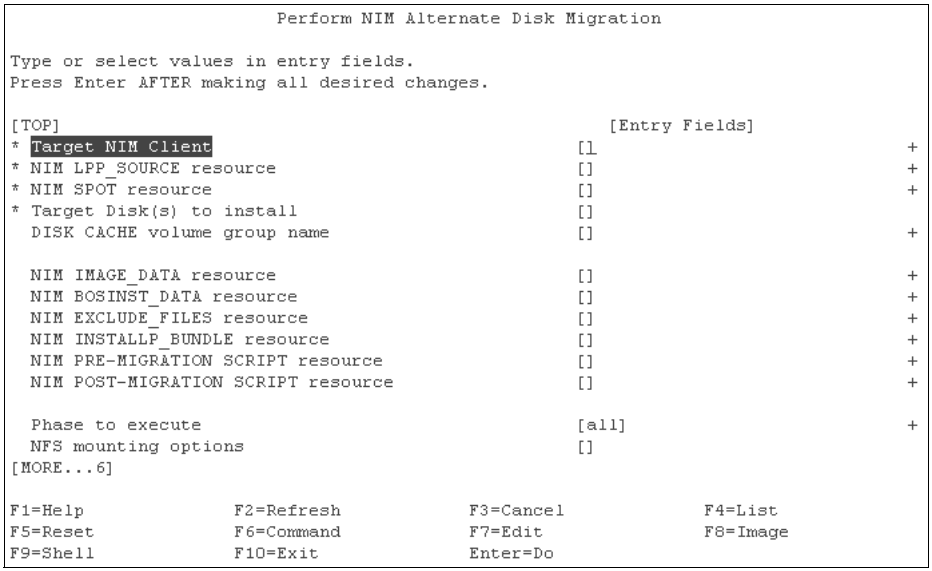

Figure 5-21 Perform NIM Alternate Disk Migration panel

5.17 Enhancements to Alternate Disk Installation

AIX 5L Version 5.3 has implemented a number of changes to make the alt_disk_install operations easier to use, document, and maintain.

The following functional changes have been implemented:

- alt_disk_install has been partitioned into separate modules with separate syntax based on operation and function.
- A library of common functions that can be accessed by the modules has been implemented.
- Error checking and robustness of existing alt_disk_install operations has been improved.
- Documentation has been improved by creating a separate man page for each module (currently there is one extremely large man page).

The following three new commands have been added:

- alt_disk_copy creates copies of rootvg on an alternate set of disks.
- alt_disk_mksysb installs an existing mksysb on an alternate set of disks.
- alt_rootvg_op performs Wake, Sleep, and Customize operations.

Also, a new library, alt_lib, has been added that serves as a common library shared by all alt_disk_install commands.

The alt_disk_install module will continue to ship as a wrapper to the new modules. However, it will not support any new functions, flags, or features.

Table 5-3 shows how the existing operation flags for alt_disk_install will map to the new modules. The alt_disk_install command will now call the new modules after printing an attention notice that it is obsolete. All other flags will apply as currently defined.

Table 5-3 alt_disk_install command arguments

alt_disk_install command arguments	New commands
-C <args> <disks>	alt_disk_copy <args> -d <disks>
-d <mksysb> <args> <disks>	alt_disk_mksysb -m <mksysb> <args> -d <disks>
-W <args> <disk>	alt_rootvg_op -W <args> -d <disk>
-S <args>	alt_rootvg_op -S <args>
-P2 <args> <disks>	alt_rootvg_op -C <args> -d <disks>
-X <args>	alt_rootvg_op -X <args>
-v <args> <disk>	alt_rootvg_op -v <args> -d <disk>
-q <args> <disk>	alt_rootvg_op -q <args> -d <disk>

5.18 Advanced Accounting

The Advanced Accounting subsystem is based on mainframe technology and features interval accounting, data aggregation, and dynamic classification of accounting data. You can customize Advanced Accounting for different computing environments. You can configure Advanced Accounting to produce the specific types of records needed for billing applications. By default, all accounting records are produced.

Advanced Accounting provides usage-based information for a wide variety of system resources so that you can develop comprehensive charge-back strategies. You can collect accounting data on resources such as disks, network interfaces, virtual devices, file systems, processors, and memory. Interval accounting gives you the ability to view this data over system administrator-defined time intervals in order to develop chronological views. This has several potential applications, including capacity planning.

Interval accounting can be used to periodically record accounting data, enabling the production of more accurate bills. You can configure Advanced Accounting to produce intermediate process records for active processes. These records can be added to the completed process records to produce a bill that reflects the total use of system resources.

Data aggregation is a way to control the amount of data written to a data file. This helps system performance by reducing the overall resource load needed to run Advanced Accounting. Aggregation minimizes a system's I/O requirements, adding accounting records so that fewer records are written to the accounting file. It is transparent to applications and middleware.

Policies are rules that provide for the automatic classification of processes. Classification is done according to users, groups, and applications, categorizing the use of system resources by billable entities. These categories are called projects.

APIs are provided so that applications and middleware can describe the transactional nature of their workloads, enabling charge-backs for server processes. These APIs are intended to define and delineate transactions and to identify the end user, if possible. The Advanced Accounting subsystem measures the resource use of transactions, if possible, and records all of this information in the accounting file.

Advanced Accounting also provides new statistics from previous accounting tools. For example, the process record provides microsecond-level CPU times, a memory integral based on elapsed time (standard UNIX accounting bases it on CPU times), local and distributed logical file I/O, and local and remote socket I/O.

5.18.1 Data files

Every statistic recorded by Advanced Accounting is written to a data file. When the data file reaches its capacity, the billing application can process the file. After it is processed, that file can be reused, and the cycle repeats.

You must create your accounting data files in order to begin collecting accounting statistics. This involves assessing your company's needs and determining how much data you will be collecting. You can start Advanced Accounting and let it run for a period of time to see how much data is produced, giving you an idea of how much disk space to reserve for accounting, as well as how many files you will need to manage your data.

Although not required, it is a good idea to specify at least two data files so that Advanced Accounting can remain active at all times. Advanced Accounting writes to only one file at a time, but as it does, it requires exclusive access to this

file. With two files, Advanced Accounting can write to one file while the other file is processed by the billing application.

The following explains the life-cycle of a data file as it moves through the accounting process:

1. A pool of pre-allocated data files is created and registered with the Advanced Accounting subsystem.
2. As accounting statistics are collected from the system, data is written to the active data file.
3. The size of the data file (predetermined when the file is created) limits the amount of data that can be written to the file. Once the data file is full, no more data may be written to it. The Advanced Accounting subsystem will then switch to the next empty data file and begin writing data to it.
4. Once the data file is full, you can use post-processing tools, such as a billing application, to process the accounting data into statistics for billing purposes.
5. After the data is extracted from the data file and processed, the file is reset so that it can be reused.

The Advanced Accounting subsystem can be configured to send the system administrator notification when the data file is 90 percent full and when the data files get switched. These notifications can be delivered either through the syslog daemon or e-mail. These two notifications tell the system administrator that the data needs to be extracted from the data file and processed.

When a data file is created, it is registered with the kernel. This is how the kernel knows the identity of the file and can automatically switch. Table 5-4 provides a list of the tasks and commands used with advanced accounting.

Table 5-4 Data file management commands

Task	Command	SMIT fast path
Allocate and define an accounting file with specified file name and size. The default size is in megabytes.	**acctctl fadd** file size	**smit create_aacct_file**
Remove the specified accounting file from the accounting subsystem. This will not remove the file from the file system.	**acctctl frm** file	**smit release_aacct_file**
Indicate that the specified file can be reused by the accounting subsystem.	**acctctl freset** [file]	**smit reuse_aacct_file**

Task	Command	SMIT fast path
Query the state and current use of the specified file. If no file is specified, all files are queried.	**acctctl fquery** [file]	**smit list_aacct_file**
Force Advanced Accounting to switch to a new accounting data file. The new file may be optionally specified.	**acctctl fswitch** [file]	**smit switch_acct_file**

5.18.2 Projects and policies

Projects represent billable entities such as users, departments, divisions, companies, or tasks. Each project is composed of a project number, project attribute, and project name, which collectively represent a project definition. Project definitions are entered into the project definition database.

Projects are written to accounting records. Report and analysis commands convert project numbers into project names by looking up entries in the system project definition database. Logically, projects are indices into critical business data (for example, customer name, billing address, account number, service level agreement) that are maintained by the billing application.

Project numbers are assigned through project assignment policies, which are composed of project assignment rules. Each rule contains classification criteria that when fully satisfied, yield a classification result. The classification result is a project list that is logically assigned to the object, usually a process, being classified. The classification criteria depends on the type of policy.

Project lists enable manual project assignment. If a list of projects is specified, users can change their current project assignment to another project in the list. This provides the capability to launch jobs under different projects, which is useful when you are performing work on behalf of multiple clients. The system administrator can assign multiple projects to any user, then manually change the project assignment.

Policies

Policies automate project assignment. A policy is comprised of classification criteria and classification results. The project assignments take place during subroutines and kernel services, such as exec(), initp(), setuid(), and setgid().

Data can be classified by user, group, application, or a combination of these attributes. Depending on the type of policy that is created, administrators can specify a user name, group name, application name, and project list in the policy

file, although all four components do not have to be present for a policy to function.

Processes can be assigned in the following two ways:

- Using assignment rules when process classification attributes change. This is the most common way that processes are classified.
- Manual assignment to a class by a user with the required authority.

Advanced Accounting supports the following types of assignment policies for processes, each with different classification criteria:

- Admin policy
- User and group policies

The Admin policy supports most filters and the use of Korn shell syntax wild cards. Configuring and maintaining Admin policies requires a standalone administration tool, such as Web-based System Manager, SMIT, or an editor.

The User and Group policies provide project assignment based exclusively on user names or group names, depending on the policy. They are integrated with user administration. Project lists can be specified for users and groups when these users and groups are created.

Admin policy

The Admin policy uses the user name, group name, and application name process attributes to classify processes. The Admin policy is application-based, and provides the ability to gather accounting statistics at the application level.

By default, the Admin policy is located in the /etc/project directory. You may create alternate Admin policies to use at various times. For example, you may want to have an Admin policy that runs Monday through Friday, and another that runs on Saturday and Sunday. The alternate Admin policies are stored in subdirectories of the/etc/project/alter root directory. You must specify a name for the subdirectory when you are creating a new Admin policy. The Admin policy consists of one or more assignment rules that are placed in the Admin policy file, and conform to the following syntax:

```
user name:group name:application name:Project List::Optional comments go here
```

For details on the values that are allowed in each field of the assignment rule, see Table 5-5.

Table 5-5 User, group, and application rules

Type of rule	Description
user	May contain either a hyphen (-) or at least one valid user name as defined in the /etc/passwd file. An exclamation point (!) can be used before a name to exclude a given user from the class. The list of user names is composed of one or more names, separated by a comma (,). Patterns can be specified to match a set of user names, using full Korn shell pattern-matching syntax. If an invalid user name is used, Advanced Accounting will ignore the entire rule. If a hyphen (-) is used, Advanced Accounting will skip to the next field in the rule.
group	May contain either a hyphen (-) or at least one valid group name as defined in the /etc/passwd file. An exclamation point (!) can be used before a name to exclude a given user from the class. The list of group names is composed of one or more names, separated by a comma (,). Patterns can be specified to match a set of user names, using full Korn shell pattern-matching syntax. If an invalid user name is used, Advanced Accounting will ignore the entire rule. If a hyphen (-) is used, Advanced Accounting will skip to the next field in the rule.
application	May contain either a hyphen (-), a list of application path names, or the command name of a kernel process. This is the path name of the applications (programs) run by processes included in the class. The application names will be either full path names or Korn shell patterns that match path names. The list of application names is composed of one or more path names, separated by a comma (,). An exclamation mark (!) can be used before a name to exclude a given application. At least one application in the list must be found at load time or the rule is ignored. Rules that are initially ignored for this reason might become effective later on if a file system is mounted that contains one or more applications in the list.

For process assignment policies, the classification criteria is the user name as listed in the /etc/passwd file, the group name as listed in the /etc/groups file, and the fully qualified application name. The classification result is a project list. By default, the first project in the list is used, unless the user changes his current project assignment to a secondary project in the list.

Classification is done whenever an attribute value changes by comparing the value of the process attribute against a list of possible values given in the class assignment rules file, called rules. The comparison determines which rule is a match for the current value of the process attributes.

To classify the process, Advanced Accounting examines the top-level admin policy for the active configuration to determine in which class the process belongs. For each rule in the file, Advanced Accounting checks the current values of the process attributes against the values and lists of values specified in

the rule. Advanced Accounting goes through the rules in the order in which they appear in the admin file, and classifies the process in the project corresponding to the first rule for which the process is a match. Therefore, the order of the rules in the rules file is significant.

The following is a list of the criteria used to determine whether the value of a process attribute matches the values of the same attribute field in the admin policy file:

- If the field in the rules file has a value of hyphen (-), then any value of the corresponding process attribute is a match.
- If the value in one of the fields is preceded by an exclamation point (!), that value is always excluded.
- If the value in one of the fields is followed by an asterisk (*), then every match with that value will be recognized.

Examples of admin policy rules

The Admin policy allows you to specify more than one user, group, or application in their respective fields. For instance, if you wanted user1, user2, and user3 to be given the same attributes, you would specify the following syntax:

```
user1,user2,user3:-:-:Project List::Comments
```

This syntax shows that user1, user2, and user3 will be treated the same way for this rule. The dashes in the group name and application fields are wildcards. You can also use an asterisk (*). As previously mentioned, you can also use wildcards to include all the values of a certain attribute. For example, if you wanted to include every user name beginning with B, you would type B* in the User Name field. Full Korn shell pattern matching syntax is allowed in all of the fields of the rule.

You can also set up your Admin policy to include certain users, but exclude others:

```
user1,!user2,user3:-:-:Project List::Comments
```

This syntax shows that user1 and user3 will have the same attributes, but the policy will exclude user2.

Note:

1. The kernel only reads numeric values. In these example, the user names user1, user2, and user3 are converted to numeric values after they are loaded into the kernel.
2. If any changes are made to the policy, it must be reloaded into the kernel using the **`projctl`** command.

Aliases

You can define an alias name for a list of users or groups so that you do not have to enter the full list of users, groups, or application names in a rule. Aliases provides a shorthand for referring to lists of users and groups, which simplifies admin policies, and makes them more readable and easier to maintain. A separate alias file is used with each Admin policy and is placed in the same directory as its associated Admin policy.

To create an alias where user1, user2, and user 3 are grouped into an alias named dev1, define it as follows:

```
dev1:user1,user2,user3::Development team 1
```

To use the alias name in the Admin policy, the alias name must be preceded by a dollar sign ($). For example, using the dev1 alias instead of user1, user2, and user3 in the Admin policy looks like similar to the following:

```
$dev1:-:-:Project List::This is a rule within a user alias
```

Multiple aliases, separated by a comma, may also be used. Aliases are excluded by placing an exclamation point (!) before the alias name in the Admin policy rule.

Alternate admin policies

It is possible to create multiple Admin policies to reflect different billing strategies based on the time of day or the day of the week. The **projctl** command is used to load alternate Admin policies. Previously loaded Admin policies do not have to be unloaded to load a new policy. The cron facility is used to load a policy at the given time.

Alternate Admin policies are placed in the /etc/project/alter directory. For example, an alternate Admin policy with the name “weekend” is placed in the Admin file /etc/project/alter/weekend/admin. If this policy used aliases, they would be placed in the /etc/project/alter/weekend/alias file. SMIT and Web-based System Manager are used to create, change, show, load, unload, and remove Admin policies. Alternate policies are addressed in SMIT by changing the current focus to the name of the policy.

Relative project classification

With Advanced Accounting, you can enable project assignments to be made relative to User IDs and Group IDs. To accomplish this, use the $UID and $GID keywords in a project list to tell the system that the project code must be computed. A simple expression in the form “keyword” or “keyword + constant” may be used, where “constant” is either a decimal or hexadecimal number (0xffff).

The following rule (Table 5-6) illustrates the use of relative project classification.

Table 5-6 Relative project classification

User	Group	Application	Project
*	-	-	$UID+1000000

Disable accounting for selected processes

System administrators can disable accounting for selected processes through the classification process. This can be accomplished by specifying a *NoAccounting* project list. This attribute is inherited from the parent process to the child process. Table 5-7 illustrates how this may be accomplished.

Table 5-7 Example of the No Accounting specification in a project list

User	Group	Application	Project
db2inst1	db2inst1	~db2inst1/sqllib/adm/db2start	NoAccounting
root	root	kbiod	NoAccounting

DB2® is classified in the first rule by the subroutine exec(). The NFS kproc in the second rule is classified by the subroutine initp().

User and group policies

The user and group policies use the process attributes for users and groups. A user or group name and a project list constitute a project assignment rule within the user or group policies. There is no file associated with a user or group policy because user policy data is stored in a security database.

Table 5-8 shows the structure of a user policy.

Table 5-8 Structure of a user policy

User name	Project list
user1	project1, project2
user2	biology, chemistry

To create an example group policy, complete the following:

1. Create an example group dev by typing the following:

   ```
   # mkgroup dev
   ```

2. Create an example project biology_dev with the project ID 1000 by entering the following:

   ```
   # projctl add biology_dev 1000
   ```

3. Associate the project biology_dev with the newly created group dev:

```
# chgroup projects=biology_dev dev
```

To create a user policy, complete steps one through three, substituting **mkuser** and **chuser** for **mkgroup** and **chgroup**. Once a user or group has been created, a project is associated with it. The users or groups can have several projects associated with them, but only one can be active at a time.

Classification semantics

For every exec(), initp(), setuid(), and setgid() subroutine, the process will be reclassified using the project assignment rules to determine if the current project assignment should be changed. If a project assignment rule is not loaded, or if a rule cannot be successfully applied, then the current project identifier is used.

The default project system identifier is zero (0). It is applied to the base system processes before accounting is enabled and it may be used to signify general system overhead. After it is assigned, a project is inherited from the parent process to the child process using the fork() kernel service and creatp() kernel service.

The use of the application filter varies between the initp() kernel service and the exec() subroutine. In the former, the command name of the kernel process that is being started is used to perform the classification. The command name is visible through the **ps** command. In the latter, classification is performed using the file identifier (FID) of the executable, with the FID of the fully qualified path name that is specified by the policy. Only FIDs that can be computed at policy load time are accommodated.

If a process is classified through a rule that explicitly names an application, then that project identifier should be applied to the process and its children, because the intent is to label a block of work. This implies that subsequent exec(), setgid(), and setuid() subroutines in the process as well as its children do not perform project reclassification. Internally, these processes are identified with a sticky bit that indicates they were classified through a rule that explicitly named an application. The sticky bit is visible through the use of the new -P flag of the **ps** command. The **ps -P** command displays the project name and subproject identifier for the project. If the sticky bit is set for the process, the project name will be preceded by an asterisk (*) character.

When you load a new policy file, all of the processes in the system are reclassified, except those with a sticky bit. Processes with a sticky bit cannot be changed. When a different project identifier is assigned to a process, a process accounting record is written to the accounting data file, so that the use of resources by a project may be accurately reported. Whenever this occurs, the accounting statistics for the process are reset to zero.

Manual project classification

Users can manually change their current project assignment. For non-privileged users, authorization is provided through the policy file. The first project listed in the rule is considered to be the default project and is automatically chosen, unless the user changes his current project assignment to another project in the list. Changing the current project assignment applies only to the current session and it is not silently applied to other jobs with the possible exception of the parent process (for example, the shell).

Project lists may also be specified in rules that name applications, allowing administrators to associate projects with specific instances of applications. This is important for server processes, which may be externally configured to manage specific resources that have different cost structures.

This mechanism enables system administrators to charge indirectly for those resources that may not directly support usage-based accounting. For example, a system administrator that wants to charge different rates to access specific databases could start different instances of Oracle to manage those databases.

Commands

Table 5-9 shows the commands used to administer projects and policies.

Table 5-9 Projects and policies commands

Task	Command	SMIT fast path
Add a project definition	**projctl add** *projname*	**smit add_proj**
Load or reload a project definition	**projctl ldprojs** [**-r -a**]	**smit load_proj**
Remove the specified project definition	**projctl rm** *projname* [**-d** *admnpath*]	**smit rename_proj**
Show or change active project definition	**projctl chg** *projname* [**-p** *pid*]	**smit show_chg_proj**
Merge two project definitions	**projctl merge** *projpath*	None
Start a program with a project assignment	**projctl exec** *projname cmd line*	**smit start_proj_prg**
Enable or disable aggregation for a project	**projctl chattr agg** *projname*	None
Show the policies that are loaded on the system	**projctl qpolicy**	**smit query_policy**
List all active project definitions	**projctl qprojs** [**-n**]	None

Task	Command	SMIT fast path
List the specified project definition	**projctl qproj** *projname*	**smit show_chg_proj**
Change the project assignment for a process	**projctl chg**	**smit chg_proj_proc**
Show project assignment for a program	**projctl qapp** *appname*	**smit show_proj_pgm**
Load policies	**projctl** {**ld** |{{{{**ldall** | **ldadm**}[**-d** admnpath]} | **ldgrp** | **ldprojs** | **ldusr**}[**-a**]}	**smit load_admin** **smit load_users** **smit load_groups**
Unload policies	**projctl** {**unld** |{{{{**unldall** | **unldadm**}[**-d** *admnpath*]} | **unldgrp** | **unldprojs** | **unldusr**}[**-a**]}	**smit unload_admin** **smit unload_users** **smit unload_groups**
Create an admin policy	None	**smit create_admin**
Show or change current focus policy	None	**smit change_show_focus**
Remove an admin policy	None	**smit remove_admin**
Add a rule	None	**smit add_admin_rule**
Remove a rule	None	**smit remove_admin_rule**
Add a user alias	None	**smit add_usr_alias**
Add a group alias	None	**smit add_grp_alias**
Show or change the specified alias	None	**smit chg_alias**
Remove the specified alias	None	**smit remove_alias**
Manage projects	None	**smit work_project**
Remove project definition	None	**smit remove_admin_proj**
Create a project list for a user	**chuser** *user*	**smit create_user**
Create a project list for a group	**chgroup** *group*	**smit create_group**
Show or change a specified project list for a specified user	**chuser projects=***projectlist user*	**smit change_show_user_list**
Show or change a project list for a group	**chgroup** *group*	**smit change_show_group_list**
Remove a project list for a user	**chuser projects=***user*	**smit remove_user**

Task	Command	SMIT fast path
Remove a project list for a specified group	**chgroup projects**=*group*	**smit remove_group**
Show project lists for all users	**lsuser -a ALL**	None
Show project lists for all groups	**lsgroup -a projects ALL**	None

5.18.3 Transactional accounting

Applications use the Application Resource Management (ARM) interfaces to describe the transactional nature of their workloads. Advanced Accounting supports ARM interfaces by recording information that is presented through these interfaces in the accounting data file. To use this information for charge back purposes, it is necessary to understand the programming model associated with the ARM interfaces and the mechanism for passing critical business information to the Advanced Accounting subsystem, so that this information can be preserved for your billing application.

The following provides a list of the known ARM interfaces:

- arm_register_application
- arm_start_application
- arm_register_transaction
- arm_start_transaction
- arm_block_transaction
- arm_unblock_transaction
- arm_bind_transaction
- arm_unbind_transaction
- arm_stop_transaction
- arm_stop_application
- arm_destroy_application

The ARM APIs provide a way to delineate application transactions. This enables the operating system to identify them and Advanced Accounting to measure and record them in the accounting data file. The ARM APIs also enable applications to describe their transactions, so that site-specific information can be provided. This is accomplished largely by convention through the use of a named set of parameters that are recognized by the ARM implementation and Advanced Accounting.

Advanced Accounting also captures information that is generated internally. This information cannot be changed by the user, but is needed by accounting.

The ARM programming model has a fundamentally hierarchical structure. Transaction instances are derived from transaction definitions that are defined when registering transactions. Application instances are derived from application definitions that are defined when registering applications. When starting a transaction, a transaction definition that should be used within an application instance is specified, enabling the full set of information to be defined for each transaction instance. For example, all transactions have application and group names.

Advanced Accounting supports the ARM interfaces through a hierarchy of records:

- Application environment record
- Transaction environment record
- Transaction instance record

The application environment record describes a unique combination of application information including the application name, application group name, and properties. Each application environment record is assigned a unique identifier so that it can be referred to symbolically. This identifier is called the application environment identifier, and it is included within the transaction instance record.

The transaction environment record describes a unique combination of transaction information, including the transaction name and properties. Each transaction environment record is assigned a unique identifier so it can be referred to symbolically. This identifier is called the transaction environment identifier, and it is included with the transaction instance record.

Application environment identifiers and transaction environment identifiers are long lived, but they are not persistent in nature. Each time the system boots, the identifiers are regenerated. Advanced Accounting circumvents this problem by recording the application and transaction environment in each file so report and analysis commands should use the entry in the current file to determine the unique combination of values that should be applied to a transaction.

The transaction instance record is used to describe each individual transaction. It includes the application environment identifier and the transaction environment identifier defined previously. These identifiers should be used to look up the corresponding application environment record and the transaction environment records so that the application names, application group names, and transaction names can be associated with the transaction instance. The intent is to minimize

the amount of data that needs to be recorded with each transaction instance since a lot of it is static in nature.

Advanced Accounting also supports the aggregation of ARM accounting data.

5.18.4 Interval accounting

Interval accounting provides a way to collect accounting data at specified intervals. You can configure it to produce intermediate process records for active processes. These records can be added to the completed process records to produce a more accurate bill reflecting the total use of the system by a user in a given time period. Interval accounting can also be configured to periodically capture accounting data for system resources like processors, memory, disks, network interfaces, and file systems. This information can be used to generate a bill that reflects the total use of a partition.

Usage information about system resources has other applications beyond charge back. It can be used to perform capacity planning because it shows the use of physical resources like processors, memory, disks, and network interfaces. Interval accounting provides a time-based view of these resources, so that load can be determined, enabling capacity-based decisions to be made based on the data. For example, it is possible to determine the idle processor capacity in a given interval, which can be used to determine whether more processors are needed.

Interval accounting provides a historical view of resource use, which can be used for performance analysis. For example, the file system records can be examined to determine which file systems were busy when access to the system was slow. The data identifies multiple busy file systems served by the same disk adapter. This information can be used to balance file systems over disk adapters. You do not always know which file sets are being accessed at a given time, so having the ability to replay a log with this information is an asset.

System interval

System interval accounting collects resource use information about system resources such as processors, disks, network adapters, file systems, and memory usage. This information is used to profile the use of your system. For most purposes, an interval of once per hour should be sufficient. The system interval does not collect process accounting data.

Process interval

The process interval is used to capture information about long-running jobs such as data modeling applications that run for months. The standard process completion record is sufficient for most processes. Therefore, the process

interval should be relatively large, so that it does not produce unnecessary data. The process interval writes a record for every active process, including newly started jobs. In some situations this can amount to a large number of records, so the process interval should be set to capture information once per day (1,440 minutes). However, if process records are being aggregated automatically by the system, then it is recommended that the process interval be set for once per hour.

Enabling interval accounting

By default, interval accounting is turned off. Table 5-10 shows the commands you can use to turn on and turn off system-level and process-level interval accounting.

Table 5-10 Interval accounting commands

Task	Command	SMIT fast path
Enable system interval accounting, specified in number of minutes by the time parameter, or turn off system interval accounting.	**acctctl isystem** {time | off}	**smit system_interval**
Enable process interval accounting, specified in number of minutes by the time parameter, or turn off system interval accounting.	**acctctl iprocess** {time | off}	**smit process_interval**
Query the state of Advanced Accounting.	**acctctl**	None

5.18.5 Data aggregation

Data aggregation is a method of accumulating data into an accounting record that is otherwise presented through multiple records. Aggregated data is written periodically according to the intervals specified.

Note:

1. Interval accounting must be enabled in order to use data aggregation.
2. It is recommended that you set the process interval and system interval to 60 minutes.

Accounting data is totaled inside the kernel with no impact to applications or middleware. The data is made available by interval accounting, which is a kernel function that periodically writes these records to the accounting data file. When the kernel aggregates records, it maps them to a set of aggregated records that are managed internally. These records are pending in the sense that they have been input to the system, but have not been committed to the recording

mechanism inside the Advanced Accounting subsystem. Interval accounting is used to commit the aggregated records so that they are written to the accounting data file.

Because aggregated data is recorded using different data structures, you must verify that the billing application recognizes these structures. Consult the documentation provided with your billing application to determine if your billing application supports data aggregation. Data aggregation can be enabled or disabled at the system level or project level.

Enabling system-level data aggregation

Table 5-11 lists the commands you can use to turn on and off process data aggregation, kernel extension data aggregation, and ARM transaction data aggregation.

Table 5-11 System-level data aggregation commands

Task	Command	SMIT fast path
Enable or disable system-wide process aggregation	**acctctl agproc** {**on** | **off**}	**smit system_paggr**
Enable or disable system-wide aggregation for third-party kernel extensions	**acctctl agke** {**on** | **off**}	**smit system_kaggr**
Enable or disable system-wide aggregation for ARM transactions	**acctctl agarm** {**on** | **off**}	**smit system_aaggr**
Query the overall accounting state	**acctctl**	None

Enabling project-level data aggregation

Table 5-12 shows the commands to turn on and off aggregation for individual projects.

Table 5-12 Project-level data aggregation commands

Task	Command	SMIT fast path
Enable or disable aggregation for a project, specified by the projname parameter	**projctl chattr agg** *projname* {**-s**|**-u**} [**-d** *projpath*]	None
Query the aggregation state for all projects	**projctl qprojs** [**-n**]	None
Query the aggregation state for a project, specified by the projname parameter	**projctl qproj** [*projname*]	None

5.18.6 Report and analysis

The Advanced Accounting subsystem provides accounting data for a wide variety of resources that are typically included within chargeback mechanisms, but does not provide report and analysis scripts to produce customer bills. These tools are left to you to implement.

AIX documents the format of the accounting data file as well as the format of the individual accounting records in the /usr/include/sys/aacct.h header file. In addition, AIX provides the sample program **readaacct** with source code that parses the accounting data file. This command may be used to examine accounting data and import accounting data into spread sheets. The following shows the usage message for the **readaacct** command:

```
# /usr/samples/aacct/readaacct -?
/usr/samples/aacct/readaacct: illegal option -- ?
Usage : /usr/sample/bin/readaacct [-F file ] [ -t trid ] [-b begin_time ] [-e
end_time ] [-c] [-h]
-F file         : Specify Accounting File
-t trid         : Prints the details of the specific transaction ID
-b begin_time   : Prints the transaction records collected on and after the
time specified
                  The date format should be MMddhhmmss
-e end_time     : Prints the transaction records collected before the end time
                  The date format should be MMddhhmmss
-c              : Prints the records in colon separated format
-h              : Prints the Accounting File Header
```

The source code for this command is shipped in the same directory as the command so that you can modify the command to produce the relevant information in the required format for your specific billing application. By default, the command displays text strings.

The following is an example of the output displayed if you are running the sample command just after the /var/aacct/acctdata accounting data file was created:

```
# /usr/samples/aacct/readaacct -F /var/aacct/acctdata -h
File Name=/var/aacct/acctdata
Version=1
Flags=0
Offset=4096
File Size=104857600
State=0
ID=0
First Time=0
Last Time=0
System ID=IBM,01100316A
```

```
System Model=IBM,7028-6E1
Host Name=server3
Partition Name=NULL
Partition Number=-1
```

5.18.7 Data file management

The Advanced Accounting subsystem continuously writes accounting data to registered accounting data files. It is necessary to monitor the state of these files to ensure that Advanced Accounting always has a place to record data.

Advanced Accounting produces messages to monitor the state of data files and the subsystem status. Use messages related to the data files to trigger actions based on events as they occur. A message is produced when a file is 90 percent full and when it is 100 percent full, indicating that administrative action is needed. The **acctctl** command is used to manage accounting data files. You can use the **cron** command to periodically execute a shell script or command that examines and manages accounting data files.

Note: By default, AIX records information about the Advanced Accounting subsystem using the syslog daemon.

E-mail notification must be manually configured to function. Table 5-13 lists messages that are sent through e-mail notification.

Table 5-13 E-mail notification messages

Subject line	Body
AACCT: File nearly full	1400-330: The accounting file is 90% full.
AACCT: File ready	1400-331: The accounting file is ready for processing.
AACCT: Subsystem out of file	1400-332: The Advance Accounting subsystem has run out of files for use.
AACCT: Subsystem out of kernel buffers	1400-333: The Advance Accounting subsystem has run out of kernel buffers.
AACCT: File I/O error	1400-334: The accounting file encountered an I/O error while writing.

Most e-mail programs provide filtering capability so that shell scripts and commands can trigger specific actions when messages are received. Use the string AACCT to identify incoming messages related to Advanced Accounting. To identify the significance of the message, use the message number in the body

of the message. The **readaacct** command may be used to extract accounting records from accounting data files. It is a sample command, so you should be aware that it is not formally supported by IBM and is used here for example only.

5.18.8 Accounting records

Advanced Accounting produces several accounting records, which are defined in the /usr/include/sys/aacct.h file. The following paragraphs describe these records.

Pad record (type 0)

This record does not provide any meaningful accounting data. Report and analysis tools should skip this record. It is generated for alignment purposes only.

Process record (type 1)

This record is written when a process exits, when a process is reclassified (setuid(), chproj(), exec()), and when the system is reclassified. This record is written by the process interval.

This record contains the following information:

- User ID
- Group ID
- Process ID
- Process flags (exited, core, killed by signal, killed by checkpoint)
- Base command name
- WLM class
- Controlling terminal
- Process start time (in seconds from the EPOCH)
- Process elapsed time in micro seconds
- Combined thread elapsed time in micro seconds
- Process (combined threads) processor time in micro- seconds
- Elapsed page seconds of disk pages
- Elapsed page seconds of real pages
- Elapsed page seconds of virtual memory
- Local logical file I/O (JFS, J2) in bytes
- Other logical file I/O (NFS, DFS™) in bytes

- Local socket I/O (UNIX domain and loopback) in bytes
- Remote socket I/O in bytes

The process start time and Process ID can be used to correlate interval records for a particular process. The exit flag can be used to distinguish between interval and exit records.

Aggregation process record (type 2)

This record is derived from the process record. A different record is produced for each user by project. This record is produced by the process interval and contains the following information:

- User ID
- Time of first record aggregated (in seconds from the EPOCH)
- Number of processes aggregated
- Aggregate process elapsed time in micro seconds
- Aggregate thread elapsed time in micro seconds
- Aggregate process (combined threads) processor time in micro seconds
- Aggregate elapsed page seconds of disk pages
- Aggregate elapsed page seconds of real pages
- Aggregate elapsed page seconds of virtual memory
- Aggregate local logical file I/O (JFS, J2) in bytes
- Aggregate other logical file I/O (NFS, DFS) in bytes
- Aggregate local socket I/O (UNIX domain and loopback) in bytes
- Aggregate remote socket I/O in bytes

Aggregation application record (type 3)

This record is derived from the process record. Records are produced at the user, project, and application level. This record is similar to the aggregated process record, except that the application is named. This record is produced when the process is classified with an application-specific rule, which is supported only through the Admin policy. This record is produced by the process interval and contains the following information:

- User ID
- Time of first record aggregated (in seconds from the EPOCH)
- Inode of the command that generated the project classification
- Device number of the command that generated the project classification
- Number of applications aggregated

- Aggregate process elapsed time in micro seconds
- Aggregate thread elapsed time in micro seconds
- Aggregate process (combined threads) processor time in micro seconds
- Aggregate elapsed page seconds of disk pages
- Aggregate elapsed page seconds of real pages
- Aggregate elapsed page seconds of virtual memory
- Aggregate local logical file I/O (JFS, J2) in bytes
- Aggregate other logical file I/O (NFS, DFS) in bytes
- Aggregate local socket I/O (UNIX-domain and loopback) in bytes

Processor and memory use record (type 4)

This record provides information about the use of processors and memory at the system level. It is generated before and after Dynamic Logical Partitioning operations and when the size of the large page pool changes. This record is also generated by the system interval and contains the following information:

- Reason the record was generated
- Number of online logical processors
- Entitled physical processor capacity of the partition
- Total idle time, in milliseconds
- Total I/O wait time, in milliseconds
- Total kernel process time, in milliseconds
- Total user process time, in milliseconds
- Total interrupt time, in milliseconds
- Size of physical memory, in megabytes
- Size of the large page pool, in megabytes
- Large pages in use, in megabytes
- Number of page ins from paging space
- Number of page outs to paging space
- Number of start I/Os v Number of page steals

Policy record (type 5)

This record is written when a policy file is loaded or unloaded. It is provided for informational purposes only. This record contains the following information:

- Type of policy: Admin, User, or Group

- Load or unload

File system activity record (type 6)

This record describes the use of file systems at the system level. A separate record is generated for each mounted file system. This record is produced by the system interval and has the following information:

- Total bytes transferred through read and write
- Total number of read and write requests
- Total number opens
- Total number of creates
- Total number of locks
- File system type
- Device name
- Mount point

Network interface I/O record (type 7)

This record provides information about the use of network interfaces at the system level. This record is produced by the system interval and contains the following information:

- Logical name of the network interface
- Number of I/Os
- Total bytes transferred

Disk I/O record (type 8)

This record provides information about the use of disks at the system level. A separate record is written for each logical disk device. This record is produced by the system interval and contains the following information:

- Logical name of the disk
- Total disk transfers
- Total reads
- Total writes
- Block size of the disk transfer

Lost data record (type 9)

This record provides information about accounting records that were deleted because Advanced Accounting did not have the ability to record them. This occurs when all of the accounting data files are full. When the ability to write new

accounting records is restored, Advanced Accounting produces the lost data record describing the outage. This record contains the following information:

- Number of lost records
- Number of microseconds of lost processor time associated with process records
- The time that data loss began (in microseconds from EPOCH)
- Number of microseconds of lost processor time associated with third party kernel extension records

Server I/O record (type 10)

This record is produced in hosting partitions. A separate record is produced for each logical device that is shared with a client partition. The system interval may be used to periodically produce this record. This record contains the following information:

- Client partition number
- Server unit ID
- Device logical unit ID (LUN)
- Number of bytes in
- Number of bytes out

Client VIO record (type 11)

This record is produced in client partitions. It describes the use of virtual devices in client partitions. A separate record is recorded for each instance of a virtual device. The system interval may be used to periodically produce this record. This record contains the following information:

- Server partition number
- Server unit ID
- Device logical unit ID
- Number of bytes in
- Number of bytes out

Third-party kernel extension common aggregation record (type 12)

This record provides accounting information for the named accounting record. It is derived from aggregated accounting records that are produced by third-party kernel extensions. This record is written to the Advanced Accounting subsystem by the system interval. This record contains the following information:

- Command name of the kernel extension (from u-block)
- Third-party kernel extension transaction ID, in the range of 129 to 256

- Number of accounting records that have been aggregated
- Resource use, or accumulated processor time, for this class of transactions
- Time of first record aggregated (in seconds from the EPOCH)

ARM application environment record (type 13)

This record describes an application environment instance. It is created from data that is passed to the operating system through the arm_register_application() system call and the arm_start_application() system call. This record is variable in length. All offsets are calculated relative to the start of the record. This record contains the following information:

- Character set in which the data in this record is recorded
- Application environment identifier
- Offset to application name
- Offset to application group
- Offset to application identity properties
- Offset to application context properties

The operating system attempts to record the content of the application environment in each accounting data file, so that each accounting data file can be post-processed as a stand-alone item. This is designed to eliminate the dependency between accounting data files.

ARM transaction environment record (type 14)

This record describes a transaction environment instance. It is created from data that is passed to the operating system through the arm_register_transaction() system call. This record is variable in length. All offsets are calculated relative to the start of the record. This record contains the following information:

- Character set in which the data in this record in recorded
- Transaction environment identifier
- Offset to transaction name
- Offset to application identity properties
- Offset to application context properties (names only)

The operating system attempts to record the content of the transaction environment in each accounting data file (not guaranteed), so that each accounting data file can be post-processed as a stand-alone item. This is designed to eliminate the dependency between accounting data files.

ARM transaction instance record (type 15)

This record describes an ARM transaction instance. It is created from data that is passed to the operating system through the arm_start_transaction() and the arm_stop_transaction() system calls. It is variable in length. All offsets are calculated relative to the start of the record. This record contains the following information:

- Completion status of the transaction
- Application environment identifier
- Transaction environment identifier
- Offset to user identifier (not User ID)
- Offset to user name (not uname)
- Offset to accounting code
- Response time, in milliseconds
- Queued time, in milliseconds
- Resource use

The application and transaction environment identifiers are defined respectively in the application and transaction environment records. These records must be used to associate application names, application groups, transaction names, and properties with the transaction instance.

ARM aggregated transaction instance record (type 16)

This record is produced instead of the ARM transaction instance record (type 15), when aggregation is enabled for ARM transactions. This record contains the following information:

- Completion status of the transaction
- Time of first record aggregated (in seconds from EPOCH)
- Application environment identifier
- Transaction environment identifier
- Offset to user identifier (not User ID)
- Offset to user name (not uname)
- Offset to accounting code
- Aggregate response time, in milliseconds
- Aggregate queued time, in milliseconds
- Aggregate resource use

Project definition record (type 17)

This record provides a list of project definitions. It is written when the project definition file is loaded. Multiple records may be needed to record all project definitions. This record is used to provide the full set of project information in each data file, so that data files may be treated as stand-alone entities. This may not be required by the billing application, depending on the nature of the billing application. This feature may be disabled by disabling the project definition accounting record. This record is variable in length and contains the following information:

- Number of projects
- Number of bytes in the project definition area
- Project definition area

5.19 Date APIs past 2038

The current time API does not allow the representation of times past Tuesday, January 19, 2038 at 03:14:07 UTC (Coordinated Universal Time). UNIX systems represent times as the number of seconds past January 1, 1970 at 00:00:00 UTC (the epoch). The time_t type is a 32-bit signed integer on both 32- and 64-bit machines. On January 19, 2038 03:14:08 UTC it will be more than 2^31 seconds making it impossible for the date to be represented by a time_t type. Applications wishing to represent dates past 2038 cannot rely on the time API because of this limitation.

AIX 5L Version 5.3 has introduced a new 64-bit time API that will allow user programs to use dates past 2038. It will use the time64_t type instead of the time_t type. With the new API the user will be able to call the new time functions to manipulate times up till December 31, 9999.

The following functions have been added to Version 5.3:

- ctime64()
- ctime64_r()
- localtime64()
- localtime64_r()
- gmtime64()
- gmtime64_r()
- asctime64()
- asctime64_r()
- difftime64()
- mktime64()

Each function mirrors its respective time32 counterpart. The only difference is time_t has been replaced by a new datatype time64_t which is a signed 64-bit integer.

Note: The new interface will not provide function to set the date past 2038. AIX 5L Version 5.3 will, most likely, not be supported past 2038.

```
# date 070812562050

date: 0551-403 Format of date/time specification is not valid.
```

5.20 System V printer enhancements

AIX 5L Version 5.3 includes the following new or enhanceed functions for System V printing:

- The remote lpd printing daemon **`lpNet`** now provides improved speed and performance.
- The same JetDirect software now services both AIX printing and System V printing. This leads to a more robust connection to the printing devices.
- The **`lpsched`** and **`lpNet`** programs have been made more secure.
- Performance of **`lpsched`** has been enhanced by improving the filtering process.
- Error reporting from **`lpsched`** and **`lpNet`** has been improved.

5.21 Mozilla browser for AIX

AIX 5L Version 5.3 introduces support for the Mozilla 1.4.2 Web browser as the default browser for AIX. Netscape Communicator Version 4 is not supported on AIX 5L Version 5.3.

The Mozilla Web browser for AIX is available on a CD that can be ordered with AIX, or it can be downloaded from the following Web site:

`http://www.ibm.com/servers/aix/browsers`

Mozilla for AIX requires GNOME libraries, which are available on the AIX Toolbox for Linux Applications CD or from the following Web site:

`http://www.ibm.com/servers/aix/products/aixos/linux`

5.21.1 Installing Mozilla for AIX

Mozilla for AIX can be installed as an option during the AIX Base Operating System installation process, or it can be installed later. All listed installation methods use the Mozilla installation bundle, which includes Mozilla and the required GNOME libraries.

The Mozilla installation process fails if the required GNOME libraries are not found. The required rpm filesets are listed in the Mozilla install log.

Use one of the following installation methods:

- Install Mozilla using the following AIX BOS installation process:
  - You can select Mozilla for installation during the AIX Base Operating System installation process by selecting these options in the following order:
    - 2 = Change/Show Installation Settings and Install
    - 3 = More Options
    - 6 = Install More Software
    - 1 = Mozilla (Mozilla CD)
  - The default setting is to *not* install Mozilla.
  - When prompted to do so, insert the Mozilla CD and the AIX Toolbox for Linux Applications CD.
- Install Mozilla as a bundle using the following Configuration Assistant process:
  - Start **`configassist`**.
  - Select Manage software, and click Next.
  - Select Install additional software, and click Next.
  - Select Install by bundle, and click Next.
  - Specify the device or directory that contains the installation images, and click Next. If the location is a directory, such as /usr/sys/inst.images, verify the following:
    - The Mozilla.base installp package is in the /usr/sys/inst.images/installp/ppc directory
    - The toolbox rpm filesets are in the /usr/sys/inst.images/RPMS/ppc directory
  - Select the Mozilla bundle, and click Next.
  - Accept the license agreement, and click Next to start the installation process.

- Install Mozilla as a bundle using the following **smit** or **smitty** process:
  - Run the **smitty install_bundle** or **smit install_bundle** command.
  - Specify the INPUT device and directory for software. If the location is a directory, such as /usr/sys/inst.images, verify the following:
    - The Mozilla.base installp package is located in the /usr/sys/inst.images/installp/ppc directory
    - The toolbox rpm filesets are located in the /usr/sys/inst.images/RPMS/ppc directory
  - Select the Fileset Bundle = Mozilla.
  - In the Install Software Bundle screen, accept the license agreement, and press Enter to start the installation process.

5.21.2 Mozilla as the browser for AIX Documentation Services

Mozilla can be configured as the default browser that is used to view the AIX Documentation using Configuration Assistant or SMIT.

- Configure Mozilla using the following Configuration Assistant process:
  - Start **config_assist**.
  - Select the Configure documentation server task.
  - If Mozilla is detected as already installed, select Yes, use Mozilla as the default browser, and click Next.
- Configure Mozilla using the following **smit** or **smitty** process:
  - Run the **smit change_documentation_services** command or the **smitty change_documentation_services command**.
  - Verify that the /usr/bin/mozilla directory is the DEFAULT_BROWSER.

5.21.3 Migrating Netscape Communicator Version 4 profile

If a Netscape Communicator Version 4 profile exists in your home directory and Mozilla is run for the first time, Mozilla prompts whether or not it should convert the Communicator profile, including the bookmarks to be used within Mozilla.

For more information about Mozilla for AIX, see the file:///usr/mozilla/base/README.HTML file after Mozilla is installed.

5.22 Uniprocessor kernel support

The AIX 5L operating system previously contained both a uniprocessor 32-bit kernel and a multiprocessor 32-bit kernel. Effective with AIX 5L Version 5.3, the operating system supports only the multiprocessor kernel.

The AIX 5L Version 5.3 32-bit multiprocessor kernel supports the following systems: RS/6000, @server pSeries, p5, or OEM hardware based on the Common Hardware Reference Platform (CHRP) architecture, regardless of the number of processors.

AIX 5L Version 5.2 is the last release of AIX that supports the uniprocessor 32-bit kernel.

5.23 Enhancement of base commands and libraries

AIX 5L Version 5.3 includes improvements to the following base commands and libraries:

- The **find** command has been enhanced to provide information on file access and changes in the last N minutes (instead of days).
- The SMIT panels have been updated to include the latest set of flags for the **iostat** command.
- The following two new operators ("+=" and "%") and a special target .SCCS_GET have been added to the **make** command:

  += Appends a value to an existing variable's value

  % Implements pattern matching for substitution

  .SCCS_GET target source Specifies the commands for retrieving source files from SCCS
- The **ps** command has been enhanced to provide process hierarchy information and a listing of descendant processes for given pid(s).
- A new flag has been added to the **tar** command which will specify the list of files and/or directories to be excluded from the **tar** file being created, extracted, or listed.
- Flags have been added to the **tar** command to process directory of files recursively. An option has also been added to specify an input file for tar extraction much like that which can be used for **tar** command file creation.
- The **fuser** command has been enhanced to accept any of the signals displayed by the **kill -l** command. Only the root user can kill a process of another user.

- AIX 5L text processing commands now support unlimited line lengths except for **ed/ex/vi/awk**. For **ed/ex/vi** the line length limit is 8192 characters and for **awk** it is 100 K characters.
- Command buffer sizes are unlimited now. This allows an increased number of arguments on the command line and reduces the number of failures of the following type:

```
grep foo /usr/include/*/*
ksh: /usr/bin/grep: arg list too long
```

- An option has been added to the **restore** command to exit on error rather than recover and continue. An option has also been added to provide a long style listing of backup files as in **ls -l.**
- Options have been added to the **grep** command to do recursive search.
- The following two formats have been added to the **date** command:
  - %k representing the 24-hour-clock hour clock as a right-justified space-filled number (0 to 23).
  - %s representing the number of seconds since January 1, 1970, Coordinated Universal Time (CUT).
- The **snap** command now includes installed package information from System V and rpm.
- If Manual Pages with the same title are resident under directories in MANPATH and /usr/share/man/info, previously it displayed all matching entries. Now a flag has been added to display only the first matching entry.
- When a new environment variable EXTENDED_HISTORY is set the shell histories in **ksh** and **ksh93** will have time stamp.
- AIX 5L Version 5.3 now includes a restricted version of **ksh** and **ksh93**.
- A backtag function has been added to the **vi** command. What this means is that if you use ctl-] to tag forward to a specific line, then using ctl-T returns you to the previous position from where you tagged forward. It maintains a stack of tags. Anytime that you do ctl-] it pushes the previous position onto the stack, and anytime you do ctl-T, it pops the previous position off of the stack and changes the current editing position to be that position.
- The **cron** function now accepts user-specified location, type, and size for the cron log file when specified in the /etc/cronlog.conf configuration file.
- Queues g-z are now available for user-defined queues for **at** jobs.
- A flag has been added to the **at** command so that you can see both the time of the job and the actual command that is scheduled to run.
- An option has been added to **nohup** command to nohup an existing process.

6

Performance monitoring

This chapter describes the updated and new performance monitoring tools included in AIX 5L version 5.3. The following enhancements are included:

- The **procmon** command - process monitoring tool
- Asynchronous I/O statistics
- Disk service and wait time monitoring
- PMAPI M:N pthreads support
- Support for Micro-Partitioning and SMT
  - perfstat library
  - **vmstat** command enhancements
  - **iostat** command enhancements
  - **mpstat** command - CPU performance monitoring
  - **lparstat** command
  - **sar** command enhancements
  - **topas** command enhancements
- Per file system I/O pacing

6.1 General information

The *stat* commands (**iostat**, **vmstat**, **sar**, and the new **mpstat** and **lparstat** commands) now can detect and tolerate system configuration changes. This new function requires output changes. All five commands now display a system configuration line, which appears as the first line displayed after the command is invoked. Scripts that parse command output may need to be modified to detect and handle this new output. If a configuration change is detected during a command execution iteration, a warning line will be displayed before the data which is then followed by a new configuration line and the header.

There are many new tunable parameters added to AIX in AIX 5L Version 5.3. Some have also appeared in AIX 5L Version 5.2 ML5 which was made available at the same time. It is always a good policy to check for new options when major system service is applied as these enhancements do not always appear in product documentation. Use the -L option with the **ioo**, **vmo**, **schedo**, **no**, and **nfso** commands to get the current list of tunable parameters. Use the -h option to display their descriptions and usage.

6.2 The procmon command - process monitoring tool

procmon is an Eclipse-based tool that graphically displays performance characteristics of processes in the system. There are two different uses of this tool:

- It is a monitoring tool that displays a dynamic, sorted list of processes, and information about them.
- It provides the ability to carry out actions on the processes listed. These actions are basic administration commands such as **kill**, **renice** for example, as well as other advanced tools such as **svmon** and **proctools**.

The output displayed (Figure 6-1 on page 257) has the following flexibility:

- Columns can be configured
  - Select sorting column
  - Select ascending/descending order
  - Change order of columns
  - Add or delete columns
- Table can be searched, for example to highlight:
  - All processes with a common ppid
  - All processes belonging to a given user

- Highlights are maintained after refresh
- Actions available on processes include:
  - **kill**
  - **renice**
  - Show list of threads (also in a sorted table)
  - Run performance tools (**svmon** and **proctools**)

    Shows text results in a new window.

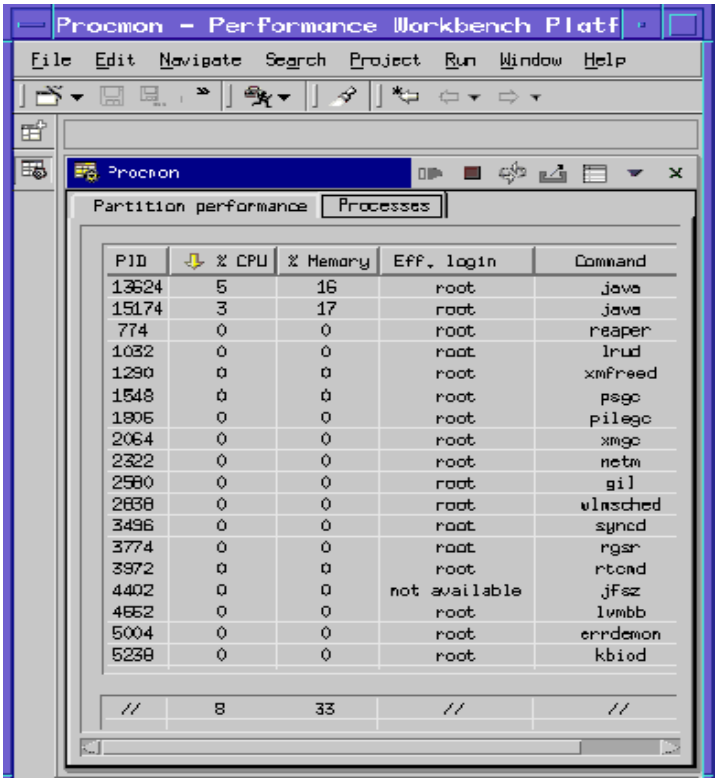

Figure 6-1 procmon - new process monitoring tool

Figure 6-2 on page 258 displays a sample of global metrics for partition performance.

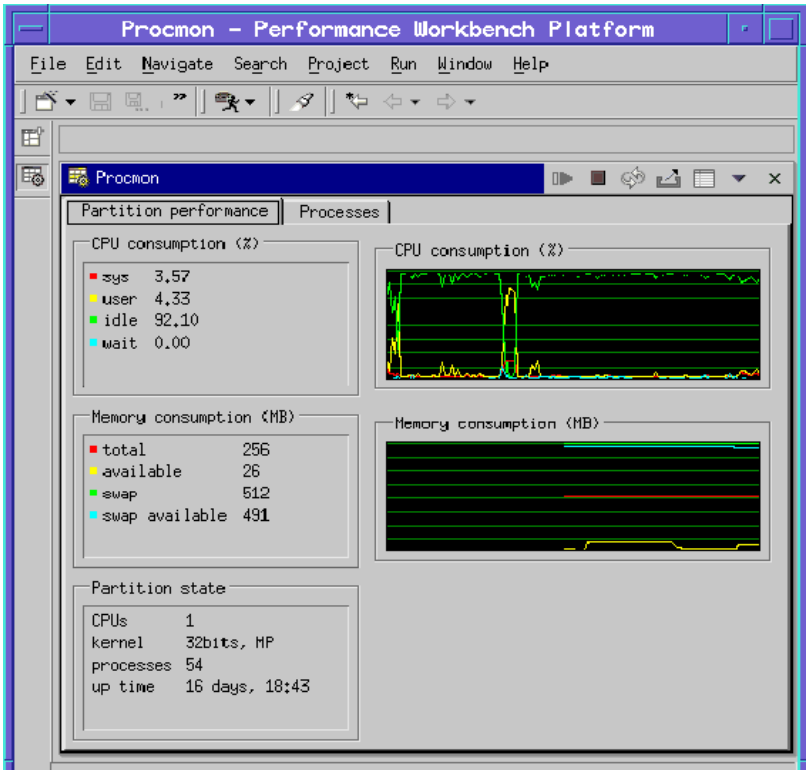

Figure 6-2 Global metrics for partition performance

6.3 Asynchronous I/O statistics

AIX 5L Version 5.3 introduces enhancements to the **iostat** command that allow the user to obtain AIO statistics. To give you a better understanding of the new features, we begin this discussion with an overview of the already existing AIO device driver logic, followed by an explanation of the new enhancements.

6.3.1 Asynchronous I/O basics

The Asynchronous I/O (AIO) as well as its POSIX version are already available in the previous versions of AIX. AIO is an environment that speeds up the application access to the files by using separate kernel processes (aioserver) to do the real I/O transfer. There are two AIO drivers available in the system:

- Legacy AIO
- POSIX AIO

Both of the drivers are similar in function. The implementation of the POSIX AIO complies with the POSIX standards and also implements its extensions. Since the internal logic of both drivers is the same, we are describing the AIO logic using the legacy AIO driver and point out the differences for the POSIX AIO driver.

Figure 6-3 shows how the AIO flow goes from the application to the logical volume.

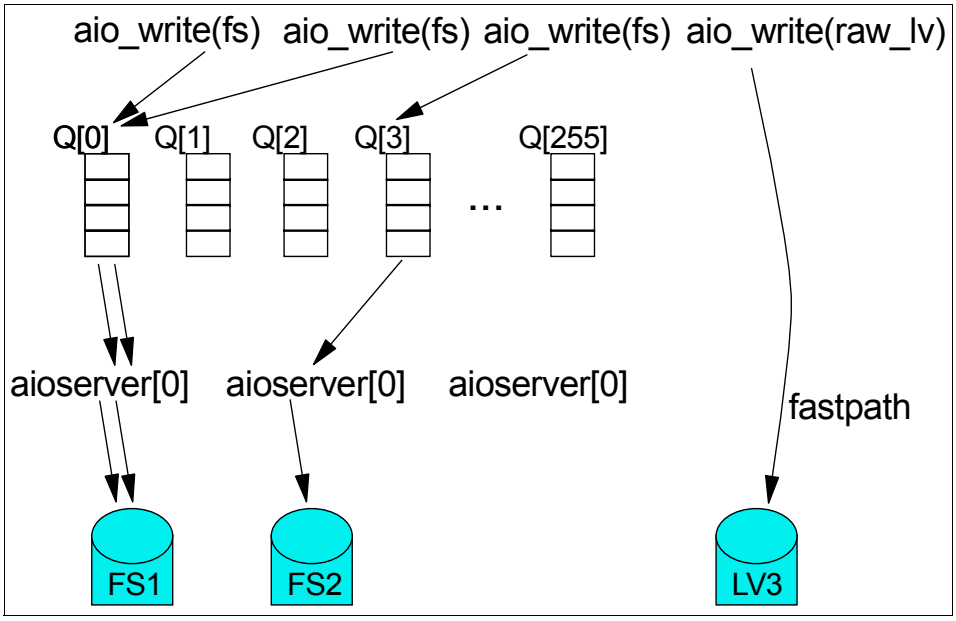

Figure 6-3 AIO flow

The user program calls the aio_write(), aio_read(), or the lio_listio() system call to write or read a data buffer to or from a file. The file can be a regular file or also a raw device.

Today, there is a fixed number of preallocated AIO request queues in the system, but this number may change in the future. The AIO device driver allocates one of those queues and puts the data from the buffer into the allocated queue. There is an internal logic of assigning the queues to specific AIO requests. Usually writes to the same file system go into the same queue, while writes to different file systems are preferred to go to different queues. However, there may be situations when two or more different file system requests go into the same queue or requests to the same file system are spread to several queues.

The AIO requests are then picked up by aioserver kernel processes, which transfer each AIO request to the output file. The POSIX AIO kernel processes are named posix_aioserver. You can check them by using, for example, the **ps aux** command.

If the destination file is a raw device (raw logical volume), the AIO device driver will use the fastpath routine instead of the AIO queues. The fastpath routine moves the input buffer of the AIO system call directly to the raw device, thus bypassing the AIO and its queues.

In order to use the AIO drivers we have to enable the AIO device driver using the **mkdev -l aio0** command or through **smit aio**. In order to enable the POSIX AIO device drivers, you have to use the **mkdev -l posix_aio0** or **smit posixaio** commands.

Using SMIT and going through menus **Devices** → **Asynchronous I/O** → **Asynchronous I/O (Legacy)** → **Change/Show Characteristics of Asynchronous I/O** you can set the characteristics of the AIO, like minservers, maxservers, maxreqs, kprocprio, autoconfig, or fastpath.

6.3.2 AIO enhancement of the iostat command in Version 5.3

In the previous versions of AIX there were no tools available to monitor the AIO; from Version 5.3 the **iostat** command is enhanced to monitor the AIO.

The **iostat** command reports CPU and I/O statistics for the system, adapters, TTY devices, disks, and CD-ROMs. This command is enhanced by new monitoring features and flags for getting the AIO and the POSIX AIO statistics.

The following new flags are added to the **iostat** command:

-A Reports legacy AIO statistics along with utilization metrics.

-P Reports POSIX AIO statistics along with utilization metrics.

-q Reports each AIO queue's request count.

-Q Reports AIO queues associated with each mounted file system and the queue request count.

-l Displays the data in a 132 column width. This flag is simply a formatting flag.

The iostat -A option

When using the -A option, the output of the **iostat** command gives additional statistics of the AIO. The following information is added:

avgc Average global non-fastpath AIO request count per second for the specified interval.

avfc Average AIO fastpath request count per second for the specified interval.

maxg Maximum non-fastpath AIO request count since the last time this value was fetched.

maxf Maximum fastpath request count since the last time this value was fetched.

maxr Maximum AIO requests allowed. This is the AIO device maxreqs attribute.

When the AIO device driver is not configured in the kernel, the **iostat -A** command gives an error message that the AIO is not loaded, such as in the following example:

```
# iostat -Aq

aio: avgc avfc maxg maxf maxr avg-cpu: %user %sys %idle %iow physc %entc
iostat: 0551-157 Asynchronous I/O not configured on the system.
```

Example 6-1 demonstrates the use of the **iostat -A** command.

Example 6-1 iostat command AIO statistics in a Micro-Partitioning environment

```
# iostat -A 1 1

System configuration: lcpu=4 drives=1 ent=0.50

aio: avgc avfc maxg maxf maxr avg-cpu: %user %sys %idle %iow physc %entc
       25    6   29   10 4096           30.7 36.3  15.1 17.9   0.0  81.9

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0          100.0     61572.0     484.0       8192     53380
```

Notice the avgc column, which it shows the average number of AIO requests in the queues; and the maxg column, which shows the maximum number of AIO requests in the queues for the last measuring period. If the avgc or maxg is getting close to the maxr then tuning of the maxreqs and maxservers attributes is required.

If you use raw devices, the avfc is interesting because it reflects the use of the average AIO fastpath calls, as is the maxf, which shows the maximum value of fastpath count value. Since the fastpath calls bypass the AIO queues, these statistics give only information on how fastpath AIO is used.

The iostat -Aq option

If you are interested in the allocation of the AIO queues and their use, then the **iostat -Aq** command is useful. Example 6-2 on page 262 shows output from this command.

Example 6-2 Output of the iostat -Aq command with AIO queue statistics

```
# iostat -Aq 1 1

System configuration: lcpu=4 ent=0.50

aio: avgc avfc maxg maxf maxr avg-cpu: %user %sys %idle %iow physc %entc
       25   10   29   10 4096           29.5 37.9  23.0  9.5   0.0  83.6

q[  0]=0       q[  1]=0       q[  2]=0       q[  3]=0       q[  4]=0
q[  5]=0       q[  6]=0       q[  7]=0       q[  8]=0       q[  9]=0
...
q[120]=0       q[121]=0       q[122]=0       q[123]=0       q[124]=0
q[125]=0       q[126]=0       q[127]=0       q[128]=0       q[129]=0
q[130]=0       q[131]=0       q[132]=0       q[133]=0       q[134]=0
q[135]=0       q[136]=0       q[137]=0       q[138]=0       q[139]=25
q[140]=0       q[141]=0       q[142]=0       q[143]=0       q[144]=0
...
q[250]=0       q[251]=0       q[252]=0       q[253]=0       q[254]=0
q[255]=0       q[256]=0       q[257]=0       q[258]=0
```

If statistics for any single queue are significantly higher than the others, this indicates that applications are using one file system significantly more than other file systems. The queues are usually allocated one per file system.

The iostat -AQ option

If a specific AIO queue is filling up unusually and you want to know to which file system the queue is related, then the **`iostat -AQ`** command is useful. Figure 6-3 shows the distribution of the AIO queues to specific file systems, as shown in Example 6-3.

Example 6-3 The iostat -AQ command output

```
sq 156 # /ad/iostat.fix -AQ 1 1
System configuration: lcpu=4 ent=0.50
aio: avgc avfc maxg maxf maxr avg-cpu: %user %sys %idle %iow physc %entc
       26    7   29   10 4096           27.1 36.4  19.6 16.8   0.0  78.1

Queue#         Count          Filesystems
129            0              /
130            0              /usr
132            0              /var
133            0              /tmp
136            0              /home
137            0              /proc
138            0              /opt
139            26             /ad
```

The iostat -P option and related options for POSIX AIO

In order to get the basic POSIX AIO statistics we have to use the **iostat -P** command. The output is in the same format as that for the **iostat -A** legacy AIO command; the meaning of the metrics is the same as well, but related to the POSIX AIO calls.

The output format of the **iostat -Pq** command corresponds with the **iostat -Aq** and the **iostat -PQ** corresponds with the **iostat -AQ** command.

Other iostat -A usage

There may be several possible combination of the **iostat** flags, but it is better to run the **iostat** command twice in two different windows then use too many flags and get too much information. There are also several unsupported combinations of flags. One example of a reasonable use would be the **iostat -A -l hdisk3 hdisk4 1 10** command that has combined flags.

6.4 Disk service and wait time monitoring

The following sections discuss the enhancements made in AIX 5L Version 5.3 in the area of disk service and wait time monitoring.

6.4.1 dkstat and device support

The dkstat structure is the base statistical interface defined in the <sys/iostat.h> header file that is used by the device drivers to provide the statistical data and the performance monitoring tools, such as **iostat -D** or **sar -d**, to read the data.

The dkstat structure is updated in AIX 5L Version 5.3 to support additional statistics relative to disk service times. The dkstat structure is referenced in the /usr/include/sys/iostat.h header file.

6.4.2 avwait and avserv enhancements for the sar -d

Starting with Version 5.3, the -d flag is modified in the **sar** command. The -d flag of the **sar** command is intended to give statistics about the I/O activity on the disks. The **sar -d** command reads the dkstat structures of the devices that supply the necessary information.

In older versions of AIX, avwait and avserv are always reported as zero. From Version 5.3 the avwait and avserv are implemented. The meaning of the avque statistics has changed from Version 5.3. See Example 6-4 on page 264 for sample output of the **sar -d** command.

Example 6-4 sar -d command output

```
# sar -d 1 2

AIX sq1test1 3 5 00CDDEDC4C00    06/22/04
System configuration: lcpu=2 drives=1 ent=0.30

10:01:37     device    %busy    avque    r+w/s    Kbs/s   avwait   avserv

10:01:38     hdisk0    100      36.1      363    46153     51.1      8.3

10:01:39     hdisk0     99      38.1      350    44105     58.0      8.5

Average      hdisk0     99      37.1      356    45129     54.6      8.4
```

There are two new measured statistics and one that has been modified:

avwait Average wait time per request in milliseconds. This is the time that the request is waiting in the device driver I/O queue to be passed to the I/O device.

avserv Average service time per request in milliseconds. This is the length of time between when the I/O is passed to the device to when the result is returned back from the device.

avque Average number of requests waiting to be sent to disk. Imagine this as the average length of the occupied I/O queue.

There is a change in the avque interpretation introduced in Version 5.3. The avque statistics used to represent the instantaneous number of requests sent to disk but not completed yet in the previous versions of AIX.

The other columns, like Kbs/s, r+w/s, %busy, device and so on have the same meaning as before.

6.5 PMAPI M:N pthreads support

Under the M:N threading model, M user threads are mapped to N kernel threads, with M being typically considerably bigger than N to allow large numbers of pthreads to run. Making PMAPI calls from a program running in this mode was previously not supported.

The PMAPI library has been updated to handle the M:N thread model; the current unchanged interfaces simply work in M:N mode. The only significant change is for third party API callers, for example debuggers, where new interfaces with pid, tid, and ptid must be used.

6.6 Support for Micro-Partitioning and SMT

The introduction of Micro-Partitioning and Simultaneous Multi Threading (SMT) required additional methods for recording utilization to be added to hardware and used by AIX.

The Process Utilization Resource Register (PURR) is a new register, provided by the POWER5 processor, which is used to provide an actual count of physical processing time units that a logical processor has used. All performance tools and APIs utilize this PURR value to report CPU utilization metrics for Micro-Partitioning and SMT systems.

6.6.1 perfstat library

The performance library API libperfstat is enhanced to capture the new PURR-based utilization statistics.

The following enhancements are introduced by AIX 5L Version 5.3 to the libperfstat library:

perfstat_cpu_t The structure defines the basic processor statistics on a per processor basis. The structure is extended with PURR-based utilization and Micro-Partitioning statistics. It has also been extended with statistics displayed by the **mpstat** command, like affinity and interrupt statistics.

perfstat_cpu_total_t The structure defines the basic processor statistics system-wide. The structure is extended with PURR-based utilization and Micro-Partitioning statistics. It has also been extended with more detailed interrupt statistics.

perfstat_partition_total_t
This is a new structure that contains partition-specific information and statistics similar to what the **lparstat** command can display.

perfstat_partition_total()
This is a new function that returns information in a perfstat_partition_total_t structure.

6.6.2 vmstat command enhancements

The **vmstat** command has been enhanced to support Micro-Partitioning and can now detect and tolerate dynamic configuration changes.

The **vmstat** command has two new metrics that are displayed: physical processor consumed and percentage of entitlement consumed. They are

represented as pc and ec in the output format. The physical processor consumed represents the number of physical processors consumed by the partition during an interval. The percentage of entitlement consumed is the percentage of entitled capacity consumed by a partition during an interval (pc/ent)*100. These new metrics will be displayed only when the partition is running as a shared processor partition. If the partition is running as a dedicated processor partition the new metrics will not be displayed.

Changed from previous releases, the first interval of the command output is now meaningful and does not represent statistics collected from system boot. Internal to the command the first interval is never displayed, and therefore there may be a slightly longer wait for the first displayed interval to appear. Scripts that discard the first interval should function as before.

With these new changes, the output of the **vmstat** command will be as shown in the following examples.

On a dedicated partition:

```
# vmstat
kthr    memory              page              faults                    cpu
---- ------------- ---------------------- ------------ -----------------------
r  b  avm     fre       re  pi  po  fr  sr  cy  in       sy   cs  us  sy  id  wa
1  1  19782   32762      0   0   0   0   0   0  125      10   47  50  50  0   0
```

On a Micro-Partitioning machine:

```
# vmstat
kthr   memory             page                       faults                  cpu
----  -------------      ------------------------     ------------ -----------------------------------
r  b  avm     fre        re  pi  po  fr  sr  cy  in     sy   cs     us  sy  id  wa   pc   ec
1  1  19782   32762       0   0   0   0   0   0 125     10   47     50  50   0   0   1.5  85
```

6.6.3 iostat command enhancements

Beginning with AIX 5L Version 5.3, the **iostat** command reports the number of physical processors consumed (physc), the percentage of entitled capacity consumed (% entc), and the processing capacity entitlement when running in a shared processor partition. These metrics will only be displayed on shared processor partitions.

Changed from previous releases, the first interval of the command output is now meaningful and does not represent statistics collected from system boot. Internal to the command, the first interval is never displayed, and therefore there may be a slightly longer wait for the first displayed interval to appear. Scripts that discard the first interval should function as before.

The following **iostat** command output shows an example. The newly introduced information is in bold type.

```
# iostat

System configuration: lcpu=4 drives=2 ent=0.30

tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait physc   % entc
          0.3           98.2               8.9       0.2       90.8       0.2 0.03    10.2

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk1           0.2       5.9       1.5        128     14360
hdisk0           0.0       0.0       0.0          0         0
#
```

Percentage of entitled capacity consumed is display in the %entc column. The number of physical processors consumed is shown in the physc column.

In the system configuration information you can see the currently assigned processing capacity specified as ent.

6.6.4 mpstat command - CPU performance monitoring

The **mpstat** command is the basic monitoring tool for logical CPU usage. It accepts the following flags:

no flag Basic statistics

-d Dispatcher statistics

-i Interrupt statistics

-s SMT statistics

-a Display all statistics (wide output)

-w Format output to wide columns. The -a flag implicitly turns on this flag.

In the following sections we describe the use of the main **mpstat** command flags to obtain basic (no flag), dispatcher (-d), interrupt (-i) and SMT (-s) statistics. The -a and -w flags are focussed on the formatting of the output; they combine the statistics received using the described flags.

Enhanced basic statistics

The calculation of the performance statistics has gone through significant changes because of the Micro-Partitioning and SMT technologies. In previous versions of AIX the percentage of the processor usage calculation was based on calculating the average usage. The calculation was simple:

```
avg_usage = (cpu1_usage + cpu2_usage + cpun_usage) / n
```

In the case of Micro-Partitioning architecture we have to consider that the system-wide utilization is reported as a percentage of the allocated entitled capacity and that there could be an unused slice of each entitled processor capacity. This unused part needs to be integrated into the calculations. Figure 6-4 shows the basics for statistics calculations.

Note: The enhanced statistics also apply to the `sar -P` command.

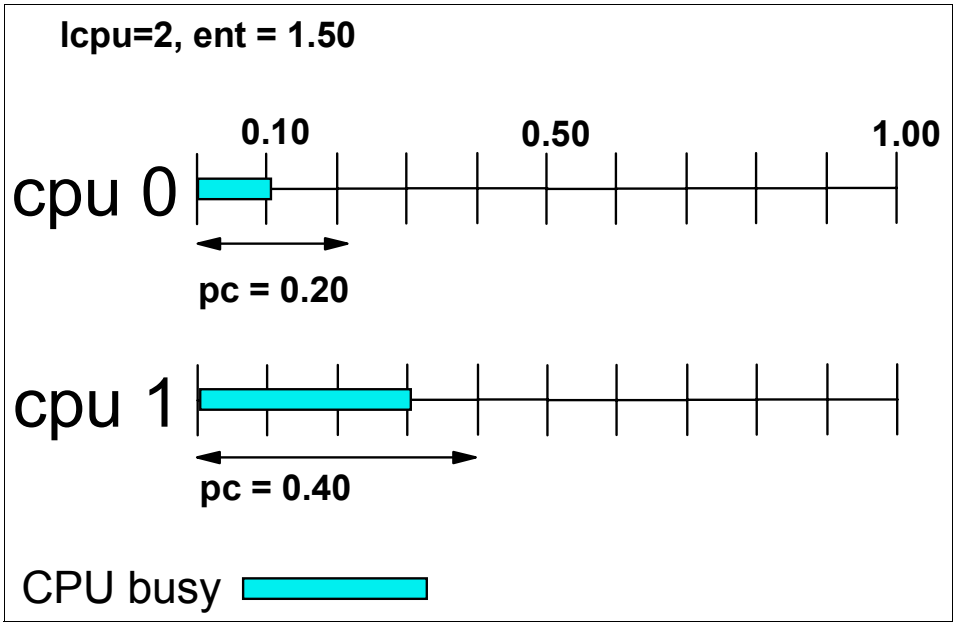

Figure 6-4 CPU statistics for POWER5 systems

lcpu, ent, pc, %ec

In Figure 6-4 we have one physical processor that maps to two logical processors (cpu0, cpu1), using SMT. The number of logical processors is marked by lcpu.

The processor entitlement is the CPU capacity we are entitled to use. We are using the *ent* metric for the total entitlement. In Figure 6-4 we have a 1.50 processor entitlement.

The entitlement does not need to be consumed entirely. If there are idle processor cycles, they can be used by other partitions. The processors can use different fractions of the entitlement. The fraction of the processor that is consumed by the partition using Micro-Partitioning technology is marked by pc. In Figure 6-4 the fraction of processor consumed is pc(0)=0.20 for processor 0

and pc(1)=0.40 for processor 1. The total fraction of processor consumed is pc(all)=0.60. There is also an unused fraction of the processors pc(u)=ent - pc(all)=0.90. The processing capacity consumed is the sum of the pc: pc(0)+pc(1)=pc(all), and pc(all)+pc(u)=ent.

The %ec is the percentage form of the usage of the entitlement. The cpu0 is using %ec(0)=13.3%=100*pc(0)/ent and cpu1 is using %ec(1)=26.7%=100*pc(1)/ent. The unused part of the entitlement is %ec(u)=60%=100*pc(u)/ent. The total percentage of entitlement consumed, or %ec is the sum %ec(0)+%ec(1)=ec(all), and ec(all)+%ec(u)=100%

us, sy, wa, id

The calculation of the percentage of the CPU time consumed by user (us), kernel (sy), waiting for I/O operation to complete (wa) or idle (id) is calculated relative to the consumed entitlement.

In Figure 6-4 on page 268 we assume that the bar shows the CPU use by the user space (us) and the rest is idle (id). In order to simplify the example, we assume that the CPU use by kernel (sy) or waiting for I/O (wa) is zero. The cpu0 is running 0.10 processor fraction in user space that is 50% of the used processor fraction by this processor. The remaining part is 0.10 processor fraction, that is 50% of the processor fraction used by this processor. The cpu1 is using 0.30 of 0.40 processor fraction that is 75%.

The per CPU sy, wa, and id statistics are calculated in the same way as the us statistics, just using the kernel space, wait I/O, or idle values.

Note: While the per CPU us, sy, wa, and id statistics are calculated as percentages of the fraction of the physical CPU consumed by a particular logical CPU, the all CPU us, sy, wa, and id statistics are calculated as percentages of the allocated entitlement.

The wa and id statistics for the U line are related to the allocated entitlement. The us and sy statistics are not applicable for the line representing unused entitlement.

The statistics for all CPUs used by the user space is the processor fraction used for user space of all processors, that is 0.40. This fraction is than related to the total entitlement in percents that is the value of the us: us=26.7%=100*0.40/1.50.

The statistics for all CPUs are calculated as the sum of the idle processor fractions plus the unused entitlement related to the total entitlement. The cpu0 spent 0.1 processor fraction and cpu1 spent also 0.1 processor fraction within the used entitlement, but running the wait kernel process. Both processors were using 0.2 processor fractions in idle state from the kernel point of view. But the

total time the processors were idle is more when you add in the unused entitlement of 0.90. So the total idle time was 0.10+0.10+0.90=1.10. Calculating a percentage related to the total entitlement of 1.50 results in id(all)=73.3%=100*1.10/1.50.

The idle statistics for the U line (representing unused entitlement) is calculated as the percentage of the unused entitlement which was unused because the processor was idle. In our case all the unused processor entitlement was because the processor was idle and so id(u)=100.0%.

If there were some unused entitlement because of I/O wait, then the wa(u) would be non zero.

Table 6-1 shows a sample calculation of the statistics with easy to calculate numbers. In the pc and %ec columns, the sum of the CPUs is reflected in the ALL row.

Table 6-1 Sample CPU usage calculation

CPU	us	sy	wa	id	pc	%ec
0	50.0	0.0	0.0	50.0	0.20	13.3
1	75.0	0.0	0.0	25.0	0.40	26.7
U	n/a	n/a	0.0	100.0	0.90	60.0
ALL	26,7	0.0	0.0	73.3	0.60	40.0

Example 6-5 shows how the processor usage and the entitlement goes together on a real system. The example was created by running the **mpstat** command on a slightly loaded system.

Example 6-5 mpstat basic statistics

```
# mpstat 1 2

System configuration: lcpu=2 ent=0.3

cpu  min  maj  mpc  int   cs  ics   rq  mig lpa sysc us sy wa id   pc  %ec  lcs
  0    0    0    0   31  148   74    1    1 100  204 68 22  0 10 0.01  3.5  197
  1    0    0    0  177    0    0    0    0   -    0  0 11  0 89 0.00  1.2  150
  U    -    -    -    -    -    -    -    -   -    -  -  -  0 95 0.29 95.3    -
ALL    0    0    0  208  148   74    1    1 100  204  2  1  0 97 0.01  4.7  173
------------------------------------------------------------------------------
  0    0    0    0   32  151   75    1    0 100  173 66 24  0 10 0.01  3.2  194
  1    0    0    0  172    0    0    0    0   -    0  0 12  0 88 0.00  1.1  144
  U    -    -    -    -    -    -    -    -   -    -  -  -  0 96 0.29 95.7    -
ALL    0    0    0  204  151   75    1    0 100  173  2  1  0 97 0.01  4.3  169
```

This example shows the basic statistics obtained from the **mpstat** command. The first line tells us the number of logical processors available to the operating system and the total processor entitlement assigned to this partition using Micro-Partitioning technology.

The *cpu* column represent the ID of the logical processor. The U line stands for the unused entitlement. The line U shows up only for POWER5 systems and only if part of the entitled capacity is not used by this partition. The last line “ALL” stands for all processors’ statistics.

Note: Many statistics for the U line statistics are not applicable. For example, an unused time slice cannot generate a context switch.

The us, sy, wa, id columns stand for the processor usage statistics in user mode, kernel mode, waiting for I/O completion or idle. Their meaning and calculation is explained earlier in this section.

The pc stands for the fraction of processor consumed. Its meaning and calculation is explained earlier in this section.

The %ec stands for the percentage of the consumed entitled capacity. Its meaning and calculation is explained earlier in this section.

The min stands for the minor page faults. Minor page faults do not involve disk I/O activity.

The maj stands for the major page faults. Major page faults involve disk I/O activity.

The mpc stands for Multi Processor Communication interrupts and is the sum of the mpcs and mpcr as described in “Interrupt statistics” on page 272.

The int stands for the sum of all interrupts as described in “Interrupt statistics” on page 272.

The cs stands for the context switches and the ics stands for involuntary context switches. The cs is the sum of voluntary context switches and involuntary context switches. The context switches are described in “Affinity and migration statistics” on page 273.

The rq stands for the run queue that is maintained per processor. The value for all CPUs is the sum of each processor’s run queue. This is also described in “Affinity and migration statistics” on page 273.

The mig stands for the total number of thread migrations to another processor. This is the sum of the migrations in affinity domains S1 to S5, that is,

mig=S1rd+S2rd+S3rd+S4rd+S5rd. Affinity domains are described in “Affinity and migration statistics” on page 273.

The lpa stands for the logical processor affinity. It is the percentage of logical processor redispatches within the scheduling affinity domain 3. Affinity domains are described in section “Affinity and migration statistics” on page 273.

The sysc stands for the number of system calls per each processor and the sum for all processors.

The lcs stands for the number of logical processor (hardware) context switches. This is the sum of involuntary and voluntary context switches as described in “Affinity and migration statistics” on page 273.

The min, maj, mpc, int, cs, rq, mig, lpa, sysc, and lcs statistics are per second within the interval.

Interrupt statistics

The interrupt-oriented statistics are obtained by the **mpstat -i** command.

Example 6-6 mpstat -i output

```
# mpstat -i 1 1

System configuration: lcpu=4 ent=0.5

cpu   mpcs   mpcr    dev   soft    dec     ph
  0      0      0      2   2160     25     14
  1      0      0      1      1    100    126
  2      0      0      2   2342    114      8
  3      0      0      3     74     10      6
ALL      0      0      8   4577    249    154
```

The lcpu, ent, and cpu columns are the same as in all **mpstat** command outputs. The mpcs, mpcr, dev, soft, dec, and ph columns are the interrupt statistics per second during the sample interval if specified, or the total since boot if the interval is not specified.

The mpcs and mpcr columns show the multiprocessor communication send and receive interrupts. The MPC interrupts are forwarded among processors in similar manner as for the interprocess communications (IPCS), but on the processor level. For example, whenever a TLB line is invalidated, the processor that invalidates the line sends MPC interrupts to other processors using that line. Their sum is displayed in the mpc column of the basic statistics.

The dev column stands for the interrupts initiated by a device. Devices generate these interrupts, for example, when the buffer is ready for the kernel.

The soft column stands for the software interrupts. These interrupts are generated whenever a device driver needs more time to finish processing a device interrupt.

The dec column stands for the number of decremeter interrupts. These interrupts are generated from timer counters.

The ph column stands for the number of phantom interrupts. The best way to describe these interrupts is with an example. On a system running two partitions, SPLPAR1 and SPLPAR2, a device originates an interrupt to processor belonging to SPLPAR1. In the time when the interrupt is initiated, the SPLPAR1 is outside its entitled capacity and SPLPAR2 is running on the processor. SPLPAR2 gets the interrupt, detects that it is not for it, counts it as a phantom interrupt, and calls the hypervisor to forward it to SPLPAR1. When the SPLPAR1 gets the processor again, the interrupt is forwarded to the SPLPAR1. Phantom interrupts show that other partitions are doing I/O operations.

Affinity and migration statistics

The operating system kernel dispatches the processes with an attempt to optimize performance. The process performance is dependent on which processor is dispatched to. If the process is dispatched to the same processor as it was dispatched to in the previous time slice, the performance is much better than if it were dispatched to a different MCM module due to cache and addressing requirements. There is a need to measure the dispatching of the processes and evaluate the efficiency of the dispatches.

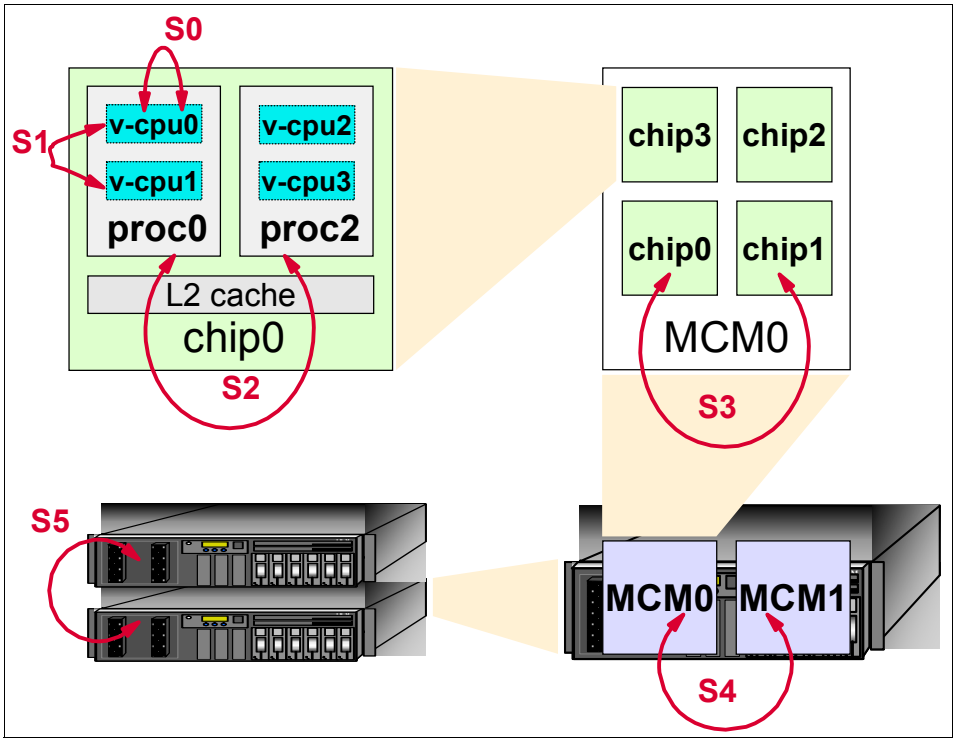

Figure 6-5 Affinity domains and process dispatches

Figure 6-5 shows the logic of the affinity domains. The IBM @server p5 systems with SMT turned on may feature the following affinity domains:

S0 The process redispatch occurs within the same logical processor.

S1 The process redispatch occurs within the same physical processor, among different logical processors. This can happen in the case of SMT-enabled systems. This involves sharing of the L1 cache.

S2 The process redispatch occurs within the same processor chip, but among different physical processors. This involves sharing of the L2 cache.

S3 The process redispatch occurs within the same MCM module, but among different processor chips. This involves sharing of the L3 cache.

S4 The process redispatch occurs within the same CEC plane, but among different MCM modules. This involves access to the main memory or L3-to-L3 transfer.

S5 The process redispatch occurs outside of the CEC plane. This domain is not currently used.

The **mpstat -d** command gives detailed dispatcher statistics. The statistics focus on process dispatches and the affinity of them related to each logical processor as well as averaged to global view.

```
# mpstat -d 1 1

System configuration: lcpu=4 ent=0.5

cpu     cs    ics  bound     rq   push S3pull  S3grd  S0rd  S1rd  S2rd  S3rd  S4rd  S5rd   ilcs   vlcs
  0      0      0      1      1      0      0      0     -     -     -     -     -     -    100      0
  1    156     70      1      1      0      0      0 100.0   0.0   0.0   0.0   0.0   0.0      1    218
  2      0      0      1      1      0      0      0     -     -     -     -     -     -    100      0
  3      1      1      1      1      0      0      0 100.0   0.0   0.0   0.0   0.0   0.0    100      0
ALL    157     71      4      4      0      0      0 100.0   0.0   0.0   0.0   0.0   0.0    150    109
# mpstat -d 1 1

System configuration: lcpu=4 ent=0.5

cpu     cs    ics  bound     rq   push S3pull  S3grd  S0rd  S1rd  S2rd  S3rd  S4rd  S5rd   ilcs   vlcs
  0    116    105      1      3      0      0      0  99.1   0.0   0.0   0.9   0.0   0.0    100      0
  1   1612    137      1      2      0      0      0  99.4   0.6   0.0   0.0   0.0   0.0    100      0
  2    250     78      1      2      0      0      0  99.2   0.4   0.0   0.4   0.0   0.0    100      0
  3   1094     80      1      3      0      0      0  99.5   0.5   0.0   0.0   0.0   0.0    100      0
ALL   3072    400      4     10      0      0      0  99.4   0.6   0.0   0.1   0.0   0.0    200      0
#
#
#
#
```

Figure 6-6 mpstat -d command output

Figure 6-6 shows two samples taken by the **mpstat -d** command. The first sample was taken when the system was slightly loaded with two processes running in cycle. The second example was taken when the system was loaded with more processes.

In our examples we have two physical processors of an p5-520 server configured to run SMT. SMT architecture creates four logical processors above the two physical processors. The *cpu* column stands for the logical CPU ID and does not change until a DLPAR reconfiguration takes place. The ALL stands for the sum of processor statistics.

The context switch statistics, shown in the cs column, are counted per each processor and the ALL processor cs value is the sum of each cs line.

The ics column stands for the involuntary context switches. These context switches are due to the exhaustion of available time slices.

The bound column stands for the total threads bound to the processor. This happens if the user runs the bindprocessor command or when the program itself binds threads to a processor. When there are processor bound threads, the push column may show non zero values.

The rq column stands for the process run queue. The ALL processor rq value is the sum of all rq lines.

The push column stands for the number of thread migrations due to starvation load balancing. This happens when a thread cannot get processor time for a long period of time. The scheduler determines if there is a better processor to run the process, then it pushes the thread to that processor. If the thread is pushed outside the affinity domain S3 (outside MCM) then this counter is incremented.

The S3pull column stands for the number of process migrations outside the affinity domain S3 due to idle stealing. This happens when one particular CEC board (S3 affinity domain) is underloaded and it is stealing threads from CPUs from other CEC boards.

The S3grd column stands for the number of dispatches from the global run queue, outside the affinity domain S3. This happens on multiple-CEC systems, like the p5-570, when a process is dispatched to a different CEC board. Such dispatches may occur when the loads to the CEC boards are significantly unbalanced.

The S0rd, S1rd, S2rd, S3rd, S4rd and S5rd stand for the thread redispatches within the affinity domains S0 (same logical cpu), S1 (same physical processor), S2 (same processor chip), S3 (same MCM), S4 (same CEC plane) and S5 (outside CEC plane). The values for ALL are averaged per each column.

When the processor was slightly loaded, all the thread redispatches were dispatched back to the same logical CPU and there were no available threads on the run queue that could be dispatched to cpu0 or cpu2. In this case cs=0 for the logical cpu, and the SXrd values show only a dash.

When the processor is loaded by more processes (or threads), the threads cannot stay on the same logical CPU and sometimes they need to be redispatched to different logical CPUs. This will cause the S0rd values to drop from 100% to 99% or even more, and the S1rd values will start to grow as the threads are redispatched to the closest available logical CPU. The S1rd column is zero in our example, because we are using single chip modules of the p5-520 system and there is just one chip per processor module (SCM). When we dispatch threads to the other physical processor, it is dispatched to the second SCM module. These thread redispatches appear in the S3rd column.

The ilcs and vlcs columns show, respectively, involuntary and voluntary logical context switches. These are hardware context switches due to partitioning using Micro-Partitioning technology.

SMT utilization

The addition of Simultaneous Multi Threading (SMT) requires statistics to be reported on the use of the particular logical processors. Example 6-7 on page 277 shows a sample of how to use the command and explanation follows.

Example 6-7 mpstat -s command output

```
# mpstat -s 1

System configuration: lcpu=4 ent=0.5

     Proc1           Proc0
      0.27%          49.63%
 cpu0    cpu2    cpu1    cpu3
  0.17%   0.10%   3.14%  46.49%
```

The **mpstat -s 1 2** command is run to get SMT statistics in 1 second samples. First, the output gives the system information about the number of logical processors and the entitlement assigned to the partition, as with any other flag. The statistics are then organized into as many columns as there are logical processors assigned to the partition.

The meanings of the columns are best described by an example: The total assigned entitlement is 0.5. This entitlement is distributed across the physical processors as 0.27% + 49.63% = 49.90%, or approximately 0.5 of the total entitlement in the system configuration line of the output. Each physical processor's consumed entitlement is then redistributed again to logical processors, for example Proc0 = cpu1 + cpu3, that is 3.14% + 46.49% = 49.63%.

6.6.5 lparstat command

The **lparstat** command is new in AIX 5L Version 5.3. It reports LPAR-related information and statistics.

The optional command flags for the **lparstat** command are:

-i Lists details on the LPAR configuration

-h Adds summary hypervisor statistics to the default **lparstat** command output

-H Provides detailed hypervisor information, including statistics for each of the hypervisor calls

The **lparstat** command with no options will generate a single report containing utilization statistics related to the LPAR since boot time. The **lparstat** command using the -h flag will add hypervisor summary statistics to the default lparstat output. If an *interval* and *count* are specified, the output report will repeat every *interval* seconds for *count* iterations.

The following is an example of this command, collecting statistics for one five-second interval, with the hypervisor summary data highlighted.

```
# lparstat -h 5 1
System configuration: type=Shared mode=Uncapped smt=On lcpu=4 mem=512 ent=0.50
%user  %sys  %wait  %idle physc %entc  lbusy   app  vcsw phint  %hypv hcalls
-----  ----  -----  ----- ----- ----- ------   ---  ---- -----  ----- ------
  0.0   0.5    0.0   99.4  0.00   1.0    0.0     -  1524     0    0.5   1542
```

Using the -H flag displays detailed hypervisor information, including statistics for many hcall functions. The command shows the following for each of these hcalls:

Number of calls	Number of hypervisor calls made
Total Time Spent	Percentage of total time spent in this type of call
Hypervisor Time Spent	Percentage of hypervisor time spent in this type of call
Average Call Time	Average call time for this type of call in nanoseconds
Maximum Call Time	Maximum call time for this type of call in nanoseconds

```
# lparstat -H 5 1
System configuration: type=Shared mode=Uncapped smt=On lcpu=4 mem=512 ent=0.50

           Detailed information on Hypervisor Calls

Hypervisor        Number of    %Total Time   %Hypervisor   Avg Call    Max Call
  Call              Calls         Spent      Time Spent    Time(ns)    Time(ns)

remove                   2            0.0           0.0         417         550
read                    21            0.0           0.0         148         294
nclear_mod               0            0.0           0.0           1           0
page_init                3            0.0           0.0         774        2280
clear_ref                0            0.0           0.0           1           0
protect                  0            0.0           0.0           1           0
put_tce                 14            0.0           0.1         544         908
xirr                    10            0.0           0.0         536         758
eoi                     10            0.0           0.0         526         695
ipi                      0            0.0           0.0           1           0
cppr                     0            0.0           0.0           1           0
asr                      0            0.0           0.0           1           0
others                   0            0.0           0.0           1           0
enter                    5            0.0           0.0         397         685
cede                  1595            0.5          99.6        7918     1343207
migrate_dma              0            0.0           0.0           1           0
put_rtce                 0            0.0           0.0           1           0
confer                   0            0.0           0.0           1           0
prod                    27            0.0           0.1         672         922
get_ppp                  1            0.0           0.0        1502        2579
set_ppp                  0            0.0           0.0           1           0
```

```
purr                    0           0.0           0.0           1           0
pic                     1           0.0           0.0         309         410
bulk_remove             0           0.0           0.0           1           0
send_crq                0           0.0           0.0           1           0
copy_rdma               0           0.0           0.0           1           0
get_tce                 0           0.0           0.0           1           0
send_logical_lan        0           0.0           0.0           1           0
add_logicl_lan_buf      0           0.0           0.0           1           0
```

To show the actual status of the partition you can use the **lparstat -i** command in the AIX command line interface of the partition:

```
# lparstat -i
Node Name                                  : applsrv
Partition Name                             : Apps_Server
Partition Number                           : 4
Type                                       : Shared-SMT
Mode                                       : Uncapped
Entitled Capacity                          : 0.30
Partition Group-ID                         : 32772
Shared Pool ID                             : 0
Online Virtual CPUs                        : 2
Maximum Virtual CPUs                       : 10
Minimum Virtual CPUs                       : 1
Online Memory                              : 512 MB
Maximum Memory                             : 1024 MB
Minimum Memory                             : 128 MB
Variable Capacity Weight                   : 128
Minimum Capacity                           : 0.20
Maximum Capacity                           : 1.00
Capacity Increment                         : 0.01
Maximum Physical CPUs in system            : 2
Active Physical CPUs in system             : 2
Active CPUs in Pool                        : -
Unallocated Capacity                       : 0.00
Physical CPU Percentage                    : 15.00%
```

To verify the dynamic action use the **lparstat -i** command on the selected partition again. We changed the partition mode from uncapped to capped for illustrative purposes.

```
# lparstat -i
Node Name                                  : applsrv
Partition Name                             : Apps_Server
Partition Number                           : 4
Type                                       : Shared-SMT
Mode                                       : Capped
```

```
Entitled Capacity                          : 0.30
Partition Group-ID                         : 32772
Shared Pool ID                             : 0
Online Virtual CPUs                        : 2
Maximum Virtual CPUs                       : 10
Minimum Virtual CPUs                       : 1
Online Memory                              : 512 MB
Maximum Memory                             : 1024 MB
Minimum Memory                             : 128 MB
Variable Capacity Weight                   : 128
Minimum Capacity                           : 0.20
Maximum Capacity                           : 1.00
Capacity Increment                         : 0.01
Maximum Physical CPUs in system            : 2
Active Physical CPUs in system             : 2
Active CPUs in Pool                        : -
Unallocated Capacity                       : 0.00
Physical CPU Percentage                    : 15.00%
Unallocated Weight                         : 0
```

6.6.6 sar command enhancements

Beginning with AIX 5L Version 5.3, the **sar** command reports utilization metrics *physc* and *%entc*, which are related to Micro-Partitioning and simultaneous multi-threading environments. These metrics are only be displayed on Micro-Partitioning and simultaneous multi-threading environments. The physc metric indicates the number of physical processors consumed by the partition (in case of system-wide utilization) or logical CPU (if the -P flag is specified) and %entc indicates the percentage of the allocated entitled capacity (in case of system-wide utilization) or consumed entitled capacity (if the -P flag is specified). When the partition runs in capped mode, the partition cannot get more capacity than it is allocated. In uncapped mode, the partition can get more capacity than is actually allocated to it. This is called consumed entitled capacity. If the -P flag is specified and there is unused capacity, the **sar** command prints the unused capacity as a separate CPU with cpu id U similar to what the **mpstat** command does.

Highlights of command options that have changed in AIX 5L V5.3 are the following:

-u Reports per processor or system-wide statistics. When used with the -P flag, the information is provided for each specified processor; otherwise, it is provided only system-wide. Because the -u flag information is expressed as percentages, the system-wide information is simply the average of each individual processor's statistics except when using a shared pool of processors using Micro-Partitioning

technology, where the percentage are relative to the entitled capacity. The following values are displayed:

%idle Percentage of time the cpu or cpus were idle with no outstanding disk I/O requests.

%sys Percentage of time the cpu or cpus spent in execution at the system (or kernel) level.

%usr Percentage of time the cpu or cpus spent in execution at the user (or application) level.

%wio Percentage of time the cpus were idle during which the system had outstanding disk/NFS I/O requests.

physc Number of physical processors consumed. This is reported only if the partition is running with shared processors or simultaneous multi-threading enabled (-P only).

%entc Percentage of entitled capacity consumed (physc/ent)*100. This is reported only if the partition is running with shared processors or simultaneous multi-threading enabled (-P only).

Note: The **sar** command reports system unit activity if no other specific content options are requested. If the -P flag is used and the partition is running with shared processors, and if the partition capacity usage is less than what is allocated, then a CPU row with cpuid U will be reported to show the system-wide unused capacity. If the partition is running with shared processors in uncapped mode, then %entc will report the percentage of consumed entitled capacity against each CPU row and percentage of allocated entitled capacity in the system-wide CPU row.

6.6.7 topas command enhancements

Figure 6-7 on page 282 displays a sample default output of the **topas** command. If **topas** runs on a partition with shared processor allocation (Micro-Partitioning), beneath the CPU utilization there are two new values displayed:

- Physc
  Number of physical processors utilized by the partition (if using Micro-Partitioning technology).
- %Entc
  Percentage of Entitled Capacity utilized by a partition (if using Micro-Partitioning technology).

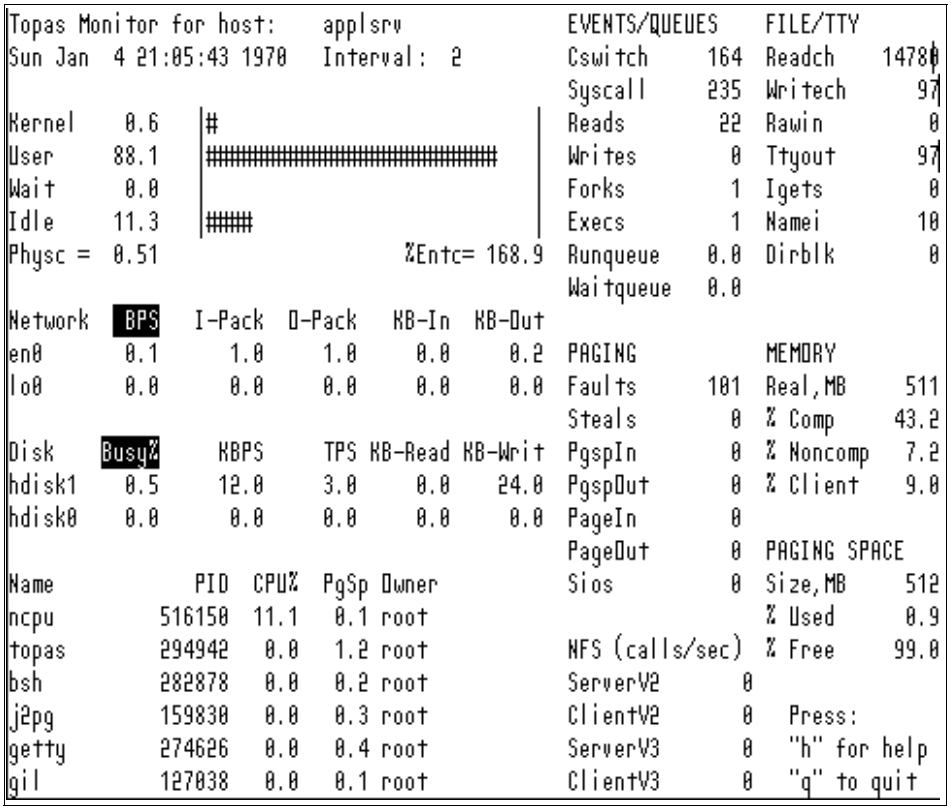

Figure 6-7 topas default sample output

Figure 6-8 on page 283 shows the output displayed when using the new -L flag of the **topas** command. The -L flag switches the output to logical partition display. You can either use -L when invoking the **topas** command, or as a toggle during running of topas. In this mode topas displays data similar to the **mpstat** and **lparstat** commands.

```
Interval:    2    Logical Partition: Apps_Server       Sun Jan  4 21:06:17 1970
Psize:       -               Shared SMT  ON           Online Memory:   512.0
Ent: 0.30                     Mode: Capped            Online Logical CPUs:  4
Partition CPU Utilization                             Online Virtual CPUs:  2
%usr %sys %wait %idle physc %entc %lbusy   app   vcsw phint %hypv   hcalls
  87    1     0    12   0.5 171.55 13.92     -   1199    12   0.0        0

LCPU  minpf majpf  intr   csw icsw runq lpa scalls usr sys _wt idl   pc   lcsw
Cpu0     0     0   362   492  204    1 100   1787  41  49   0  10 0.01    363
Cpu1     0     0    39     0    0    0   0      0   0   7   0  93 0.00    348
Cpu2     0     0   253    17    6    1  99      2 100   0   0   0 0.45    234
Cpu3     0     0    44     0    0    0   0      0   0   0   0 100 0.06    254
```

Figure 6-8 topas logical partition sample output

6.7 Per file system I/O pacing

I/O pacing is used to prevent a large number of I/O page outputs for one file from monopolizing the I/O bus. There are two tunable parameters you can set up:

minpout Minimum number of outstanding I/O pages.

maxpout Maximum number of outstanding I/O pages.

When the number of outstanding output pages for a particular file reaches the maxpout value, any other threads waiting to do additional pageouts on that segment will sleep until the number of outstanding output pages falls to the minpage value for that file.

In previous versions of AIX this behavior was tunable only system wide using the **smit chgsys** fastpath or the **chdev -l sys0 -a maxpout=24 -a minpout=32** command. There were cases when some file systems, like database file systems, required different values then other file systems, like temporary file systems.

In Version 5.3 the I/O pacing can be tuned on a per file system basis. This tuning is done when using the **mount** command:

```
mount -o minpout=40,maxpout=60 /fs
```

A more user friendly way to do this is to use SMIT or to edit the /etc/filesystems. The highlighted mount options of the **smit crjfs2lvstd** menu in Figure 6-9 show how to set the minpout and maxpout parameters on a per file system basis.

```
                    Add an Enhanced Journaled File System

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

                                                      [Entry Fields]
* LOGICAL VOLUME name                                 fslv                +
* MOUNT POINT                                        [/fs]
  Mount AUTOMATICALLY at system restart?              yes                 +
  PERMISSIONS                                         read/write          +
  Mount OPTIONS                                      [minpout=40,maxpout=60] +
  Block Size (bytes)                                  4096                +
  Logical Volume for Log                                                  +
  Inline Log size (MBytes)                           []                    #
  Extended Attribute Format                           Version 2           +
  ENABLE Quota Management?                            no                  +

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

Figure 6-9 smit crjfs2lvstd

You can check that the file system is mounted correctly by using the mount command as follows:

```
s4 # mount
  node       mounted        mounted over    vfs       date        options
-------- ---------------  ---------------  ------ ------------ ---------------
         /dev/hd4         /                jfs2   Jul 09 18:19 rw,log=/dev/hd8
         /dev/hd2         /usr             jfs2   Jul 09 18:19 rw,log=/dev/hd8
         /dev/hd9var      /var             jfs2   Jul 09 18:19 rw,log=/dev/hd8
         /dev/hd3         /tmp             jfs2   Jul 09 18:19 rw,log=/dev/hd8
         /dev/hd1         /home            jfs2   Jul 09 18:20 rw,log=/dev/hd8
         /proc            /proc            procfs Jul 09 18:20 rw
         /dev/hd10opt     /opt             jfs2   Jul 09 18:20 rw,log=/dev/hd8
         /dev/fslv00      /fs              jfs2   Jul 15 11:06
rw,maxpout=60,minpout=40,log=/dev/loglv01
```

Advanced support can check the file system in the kernel using the **kdb** kernel debugger. First, with the **pdt *** sub-command get the file system slot ID from the list of mounted file systems, then get the entries related to this slot ID. The following is an example of this:

```
# kdb
...
(0)> pdt *
              SLOT   NEXTIO           DEVICE  DMSRVAL    IOCNT <name>
vmmdseg+90A0000 0000 FFFFFFFF 8000000A00000002 00000000 00000000 paging
vmmdseg+90A5400 0080 FFFFFFFF 01460CA8 00000000 00000000 remote
vmmdseg+90A54A8 0081 FFFFFFFF 0408D100 00000000 00000000 remote
vmmdseg+90A5550 0082 FFFFFFFF 8000000A00000007 00003022 00000000 local client
vmmdseg+90A55F8 0083 FFFFFFFF 8000000A00000004 00000000 00000000 local client
vmmdseg+90A56A0 0084 FFFFFFFF 8000000A00000005 00000000 00000000 local client
vmmdseg+90A5748 0085 FFFFFFFF 8000000A00000006 00000000 00000000 local client
vmmdseg+90A57F0 0086 FFFFFFFF 0408D040 00000000 00000000 remote
vmmdseg+90A5898 0087 FFFFFFFF 0408B708 00000000 00000000 remote
vmmdseg+90A5940 0088 FFFFFFFF 8000000A00000008 00000000 00000000 local client
vmmdseg+90A59E8 0089 FFFFFFFF 8000000A00000009 00000000 00000000 local client
vmmdseg+90A5BE0 008C FFFFFFFF 040ECC18 00000000 00000000 remote
vmmdseg+90A5DD8 008F FFFFFFFF 8000002D00000008 00000000 00000000 local client
(0)> pdt 8f

PDT address F1000100090A5DD8 entry 008F of 1FFF, type: LOCAL CLIENT -- XPAGER
next pdt on i/o list  (nextio)  : FFFFFFFF
dev_t or strategy ptr (device)  : 8000002D00000008
last frame w/pend I/O (iotail)  : FFFFFFFFFFFFFFFF
free buf_struct list  (bufstr)  : F10006000D862E90
total buf structs     (nbufs)   : 0000
available (PAGING)    (avail)   : 0000
disk map srval        (dmsrval) : 00000000
i/o's not finished    (iocnt)   : 00000000
device wait list (devwait)      : 0000000000000000
buffer wait list (bufwait)      : 0000000000000000
logical volume lock (lock)      :@F1000100090A5E30 00000000
buffer list lock    (buf_lock)  :@F1000100090A5E38 00000000
flag bits            (devflags) : 84000000
max phys Xlation ent (maxphys)  : 00000020
pageout down limit   (minpout)  : 00000028
pageout up limit     (maxpout)  : 0000003C
external pager  (pager)  : 0000000001458158
(0)>
```

7

Networking

AIX 5L provides many enhancements in the networking area. Described in this chapter, they include:

- NFS Version 4
- Multipath routing enhancements
- PMTU enhancements
- DHCPv6 support
- IPv6 functional updates
- Interface layer support for hotplug network adapters
- Support for SLP client service
- Stream Control Transmission Protocol (SCTP)
- Network RAS enhancements
- Enable IP Security with intervening NAT devices
- AIX SNMP subagent enhancements
- SMBFS enhancements
- Ported 64-bit DES NFS kernel extension
- BIND domain search improvement
- Network API support for ARP

7.1 NFS Version 4

The Network File System (NFS) is a distributed file system that allows users to access files and directories located on remote computers and treat those files and directories as if they were local. Prior to Version 5.3, AIX supports the NFS Version 2 and 3 protocols. Starting with Version 5.3, AIX supports NFS Version 4 as well as NFS Version 2 and 3. NFS Version 4 includes the following features:

- More focus on security
  - RPCSEC-GSS RPC
  - Support for foreign domain access using identity mapping (requires deployment of LDAP and Enterprise Identity Mapper)
- RPC operations for file locking move into the main NFS protocol
  - No rpc.lockd, rpc.statd and rpc.mountd in NFS V4
  - Friendlier to firewalls
- NFS V4 only supports TCP
  - No support for UDP
- NFS V4 ACL support in JFS2
  - Fine-grained access control for file system objects
  - Support inheritance features

You can configure NFS V4 through SMIT, command line, and Web-based System Manager.

RPCSEC-GSS RPC authentication

NFS V4 can be configured with RPCSEC_GSS to provide stronger security for the protocol. While not mandated, other more traditional methods such as AUTH_SYS are also be provided. RPCSEC_GSS is based on the function of GSS-API. This allows for support of multiple security mechanisms without the requirement of adding new RPC authentication methods.

In addition to the authentication aspects of these mechanisms, each offers selectable levels of protection such as integrity or privacy on the message data. The RPCSEC_GSS service attribute specifies the level of message protection and is mandatory for RFC compliance. The Version 4 protocol provides support for the NFS client and server to negotiate both the mechanisms and the service levels.

File locking and open share reservations

NFS V4 brings significant changes for file locking when compared to earlier NFS versions. The RPC operations for file locking move into the main NFS protocol. The separate Network Lock Manager (NLM) and status monitor protocols in earlier NFS versions are eliminated, along with the corresponding rpc.lockd and rpc.statd daemons in NFS V4. Note that the rpc.statd and rpc.lockd are still used for NFS V2 and V3 for file locking. In addition, the connection between client and server is now stateful in NFS V4, while the previous versions of NFS were stateless.

NFS V4 ACL support

The NFS V4 protocol defines an ACL model and data types, and provides RPC operations for passing Access Control List (ACL) specifications between the server and client. The defined ACL format closely resembles the ACLs of the PC environment. The ability to provide for ACL query and specification as part of the NFS V4 protocol is a valuable feature. ACLs are an optional feature of the protocol. They are communicated in the recommended portion of the NFS V4 file.

- Server support for NFS V4 ACL

  When an AIX is an NFS V4 server, it can service a variety of clients. A client can be an AIX or third party NFS. One of the NFS V4 server's responsibilities is to service NFS V4 ACL requests. These requests are setting and getting ACLs of the underlying file system. These requests can only be serviced when the underlying file system supports NFS V4 ACL. JFS2 is enhanced to support NFS V4 ACL in Version 5.3.

Note: Refer to 3.2.6, "JFS2 ACL support for NFS V4" on page 131 for further details.

- Client support for NFS V4 ACL

  When an AIX is an NFS V4 client, it is able to set and retrieve NFS V4 ACL from its NFS V4 server if the server supports the NFS V4 ACL.

- AIX ACL support outside of NFS V4 protocol

  AIX ACLs are supported in NFS V2 and NFS V3 when both the client and server are AIX. This service is not defined in the NFS protocol. When an NFS client file system is mounted, a handshake is attempted to establish the AIX ACL support communication channel. When both the client and server are AIX, the handshake is successful and a TCP port is set up to support the off-protocol AIX ACL communication. NFS V4 will continue this off-band support for AIX ACL between client and server.

- NFS V2 and NFS V3 ACL support

  The NFS V2 and NFS V3 code will not be changed to support any new ACL function.

Note: NFS clients use the NFS version 3 protocol by default.

The new daemon processes

The following new daemon processes, highlighted in Example 7-1, are introduced:

nfsrgyd The nfsrgyd daemon provides a name translation service for NFS servers and clients. This daemon must be running in order to perform translations between NFS string attributes and UNIX numeric identities. The nfsrgyd daemon should be up and running from servers and clients using NFS V4 or RPCSEC-GSS.

gssd Some NFS security methods, such as Kerberos 5, are provided under a more general mechanism called General Security Services, or GSS. In AIX, GSS services are provided by a library in the IBM Network Authentication Service (NAS) fileset. NAS is shipped on the expansion pack. If the gssd daemon is not running, then efforts to access files using NFS using GSS security methods such as Kerberos 5 will fail.

Example 7-1 The nfsrgyd and gssd daemons

```
# lssrc -g nfs
Subsystem         Group            PID          Status
 biod             nfs              258180       active
 nfsd             nfs              413858       active
 nfsrgyd          nfs              266276       active
 gssd             nfs              446648       active
 rpc.mountd       nfs                           inoperative
 rpc.lockd        nfs                           inoperative
 rpc.statd        nfs                           inoperative
```

New command utilities

The following are new command utilities:

chnfsdom The chnfsdom command changes the local NFS domain of the system. The local NFS domain is stored in the /etc/nfs/local_domain file.

nfs4cl Displays or modifies current NFSv4 statistics and properties.

nfshostkey An NFS server (or full client) using RPCSEC_GSS RPC security must be able to acquire credentials for its host principal to accept

requests. The **nfshostkey** command is used to configure this information; it takes the principal and the path to a keytab file for that principal. This information goes into /etc/nfs/hostkey.

chnfsim Changes NFS foreign identity mappings.

Setting up a network for RPCSEC-GSS

Figure 7-1 shows the setup of an NFS V4 server and client in which a network is configured for RPCSEC-GSS.

Note:

1. RPCSEC_GSS also can be configured with NFS V3 in AIX 5L Version 5.3.
2. Although NFS V4 can be configured without RPCSEC-GSS in Version 5.3, the following example uses RPCSEC-GSS for better security.

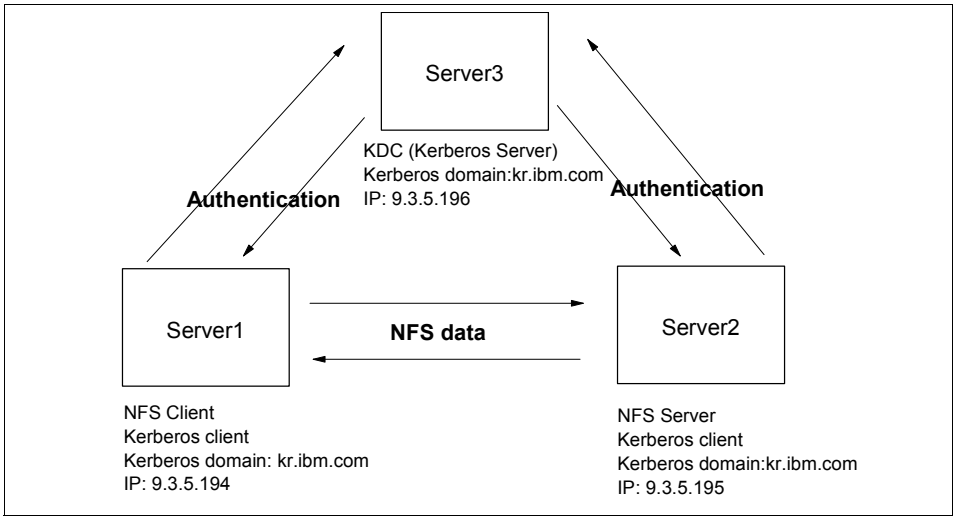

Figure 7-1 An example configuration

The system server3.kr.ibm.com® will be configured as the Key Distribution Center (KDC) server, the Kerberos realm KR.IBM.COM will be created, and server2.kr.ibm.com will be NFS server offering file systems exported with RPCSEC-GSS.

In addition, this network has the following users: **slson** and **myahn**.

Create the /etc/exports file through SMIT or using the **vi** command.

```
# vi /etc/exports
```

```
/slson -vers=4,sec=krb5:krb5i:krb5p,rw
```

Step1: Set up the KDC server

Note: Kerberos requires that the system time be reasonably close throughout the network. Before beginning this procedure, you should set up a mechanism to automatically synchronize time throughout the network, such as the AIX timed daemon or an NTP setup.

In this scenario, server3.kr.ibm.com will be configured as the KDC server (Example 7-2).

1. Install the krb5.server and modcrypt.base file set on server3. Install the krb5.client and modcrypt.base file set on server2 and server1. These file sets are available from AIX 5L Version 5.3 expansion pack.

   Run the following command:

   config.krb5 -S -d kr.ibm.com -r KR.IBM.COM

   After running this command, the system will ask for a Master Database password and a password for the administrative principal.

Example 7-2 Setting up KDC server

```
# lslpp -L |grep krb5
  krb5.client.rte          1.4.0.0    C     F    Network Authentication Service
  krb5.client.samples      1.4.0.0    C     F    Network Authentication Service
  krb5.server.rte          1.4.0.0    C     F    Network Authentication Service

# lslpp -L |grep modcrypt
  modcrypt.base.includes    5.3.0.0    C     F    Cryptographic Library Include
  modcrypt.base.lib         5.3.0.0    C     F    Cryptographic Library
                                                    (libmodcrypt.a)

# config.krb5 -S -d kr.ibm.com -r KR.IBM.COM
Initializing configuration...
Creating /etc/krb5/krb5_cfg_type...
Creating /etc/krb5/krb5.conf...
Creating /var/krb5/krb5kdc/kdc.conf...
Creating database files...
Initializing database '/var/krb5/krb5kdc/principal' for realm 'KR.IBM.COM'
master key name 'K/M@KR.IBM.COM'
You are prompted for the database Master Password.
It is important that you DO NOT FORGET this password.
Enter database Master Password:
Re-enter database Master Password to verify:
WARNING: no policy specified for admin/admin@KR.IBM.COM;
  defaulting to no policy. Note that policy may be overridden by
  ACL restrictions.
```

```
Enter password for principal "admin/admin@KR.IBM.COM":
Re-enter password for principal "admin/admin@KR.IBM.COM":
Principal "admin/admin@KR.IBM.COM" created.
Creating keytable...
Creating /var/krb5/krb5kdc/kadm5.acl...
Starting krb5kdc...
krb5kdc was started successfully.
Starting kadmind...
kadmind was started successfully.
The command completed successfully.
```

2. Create principals for each user and host by running the **/usr/krb5/sbin/kadmin.local** command on the KDC server. This example creates Kerberos principals (Example 7-3) that match the UNIX user name of the associated user. The principal name will be mapped to the user name by NFS to determine the UNIX credential associated with the principal. For this network, we created the following principals:

   – slson

   – myahn

   – nfs/FQDN (principals for hosts)

   The chosen user principal names must match the corresponding user names in the local registry (/etc/passwd, for example). NFS uses the principal name as a user name to obtain user and group IDs on the local system. If the names do not match, the access will be treated as an anonymous access.

Example 7-3 Add principal on KDC server

```
# /usr/krb5/sbin/kadmin.local
kadmin.local:  addprinc slson
WARNING: no policy specified for slson@KR.IBM.COM;
  defaulting to no policy. Note that policy may be overridden by
  ACL restrictions.
Enter password for principal "slson@KR.IBM.COM":
Re-enter password for principal "slson@KR.IBM.COM":
Principal "slson@KR.IBM.COM" created.
kadmin.local:  addprinc myahn
WARNING: no policy specified for myahn@KR.IBM.COM;
  defaulting to no policy. Note that policy may be overridden by
  ACL restrictions.
Enter password for principal "myahn@KR.IBM.COM":
Re-enter password for principal "myahn@KR.IBM.COM":
Principal "myahn@KR.IBM.COM" created.
kadmin.local:  addprinc nfs/server2.kr.ibm.com
WARNING: no policy specified for nfs/server2.kr.ibm.com@KR.IBM.COM;
  defaulting to no policy. Note that policy may be overridden by
  ACL restrictions.
```

```
Enter password for principal "nfs/server2.kr.ibm.com@KR.IBM.COM":
Re-enter password for principal "nfs/server2.kr.ibm.com@KR.IBM.COM":
Principal "nfs/server2.kr.ibm.com@KR.IBM.COM" created.
```

Step 2: Configure the NFS client and server

The NFS client and server will now be configured as Kerberos clients (Example 7-4) by using the config.krb5 command. How this is done will depend on how the KDC was configured. In this scenario we ran the following command on each NFS system:

```
#config.krb5 -C -d kr.ibm.com -r KR.IBM.COM -c server3.kr.ibm.com -s
server3.kr.ibm.com
```

It is now possible to run kinit as any of the user principals on any of the configured systems. For example, to kinit as user myahn, run the following command:

```
#/usr/krb5/bin/kinit myahn
```

You will need to specify myahn's Kerberos, not AIX, password. This example uses kinit to authenticate the user. It is possible to configure AIX to use Kerberos authentication during system login. For more information, see Authenticating to AIX Using Kerberos in AIX 5L Version 5.3 Security Guide.

Example 7-4 Setting up the Kerberos client

```
#Run the following commands from NFS server(server2) and NFS client(server1).

# config.krb5 -C -d kr.ibm.com -r KR.IBM.COM -c server3.kr.ibm.com -s server3
server3.kr.ibm.com
Initializing configuration...
Creating /etc/krb5/krb5.conf...
The command completed successfully.

# /usr/krb5/bin/kinit myahn
Password for myahn@KR.IBM.COM:
```

Step 3: Configure the NFS server keytab entry

The NFS server will now be configured with the appropriate keytab entry. In this scenario (Example 7-5), we configured the keytab entry for server2.kr.ibm.com.

1. From server2.kr.ibm.com, run the **kadmin** command. Then, run the following command:

   ```
   kadmin:ktadd nfs/server2.kr.ibm.com
   ```

   This creates a keytab file.

2. Set up the gssd daemon to use the keytab file you just created with the **nfshostkey** command. In this scenario, we ran the following:

   ```
   #nfshostkey -p nfs/server2.kr.ibm.com -f /etc/krb5/krb5.keytab
   ```

3. Set up the gssd daemon to start up automatically by running the following command:

   ```
   #chnfs -S -B
   ```

Example 7-5 Creating keytab file and setting up the gssd daemon for NFS Server

```
# /usr/krb5/sbin/kadmin -p admin/admin
Authenticating as principal admin/admin with password.
Password for admin/admin@KR.IBM.COM:
kadmin:  ktadd nfs/server2.kr.ibm.com
Entry for principal nfs/server2.kr.ibm.com with kvno 2, encryption type Triple
DES cbc mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal nfs/server2.kr.ibm.com with kvno 2, encryption type ArcFour
with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal nfs/server2.kr.ibm.com with kvno 2, encryption type AES-256
CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal nfs/server2.kr.ibm.com with kvno 2, encryption type DES cbc
mode with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.

# nfshostkey -p nfs/server2.kr.ibm.com -f /etc/krb5/krb5.keytab
# chnfs -S -B
0513-059 The gssd Subsystem has been started. Subsystem PID is 434406.
```

Step 4: Configure the NFS registry daemon

At this point, the NFS server will work, although all users will appear as nobody. Any users that do not exist will have access to the exported directory only as nobody. To get user names to map properly, you must configure the NFS registry daemon, as follows:

1. Set up the domain using the **chnfsdom** command.
2. Set up the /etc/nfs/realm.map file (Example 7-6); this file should contain one line, with the realm name followed by the local domain. The realm entry in this file is not case-sensitive, so technically, this entry is not required.

Example 7-6 Change nfs domain and set up map file

```
# chnfsdom kr.ibm.com

# vi /etc/nfs/realm.map
realm.map KR.IBM.COM kr.ibm.com
#refresh -s nfsrgyd
```

3. For server1.kr.ibm.com, which will be the NFS client, start up the gssd daemon using the **chnfs -S -B** command (Example 7-7). Before trying any Kerberos client operations, the user must use **kinit** to obtain valid credentials. You are now able to mount the NFS file system with the nfs4 and Kerberos option.

Example 7-7 Mount NFS4 file system with Kerberos on the NFS client

```
# chnfs -S -B

# mount -o sec=krb5,vers=4 server2.kr.ibm.com:/slson /mnt
# mount
  node       mounted        mounted over    vfs       date        options
-------- ---------------  ---------------  ------ ------------ ---------------
         /dev/hd4         /                jfs    Jul 09 11:03 rw,log=/dev/hd8
         /dev/hd2         /usr             jfs    Jul 09 11:03 rw,log=/dev/hd8
         /dev/hd9var      /var             jfs    Jul 09 11:03 rw,log=/dev/hd8
         /dev/hd3         /tmp             jfs    Jul 09 11:03 rw,log=/dev/hd8
         /dev/hd1         /home            jfs    Jul 09 11:05 rw,log=/dev/hd8
         /proc            /proc            procfs Jul 09 11:05 rw
         /dev/hd10opt     /opt             jfs    Jul 09 11:05 rw,log=/dev/hd8
         /dev/nimlv       /export          jfs    Jul 09 10:14
rw,log=/dev/loglv00
server2.kr.ibm.com /slson           /mnt             nfs4   Jul 13 10:41
sec=krb5,vers=4
```

If you are trying to mount the NFS file system from a machine in which Kerberos has not been set up properly, the attempt will fail with the following error messages (Example 7-8).

Example 7-8 Mount failure without Kerberos configuration

```
# mount server2.kr.ibm.com:/slson /mnt
mount: 1831-011 access denied for server2.kr.ibm.com:/slson
mount: 1831-008 giving up on:
server2.kr.ibm.com:/slson
The file access permissions do not allow the specified action.
```

7.2 Multipath routing enhancements

Starting with AIX 5L Version 5.1, you are allowed to use multiple routes to the same destination (including multiple default routes) through the Multipath Routing feature. The cost (hopcount) is used to determine the route to use when there are multiple routes to the same destination. A round-robin scheme is used to select a route when there are multiple routes with the same destination and

cost. The following feature is added to provide you with more flexibility for multipath routing in Version 5.3.

Configure multipath routing policy with SMIT

Configurable multipath routing methods are now included in the SMIT mkroute fastpath. Notice that there are six items you can configure for multipath routing as shown in Figure 7-2.

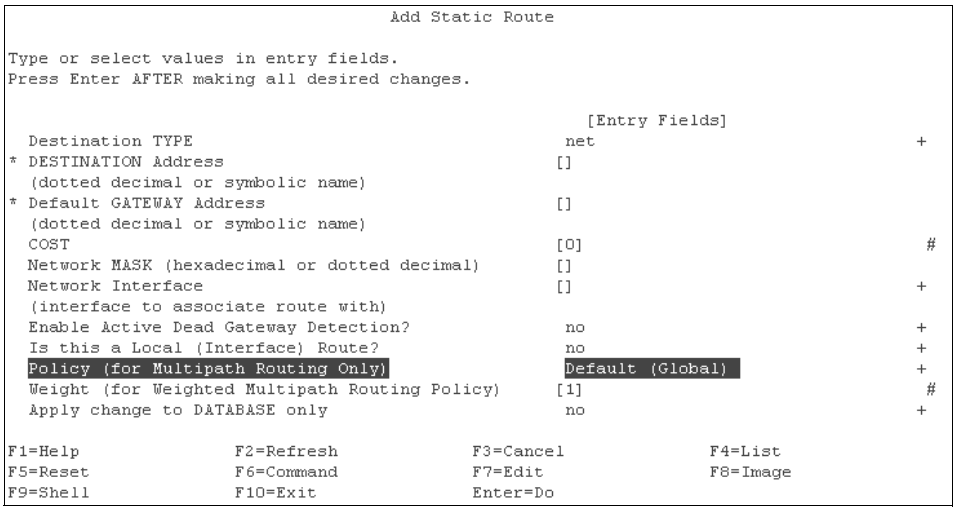

Figure 7-2 Add Static Route SMIT panel

If the policy is not explicitly set during the creation of a static route and multipath routing is used, then the global **no** command option mpr_policy determines the policy that will be used.

The default policy is Weighted Round-Robin, which behaves just like Round-Robin when the weights are all 1. If the default policy is chosen when the route is created, then the no option " mpr_policy" takes precedence and its value is used to determine the policy that will be used. On the other hand, if the policy is explicitly set to Weighted Round-Robin then this setting overrides the mpr_policy setting. Table 7-1 explains each routing method for multipath routing.

Table 7-1 Routing method for multipath routing

Policy for multipath routing	Policy number	Descriptions
Default(Global)	N/A	**no** command option named mpr_policy determines the policy that will be used.

Policy for multipath routing	Policy number	Descriptions
Weighted RR	1	User-configured weights (optional when routes are added) will be combined with round-robin.
Random	2	A route will be randomly picked among the multiple routes.
Weighted Random	3	A route will be randomly picked among the multiple routes but with weights.
Lowest Utilization	4	The route with the lowest utilization (number of active connections) will be picked.
Hash-Based	5	Hash based on Source/Dest IP address and TCP/UDP port number.

Weights

In the previous SMIT menu, shown in Figure 7-2 on page 297, you can find a new item of Weight (for Weighted Multipath Routing Policy). If there are weights assigned to multiple duplicate routes (same netmask and cost) and weighted round-robin (5) is chosen as the routing policy, then the system round-robins between these routes based on the weights. For example, consider the case where there are two routes to a specific destination and there are weights of 3 and 1 assigned to the first and the second route respectively. Now when weighted round-robin is chosen, then the first route is used three times in succession before the packets are routed using the second route.

Configure multipath routing policy through command line

In order to add routing policy and/or weights use the **route add** command:

```
route add .. -policy <routing policy> -weight <weight of the route>
```

Example 7-9 shows the command for adding a routing table to the destination of 9.3.5.194 and gateway 192.168.100.1 with a Weighted Round-Robin for policy and 3 for weight.

Example 7-9 Setting policy and weight for multipath routing

```
# route add 9.3.5.194 192.168.100.1 -policy 1 -weight 3
# netstat -rn
Routing tables
Destination      Gateway           Flags   Refs     Use  If   PMTU Exp Groups

Route Tree for Protocol Family 2 (Internet):
9.3.5.194        192.168.100.1     UGH        0        0  en2     -   -
127/8            127.0.0.1         U         12    69940  lo0     -   -
```

```
192.168.100.0    192.168.100.11    UHSb      0        0  en2     -   -  =>
192.168.100/24   192.168.100.11    U         2       97  en2     -   -
192.168.100.11   127.0.0.1         UGHS      0        0  lo0     -   -
192.168.100.255  192.168.100.11    UHSb      0        0  en2     -   -

Route Tree for Protocol Family 24 (Internet v6):
::1              ::1               UH        0       28  lo0     -   -
```

To look at the routing tables including the policy and weights use the **netstat -C** command.

Note: If the policy is not explicitly set and multipath routing is used, then the global **no** command option called mpr_policy determines the policy that will be used.

7.3 PMTU enhancements

The following sections discuss the Path Maximum Transmission Unit enhancements found in AIX 5L Version 5.3.

PMTU discovery in AIX 5L Version 5.3

The current Path Maximum Transmission Unit (PMTU) discovery implementation uses ICMP Echo Request and ICMP Echo Reply packets to discover PMTU using IPv4. This involves sending extra packets through the network. Some system administrators set up their firewall to drop ICMP Echo packets, resulting in the failure of this method of PMTU discovery. Current implementation clones a route to store the PMTU value in the routing table. This cloning mechanism does not work for multipath routing because the cloned route is always used instead of alternating between the two multipath network routes.

The new feature is implemented in Version 5.3 to have the Path MTU discovery mechanism use TCP packets and UDP datagrams rather than ICMP Echo packets. Also, the discovered MTU will not be stored in the routing table. This will enable multipath routing to work with PMTU discovery since no cloning will be necessary.

IPv6 never sends an ICMPv6 packet to detect the PMTU. The first packet of a connection always starts the process. Also, IPv6 routers must never fragment packets and must always return an ICMPv6 `Packet too big` message if they are unable to forward a packet because of a smaller outgoing MTU. Thus, for IPv6 the only change necessary is to make PMTU discovery work with Multipath routing.

Table 7-2 on page 300 lists the PMTU enhancements in Version 5.3.

Table 7-2 PMTU enhancements

	Version 5.2 or earlier	**Version 5.3**	**Improvements**
Method of PMTU discovery (IP V4)	Extra ICMP echo request and ICMP echo reply	No additional packets (Just uses TCP/UDP itself)	Network performance
PMTU data (IP V4 and IP V6)	Part of routing table	Separate PMTU table	No more cloning data in routing table so that we can use multipath routing with PMTU
Command to find PMTU data	**`netstat -rn`**	**`pmtu display`**	Take note!

Command to display PMTU table

Beginning with Version 5.3, when the Path MTU discovery is attempted for a destination, a pmtu entry gets created in a Path MTU (PMTU) table. This table can be displayed using the **`pmtu display`** command. Accumulation of pmtu entries can be avoided by allowing unused pmtu entries to expire and be deleted. PMTU entry expiration is controlled by the pmtu_expire option of the **`no`** command. pmtu_expire is set to 10 minutes by default.

The syntax of the pmtu command is as follows:

```
pmtu [-inet6] display/[delete [-dst destination] [-gw gateway] ]
```

Flags used with the **`pmtu delete`** command are the following:

`-dst` Specifies the destination of the pmtu entry to be deleted.

`-gw` Specifies the gateway of the pmtu entry to be deleted.

`-inet6` Specifies to display or delete Ipv6 pmtu entry.

If you want to see the pmtu table, simply type a **`pmtu display`** command and the output will be shown as in Figure 7-3.

```
# pmtu display

   dst           gw            If    pmtu     refcnt   redisc_t    exp

 --------------------------------------------------------------------------

9.3.4.174     9.3.5.41        en0    1500        3         11        0

9.3.5.195     9.3.5.196       en0    1500        1         24        0

9.3.5.196     127.0.0.1       lo0   16896       24         11        0

127.0.0.1     127.0.0.1       lo0   16896       12          9        0

9.3.4.75      9.3.5.41        en0    1500        1         24        0
#
# _
```

Figure 7-3 Output of the pmtu display command

7.4 DHCPv6 support

DHCP is an application-layer protocol that allows a client machine on the network to get IP addresses and other configuration parameters from the server. These parameters are defined in options. Options are obtained by exchanging packets between a daemon on the client and another on the server. DHCP as implemented in the previous AIX versions only supports IPv4. Starting with Version 5.3, support for DHCPv6 that uses IPv6 is now available. You need to configure /etc/dhcpv6/dhcpsdv6.cnf for the dhcpv6 server and /etc/dhcpv6/dhcpc6.cnf for the dhcpv6 client. The name of the /etc/dhcprd.cnf file remains unchanged, with newly added key words for DHCPv6 support.

Configuration for DHCPv6 server

There is no dhcpsdv6 file by default and you need to create one for your configuration. If you go to the /usr/samples/tcpip/dhcpv6 directory, you can see some sample files that might help you to understand dhcpv6 configurations. Example 7-10 shows a basic dhcpv6 server configuration.

Example 7-10 dhcpv6 server configuration file /etc/dhcpv6/dhcpsdv6.cnf

```
logging_info{
        logFileSize 4000
        logItem     SYSERR
        logItem     PROTERR
        logItem     WARNING
```

```
        logItem     EVENT
        logItem     ACTION
        logItem     INFO
        logItem     ACNTING
        logItem     TRACE
        numLogFiles     3
        logFileName "/var/tmp/dhcpsdv6.log"
}
duid 1 en0
numprocessthreads 10
preference-number 255
reconfig-policy no
unicast-option    yes
leaseExpiredInterval  3000 seconds
unicast-enable          yes
saveInterval            60 seconds
clientpruneintv         20  seconds

subnet dead:dead:aaaa:: 48 dead:dead:aaaa:aaaa::0006-dead:dead:aaaa:aaaa::000a
{
        interface-id "en1"
        preferred-lifetime      100 seconds
        valid-lifetime          200 seconds
        rapid-commit  yes
        option 23 dead::beef beef:aaaa::bbbb:c aaaa:bbbb::cccc
        option 24 ibm.com austin.ibm.com
}
```

Table 7-3 defines the keywords used in the /etc/dhcpv6/dhcpsdv6.cnf file.

Table 7-3 Keywords in /etc/dhcpv6/dhcpsdv6.cnf

Keyword	Description
duid	Used to identify the server. The following values are allowed: duid 1 interface duid 2 interface duid 3 enterprise number identifier duid number 0xhexdigit
numprocessthreads	Specifies the number of packet processors threads to create. Minimum of one. Each process thread handles one client. The default is 30.
preference-number	Allows the client to identify the server it prefers to obtain information from. The higher the value, the greater the chance the client will use this server for its services. The default and maximum value is 255.

Keyword	Description
reconfig-policy	Allows the server to send reconfiguration message to the client. By default this is not set and is considered off.
unicast-option	Allows containers to offer message exchange by unicasting. This can use used to turn on and off individual containers and subnets even if the server policy differs. By default this is not set and is considered off.
leaseExpiredInterval	Specifies how often addresses in the BOUND state are checked to see if they have expired. If the address has expired, the state is moved to EXPIRED. If a unit is not set, the system default is set to seconds. The default value is 900 seconds.
unicast-enable	Unicast policy for the server, this allows the server to communicate using unicast. By default this is turned on.
saveInterval	Specifies how often the DHCP server should force a save of the open databases. For heavily loaded servers, this should be 60 or 120 seconds. If unit is not set, the system default is set to seconds. The default value is 3600 seconds.
clientpruneintv	Specifies how often the DHCP server removes clients that are not associated with any address (in the UNKNOWN state) from the database. This reduces the memory use of the DHCP server. If units is not set, the system default is set to seconds. The default value is 3600 seconds.
subnet	Specifies subnet to be used. The subnetid must be an IPv6 address. The prefix-length must be a positive integer less than 128.
interface-id	This can only be listed under subnet. Client requests received on this interface will be allowed to obtain addresses.
preferred-lifetime	The preferred lifetime of the IANA or IATA. The default is 43200 seconds.
valid-lifetime	The valid lifetime of the IANA or IATA. The default is 86400 seconds.
rapid-commit	Allows for the server to do rapid commit for the container or globally set. By default this is not set and is considered off.
option 23	DNS servers (IPv6 address).
option 24	Domain list (Domain name).

Configuration for DHCPv6 client

The /etc/dhcpv6/dhcpc6.cnf file is used to configure the DHCPv6 client. There also is a sample file under the /usr/samples/tcpip/dhcpv6 directory. Example 7-11 shows a simple dhcpv6 client configuration.

Example 7-11 dhcpv6 client configuration file: /etc/dhcpv6/dhcpc6.cnf

```
interface  en1  {
               option ia-na {
                ia-id01
                renew-time0x40
                rebind-time0x60
   }
   option  request-option  { 3 23 24 }
```

Table 7-4 defines the keywords used in the /etc/dhcpv6/dhcpc6.cnf file.

Table 7-4 Keywords in /etc/dhcpv6/dhcpc6.cnf

Keyword	Description
ia-na	Specifies option 3. If specified, the client requests non-temporary addresses from the server. Option ia-na takes the following parameters: ia-id <value> renew-time <value> rebind-time <value> These parameters specify the user-preferred values and are optional. The value specified can be a decimal number or a hex number prefixed with '0x'.
request-option	If specified, the client requests a list of options from the server.

Configuration for DHCPv6 relay agent

The /etc/dhcprd.cnf file is the configuration file for the DHCP and BOOTP relay agent. Even though the file name is the same as the previous version, some key words are added to support IPv6. Example 7-12 shows a simple dhcpv6 relay agent configuration.

Example 7-12 dhcpv6 relay agent configuration file:/etc/dhcprd.cnf

```
relay IPv6
server6 9.3.5.196
```

Table 7-5 provides the keywords newly added in the /etc/dhcprd.cnf file.

Table 7-5 Key words in /etc/dhcprd.cnfi

Keyword	Description
relay	IPv4, IPv6, or ALL
server6	Specifies the IPv6 address of the dhcpv6 server.
option6	Specifies the dhcpv6 relay agent options. The keyword is valid only if the relay mode is set to IPv6 or ALL.

7.5 IPv6 functional updates

AIX 5L Version 5.3 is fully compliant with RFC 3542: Advanced Sockets Application Program Interface (API) for IPv6. This API supports advanced applications which typically use raw sockets to access IPv6 or ICMPv6 header fields. The main changes implemented in Version 5.3 for IPv6 are:

- Structure and constant definitions required for applications to use IPv6 raw sockets have been updated in the AIX header files to conform to RFC 3542.
- A number of socket options and accompanying ancillary data types have been added to allow UDP/raw socket based applications to obtain information about incoming IPv6 datagrams. For example, applications can choose to receive data about the received interface, destination address, and received hop limit of an incoming IPv6 datagram. Applications can also receive extension headers from incoming IPv6 datagrams. The following are socket options: IPV6_RECVPKTINFO, IPV6_RECVHOPLIMIT, IPV6_RECVTCLASS, IPV6_RECVRTHDR, IPV6_RECVHOPOPTS, IPV6_RECVDSTOPTS.
- Applications can also set socket options to control fields in outgoing IPv6 datagrams. They may also specify extension headers to be included in outgoing datagrams. The following are socket options: IPV6_PKTINFO, IPV6_NEXTHOP, IPV6_TCLASS, IPV6_RTHDR, IPV6_HOPOPTS, IPV6_DSTOPTS, IPV6_RTHDRDSTOPTS.
- A number of additional features can be enabled using IPv6 socket options:

  IPV6_USE_MIN_MTU — Controls Path MTU discovery.

  IPV6_DONTFRAG — Prevents fragmentation of outgoing packets on this socket.

  IPV6_RECVPATHMTU — When an application sends packets too big for the outgoing path MTU, this options allows the application to receive the actual path MTU as ancillary data.

IPV6_PATHMTU	Allows application to call this socket option to obtain the path MTU at any time.

- A number of new library functions have been implemented that allow applications to easily construct IPv6 extension headers.

  Functions to build/read a routing header:

  - socklen_t inet6_rth_space(int, int);
  - void *inet6_rth_init(void *, socklen_t, int, int);
  - int inet6_rth_add(void *, const struct in6_addr *);
  - int inet6_rth_reverse(const void *, void *);
  - int inet6_rth_segments(const void *);
  - struct in6_addr *inet6_rth_getaddr(const void *, int);

  Functions to build/read hop-by-hop/destination options header:

  - int inet6_opt_init(void *, socklen_t);
  - int inet6_opt_append(void *, socklen_t, int, uint8_t, socklen_t, uint_t, void **);
  - int inet6_opt_finish(void *,socklen_t, int);
  - int inet6_opt_set_val(void *, int, void *, socklen_t);
  - int inet6_opt_next(void *, socklen_t, int, uint8_t*, socklen_t *, void **);
  - int inet6_opt_find(void *, socklen_t, int, uint8_t, socklen_t *, void **);
  - int inet6_opt_get_val(void *, int, void *, socklen_t);

- Version 5.3 modifies the behavior of AF_INET6 IPPROTO_RAW sockets to conform to RFC 3542. Only packet data will be delivered on these sockets when an application performs a recv call. The IPv6 header will no longer be delivered along with incoming data.

  Previous versions of AIX delivered IPv6 headers along with packet data on AF_INET6 IPPROTO_RAW sockets. If an application wants to continue to receive the IPv6 header along with the data, it must set socket option IPV6_AIXRAWSOCKET and recompile.

Support for IPv6 over PPP

Version 5.3 now provides IPv6 functional support over PPP. Refer to AIX 5L Version 5.3 documentation for further details.

7.6 Interface layer support for hotplug network adapters

Prior to AIX 5L Version 5.3, the number of network interfaces defaults to 256. It is a configurable network option (**`ifsize`**). This determines the maximum number

of interface layer structures supportable for that interface type. The maximum acceptable value for ifsize is 1024. With the introduction of virtualization technologies, where there can be thousands of interfaces on a machine, each requiring interface layer structure support, the limitation needs to be increased. Starting with Version 5.3, any number of interfaces (real or virtual) can be supported by implementing if_layer hotplug support. Since most of the current focus of interface virtualization is on Ethernet, it introduces the hotplug features for Ethernet only. The other interface types such as token-ring will still continue to use the current (Version 5.2 and earlier) allocation strategies where the network option ifsize (default 256, maximum size 1024) determines the maximum number of supportable interfaces of that type at the interface layer. Table 7-6 contrasts the network option ifsize differences between releases.

Table 7-6 Differences of network option ifsize

AIX 5L Version 5.2 or earlier	AIX 5L Version 5.3
Network option ifsize defaults to 256 and maximum of 1024.	No ifsize limitation for Ethernet.

You can still see network option ifsize with the **no** command. However, it does not mean anything for Ethernet. The help text for **no -h ifsize** is shown in Example 7-13; most of the explanation applies for token-ring as well.

Example 7-13 Network option ifsize

```
# no -o ifsize
ifsize = 256
#
# no -h ifsize
Help for tunable ifsize:
Specifies the maximum number of network interface structures per interface.
This limit does not apply for ethernet interface structures for which the
infrastructure expands dynamically to handle any number of ethernet interface
structures. So the rest of the message applies to other interface types. The
default value is 256. If the system detects at boot time that more adapters of
a type are present than would be allowed by the current value of ifsize, it
will automatically increase the value to support the number of adapters
present, so there is no reason to change it in runtime. ifsize is a Reboot
attribute. ifsize ranges from 8 to 1024.
```

It needs to be noted that the interface layer for hotplug network adapters grows dynamically on an as needed basis for efficient resource allocation.

7.7 Support for SLP client service

Service Location Protocol Version 2 (SLPv2) provides a flexible and scalable framework for providing hosts with access to information about the existence, location and configuration of networked services. Any TCP/IP application can take advantage of this protocol, which requires little or no static configuration to discover the network services. SLP eliminates the need for a user to know the name of a network host supporting a service.

For example, if a user wants to resolve a hostname-to-ipaddress mapping or ipaddress-to-hostname mapping, the user has to know the DNS server IP address first so that the resolver can be configured in /etc/resolv.conf file before any name resolution can be performed properly. However, if the application is using SLP protocol, the user only needs to supply the desired type of service and a set of attributes which describe the service. Based on the description, the SLP resolves the network address of the host which provides the service that the user wants. SLP provides a dynamic configuration mechanism for applications in local area networks. Applications are modeled as clients that need to find servers attached to any of the available networks within an enterprise.

Version 5.3 supports the SLP client part of the APIs and configuration file. The SLP client APIs are the following:

- SLPOpen
- SLPClose
- SLPFree
- SLPFindSrvs
- SLPFindSrvTypes
- SLPFindAttrs
- SLPEscape
- SLPUnescape
- SLPParseSrvURL
- SLPFindScopes
- SLPGetProperty
- SLPSrvTypeCallback
- SLPSrvURLCallback
- SLPAttrCallback

The /etc/slp.conf file is the configuration file for SLP; Table 7-7 on page 309 explains the parameters used in this file.

Table 7-7 Parameters for /etc/slp.conf

Parameters	Descriptions
net.slp.maxResults	A 32-bit integer that gives the maximum number of results to accumulate and return for a synchronous request before the time out. Positive integers and -1 are legal values. A value of -1 indicates that all results should be returned. The default value is -1.
net.slp.useScopes	A value-list of strings that indicate the only scopes a UA or SA is allowed to use when making requests or registering. Otherwise, indicates the scopes a DA must support. The default scope DEFAULT is used if no other information is available.
net.slp.DAAddress	A value-list of IP addresses or DNS resolvable host names giving the SLPv2 DAs to use for statically configured UAs and SAs. The default is none.
net.slp.isBroadcastOnly	A boolean indicating whether broadcast should be used instead of multicast. The default is false (that is, multicast is used).
net.slp.multicastTTL	A positive integer less than or equal to 255, giving the multicast TTL. The default is 255 (in seconds).
net.slp.DAActiveDiscoveryInterval	A 16-bit positive integer giving the number of seconds between DA active discovery queries.
net.slp.multicastMaximumWait	A 32-bit integer giving the maximum amount of time to perform multicast, in milliseconds. This property corresponds to the CONFIG_MC_MAX parameter in the protocol specification. The default is 15000 (in ms).
net.slp.multicastTimeouts	A value-list of 32-bit integers used as time outs, in milliseconds, to implement the multicast convergence algorithm.
net.slp.DADiscoveryTimeouts	A value-list of 32-bit integers used as time outs, in milliseconds, to implement the multicast convergence algorithm during active DA discovery.
net.slp.datagramTimeouts	A value-list of 32-bit integers used as time outs, in milliseconds, to implement unicast datagram transmission to DAs.

7.8 Stream Control Transmission Protocol (SCTP)

The Stream Control Transmission Protocol (SCTP) is a new reliable transport protocol. It was proposed to the Internet Engineering Task Force (IETF) as a telephony signaling transport above the IP protocol layer.

7.8.1 SCTP basics

SCTP is a message-oriented transport layer protocol that can be compared to the UDP or TCP protocols. In general, SCTP provides more flexibility for certain applications than TCP or UDP. Figure 7-4 shows the network layers and where the SCTP fits.

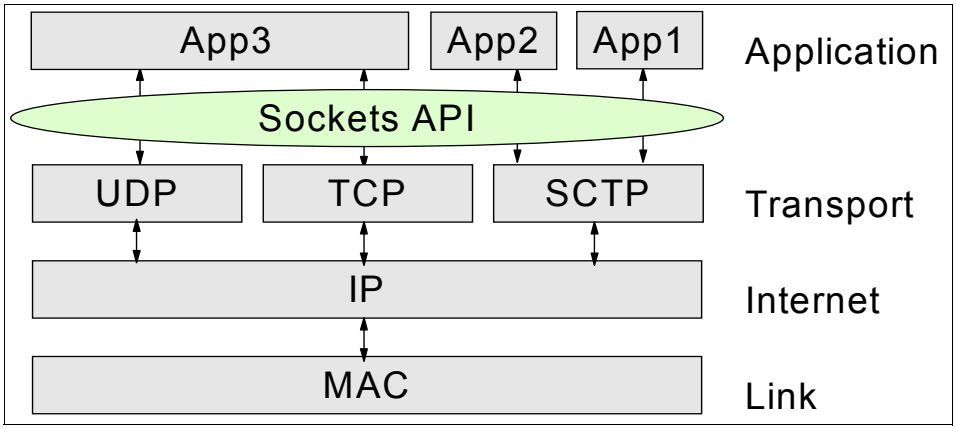

Figure 7-4 The IP network protocol layers

The SCTP architecture is based on IETF RFC 2960 available at:

http://www.ietf.org/rfc/rfc2960.txt

The implementation of the sockets API extensions for SCTP are based on a subset of the IETF draft draft-ietf-tsvwg-sctpsocket-04. At the time of writing, the latest version of the draft is available at:

http://www.ietf.org/

There are other related RFCs that describe the User Adaptation Layers and enhancements, such as RFC 3057, RFC 3331, RFC 3332, RFC 3309 or RFC 3286.

SCTP is a connection-oriented protocol, similar to TCP, but provides message-oriented data transfer, similar to UDP. For applications like Voice over IP (VoIP) the SCTP protocol works out to be ideal because TCP with its inherent

properties like byte-oriented transfer and strict in-order delivery can prove to be very slow, and UDP is too unreliable to be used directly without application overhead to provide reliability. Table 7-8 highlights the major differences between the three transport protocols.

Table 7-8 TCP, UDP, and SCTP comparison

	TCP	UDP	SCTP
Reliability	Reliable	Unreliable	Reliable
Connection management	Connection oriented	Connectionless	Connection oriented
Transmission	Byte oriented	Message oriented	Message oriented
Flow control	Yes	No	Yes
Congestion control	Yes	No	Yes
Fault tolerance	No	No	Yes
Data delivery	Strictly ordered	Unordered	Partially ordered
Security	Yes	Yes	Improved

7.8.2 SCTP API in AIX

AIX implements the SCTP at kernel level as a loadable kernel extension accessible through SCTP socket APIs. The SCTP socket APIs were designed to maintain consistency with existing socket APIs, like TCP or UDP, but at the same time, provide a base for access to new SCTP features. The SCTP APIs provide compatibility so that most existing TCP and UDP applications can be migrated to SCTP with few changes.

At the time of writing, only UDP-style SCTP sockets for IPv4 are implemented in Version 5.3.

The programming of an SCTP communication is very similar to classic UDP. Before you start to implement SCTP sockets, you need to include the necessary header files:

```
#include <sys/socket.h>
#include <netinet/sctp.h>
```

First you have to create the socket with type of SOCK_SEQPACKET and with protocol of IPPROTO_SCTP. The socket() function creates an SCTP-type socket and initializes the SCTP structures with default values. The socket has to be created on the client and server sides, as in the following example:

```
so = socket(AF_INET, SOCK_SEQPACKET, IPPROTO_SCTP);
```

You can then use the bind(), listen(), connect(), sendmsg(), recvmsg(), and close() system calls in the same way as for UDP protocol.

Since SCTP is a connection-oriented protocol, unlike UDP, multiple incoming connections from clients to a single server are handled on a single server socket. The socket buffer is, therefore, shared among all these connections. Since this can be limiting for some applications, SCTP provides new APIs in addition to the standard socket APIs to manipulate this. The two new APIs supported in this release of AIX 5L Version 5.3 are:

- sctp_peeloff()
- sctp_opt_info()

The sctp_peeloff() call is used by applications to branch off an existing association into a separate socket.

The sctp_opt_info() call is used by applications to pass information in or out of the SCTP stack.

More information about the socket API for SCTP can be found in the SCTP Socket API draft at the following URL:

http://www.ietf.org

7.8.3 SCTP management in AIX

Starting with AIX 5L Version 5.3, SCTP is delivered as the bos.net.sctp LPP fileset. The fileset is not installed at initial system installation and you will need to install it in order to use SCTP.

Once the SCTP is installed, you get the **sctpctrl** command that handles the SCTP device driver and its environment. This command accepts the following flags:

load Loads and initializes the SCTP kernel extension.

unload Unloads the SCTP kernel extension.

dump Dumps various data structures used by the SCTP kernel extensions. This flag is intended to be used by IBM service.

set Sets attributes of the SCTP driver.

get Displays specified or all SCTP attributes.

stats Displays SCTP statistics.

Example 7-14 shows the procedure of loading the kernel SCTP extension using the **sctpctrl load** command, listing the SCTP settings using the **sctpctrl get** command, and unloading the SCTP kernel extension using the **sctpctrl unload**

command. The output of the **stpctrl stats** command gives detailed device driver statistics and it is not displayed in the example.

Example 7-14 sctpctrl command

```
# sctpctrl load
# sctpctrl get
        sctp_assoc_maxerr = 10
         sctp_cookie_life = 60
        sctp_delack_timer = 4
        sctp_dontdelayack = 1
                 sctp_ecn = 1
      sctp_ephemeral_high = 65535
       sctp_ephemeral_low = 32768
           sctp_instreams = 2048
            sctp_maxburst = 8
          sctp_outstreams = 10
         sctp_path_maxerr = 5
       sctp_pmtu_discover = 1
              sctp_rttmax = 60
              sctp_rttmin = 1
           sctp_recvspace = 65536
           sctp_sendspace = 65536
       sctp_send_fewsacks = 0
# sctpctrl unload
```

The **sctpctrl** command is not persistent across reboots and you may need to call it at the system startup, for example from the /etc/inittab or the /etc/rc.tcpip script.

7.9 Network RAS enhancements

In order to improve network RAS, the following features are included in AIX 5L Version 5.3:

- For better display of the internal structures: KDB subcommands **tcb, udb, tcpcb, sock, sockinfo, netm, ndd** and **ifnet** are modified.
- For improving network memory debug: KDB subcommands **netm** and **kmbucket** have new options.
- For ns_alloc/free event tracing:

  New options are added in the **no** command to turn on/off and configure filters. (type of buffer, size of buffer).

  New kdb subcommand **nsdbg** is introduced to display events.

- For checking network options based on their dependence: new options are added in the **no** command.

KDB subcommands enhancements

The following KDB subcommands are modified to produce better display or improve network memory debug. The options in bold characters are new or modified in Version 5.3.

- tcb **[-s] | [-b <bucket_index>]** [symbol | address]
- udb **[-s] | [-b <bucket_index>]** [symbol | address]
- tcpcb [tcp | udp] [symbol | address] | **[-s [udp | tcp]]**
- sock **[-d]** [tcp | udp] [symbol | address]

  sock **[-s] [tcp | udp]**

  sock **[-f]**

  sockinfo **[-d]** [address] [<type_of_address>]
- ndd **[-s] | [-n nddname]**
- netm [[-c display_count] **[-t type[,type]*] [-s size[,size]*]]**
- kmbucket **-k <bucket_address>**

New kdb nsdbg subcommand

The **nsdbg** subcommand displays the ns_alloc and free event records stored in the kernel. This function is only available if the **ndd_event_tracing** parameter is turned on using the **no** command.

```
#nsdbg [-i starting_index] [-c display_count] [-n nddname[,nddname[,...]] ]
```

Example 7-15 shows the output of the nsdbg subcommand.

Example 7-15 The output of nsdbg subcommand

```
# no -o ndd_event_tracing=1
Setting ndd_event_tracing to 1

#In order to create some events for nsdbg, do the following
#ifconfig en0 down
#ifconfig en0 up
#kdb
(0)> nsdbg
ndd_name=ent0, ndd_addr=F10006000C662030
index =000503, type =Free , cpu=0000
refcnt=000000, flags=00000000
Stack Trace back
            .soclose2+000318
```

```
            .soclose+000014
            .soo_close+0000C8
            .closef+00006C
            .closefd+0000F8
            .close+000284
            .sys_call_ret+000000

ndd_name=ent0, ndd_addr=F10006000C662030
index =000502, type =Alloc, cpu=0000
refcnt=000000, flags=00000000
Stack Trace back
            .soconnect+000358
            ._connect+000260
            .connext+000018
            .sys_call_ret+000000
(0)>
```

New options for the no command

For network RAS support, the network options described in Table 7-9 are added to Version 5.3.

Table 7-9 New network options added in Version 5.3

no option	Description
ndd_event_tracing	Turn on/off ns_alloc/free event tracing and define event buffer size for ns_alloc/free event. (similar to net_malloc_police)
ndd_event_name	Limit ns_alloc/free event tracing to interface name/alias containing one element of the string list. If the string is all or empty, events for all interfaces will be captured. (Up to 8 different names in a list.)
net_buf_size	Limit net_malloc/free event tracing for the size list. A size of 0 means all.
net_buf_type	Limit net_malloc/free event tracing for type contained in this string list. If the string is all or empty, events for all type will be captured.

7.10 Enable IP Security with intervening NAT devices

In order to establish a secure communication network, IP Security is widely chosen in TCP/IP network. In previous versions of AIX, IP Security has been supported without support for NAT. However, Network Address Translation (NAT) protocol is also widely used in Internet Connection sharing and ships as a standard feature in every router and edge device. The IP Security protocol

depends on identifying remote endpoints and their policy based on the remote IP address. When intermediate devices such as routers and firewalls translate a private address to a public address, the required authentication processing in IP Security might fail because the address in the IP packet has been modified after the authentication digest was calculated. With the new IP Security NAT support, devices that are configured behind a node that performs network address translation are able to establish an IP Security Tunnel. The IP Security code is able to detect when a remote address has been translated. Using the new IP Security implementation with support for NAT allows a VPN client to connect from home or on the road to the office through an Internet connection with NAT enabled. Figure 7-5 illustrates the difference between a NAT-enabled IP Security implementation, with UDP encapsulated traffic, and an implementation that is not NAT-enabled.

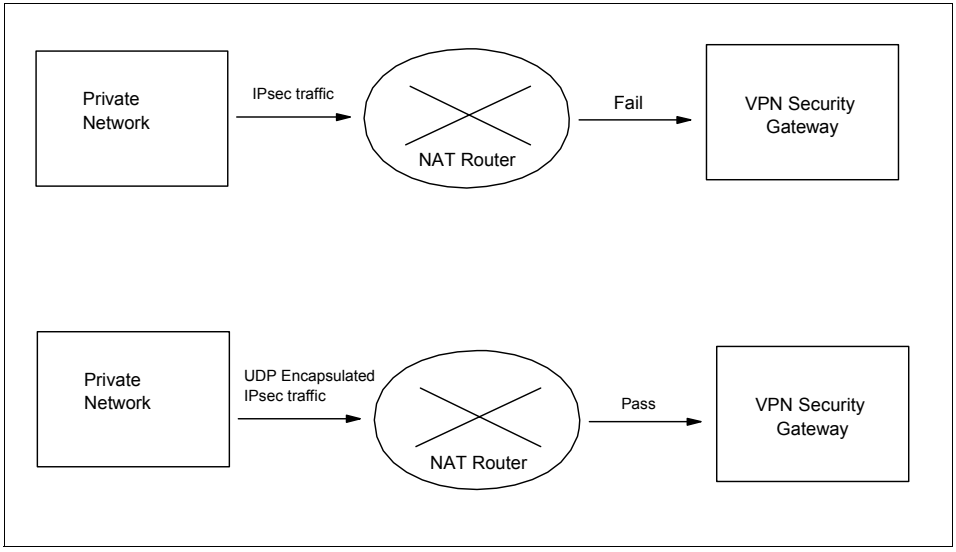

Figure 7-5 NAT-enabled IP security

NAT devices support for IP Security network

Starting with AIX 5L Version 5.3, NAT devices support for IP Security is now available. In order to use the NAT support in IP Security, you need to set the ENABLE_IPSEC_NAT_TRAVERSAL variable in the /etc/isakmpd.conf file as shown in Example 7-16.

Example 7-16 /etc/isakmpd.conf used or NAT support

```
~
#
#     MULTI_LOGIN_OK
#
```

```
# 7) Enable IPsec NAT Traversal
#    You can enable IPsec NAT Traveral and UDP encapsulation functionality
#    by specifying following text:
     ENABLE_IPSEC_NAT_TRAVERSAL
#
# 8) NAT Keep Alive Interval
#
#    The syntax for this option is:
#
#       NAT_KEEPALIVE_INTERVAL = keepalive interval value (seconds)
#
#    The NAT_KEEPALIVE_INTERVAL is optional. The default value is 20 seconds.
#
     NAT_KEEPALIVE_INTERVAL=30
#
```

When this variable is set, filter rules are added to send and receive traffic on port 4500. Example 7-17 shows the filter rules when the ENABLE_IPSEC_NAT_TRAVERSAL variable is set.

Example 7-17 Filter rules added with ENABLE_IPSEC_NAT_TRAVERSAL

```
Dynamic rule 2:
Rule action         : permit
Source Address      : 0.0.0.0 (any)
Source Mask         : 0.0.0.0 (any)
Destination Address : 0.0.0.0 (any)
Destination Mask    : 0.0.0.0 (any)
Source Routing      : no
Protocol            : udp
Source Port         : 0 (any)
Destination Port    : 4500
Scope               : local
Direction           : inbound
Fragment control    : all packets
Tunnel ID number    : 0

Dynamic rule 3:
Rule action         : permit
Source Address      : 0.0.0.0 (any)
Source Mask         : 0.0.0.0 (any)
Destination Address : 0.0.0.0 (any)
Destination Mask    : 0.0.0.0 (any)
Source Routing      : no
Protocol            : udp
Source Port         : 4500
Destination Port    : 0 (any)
Scope               : local
Direction           : outbound
```

```
Fragment control    : all packets
Tunnel ID number    : 0
```

Setting the ENABLE_IPSEC_NAT_TRAVERSAL variable also adds some additional filter rules in the filter table. Special IPSEC NAT messages use UDP encapsulation and filter rules must be added to allow this traffic to flow. In addition, in phase 1 signature mode is required. If IP Address is used as the identifier in the certificate, it should contain the private IP address. IP Security also needs to send NAT keep alive messages to maintain the mapping of the original IP Address and the NAT address. The interval is specified by the NAT_KEEPALIVE_INTERVAL variable in /etc/isakmpd.conf file. This variable specifies how frequently, in seconds, NAT keepalive packets are sent. If you do not specify a value for NAT_KEEPALIVE_INTERVAL, a default value of 20 seconds is used.

Limitations when using NAT exchanges

NAT exchanges supports ESP only. AH (Authentication Header) and ESP (Encapsulating Security Payload) can be used to implement IP security. However, AH does not a change in IP address in IP packet which is necessary for NAT. Additionally, a connection using NAT must select tunnel mode so that the original IP address is encapsulated in the packet. Transport mode and addresses with NAT are not compatible.

Refer to AIX 5L Version 5.3 Documentation Security Guide for further detail.

7.11 AIX SNMP subagent enhancements

SNMP manager, agent, and subagent are the primary functional entities. SNMP manager is the system management application, such as IBM Director, IBM Netview and other network management systems (NMS). SNMP agent accepts the requests from the manager, verifies the authentication, passes the requests to the appropriate SNMP subagent, and responds to the manager with the data or execution result from the subagent. SNMP subagent executes the specific system management tasks. AIX Enterprise SNMP MIB Support, SNMP subagent aixmibd is first shipped with AIX 5L Version 5.2. In Version 5.2, about half of the MIB objects defined in the MIB were implemented, while all are now implemented in Version 5.3.

7.11.1 MIB objects added in Version 5.3

SNMP subagent aixmibd provides the system management data, and manages the system resources through the standard SNMP. It also generates traps for

some critical system resources. It manages and provides data for the system resources shown in Table 7-10.

Table 7-10 New MIB resources implemented in Version 5.3

Supported resources	Detail
System environment information	Number of CPUs, machine type, machine serial number, system date and time, maximum number of processes allowed per user, maximum number of fixed licenses, number of available fixed licenses
Substystem	Name, PID, status
Subserver	Name, description, status, the subsystem name which the subserver belongs to
print queue	Name, device, status, start/stop queue, discipline, accounting data file, host name where the queue is running, remote queue name
system user	Name, UID, home directory, shell, primary group, groups which the user belongs to, enable or disable local login, enable/disable remote login, allowable attempts, password maximum age, status, reset login count
System group	Name, GID, admin group or not, admin users for this group, the users who are in this group
Printer device	Name, type, interface, status, description, location, port number
Tape drive	Name, type, interface, status, description, location
Hard disk	Name, type, interface, status, description, location, size
Memory	Name, type, description, location, size
CDROM	Name, type, description, location, interface
SCSI	Name, description, location, status, adapter ID
Processor	Name, type, description, speed
Network interface	Name, type, interface, status, location, description
Adapter	Name, description, type, interface, status

You might want to take a look at the MIB resources already supported with Version 5.2. Version 5.3 continues to support these resources, which are identified in Table 7-11 on page 320.

Table 7-11 MIB resources implemented in version 5.2

Supported resources	Detail
System environment information	CPU utilization, CPU utilization threshold, system run level, system state, latest trap message
Agent control variables	Polling intervals, trap thresholds, polling enable/disable, agent access
File systems	Name, mount point, total size, free size, total inodes, free inodes, type (NFS, jfs, jfs2)
Volume group	Name, identifier, state, total size, free size, number of total logical volumes, number of open logical volumes, number of active physical volumes
Logical volume	Name, the volume group name which the logical volume resides in, type, label, mount point, size, state
Paging space	Name, volume group name, physical volume name, size, percent used, status, type
Process	Process PID, UID, PPID, group name, priority, command, status, tty
Current logged in user	Name, tty, the host name from which the user got logged in, date and time
Important resource monitoring traps	File system full trap, CPU utilization high trap, volume group full trap, paging space full trap, too many attempts to get logged in trap

7.11.2 The snmptrap command enhancement

Starting with Version 5.3, you have a new option with **snmptrap** command: **snmptrap -h**. The -a flag specifies a host where the AIX SNMP agent (snmp) must be running and the SNMP agent forwards this trap to network mangers. However, the -h flag does not require the AIX SNMP agent to forward the trap message to network managers, and it sends the trap directly to the network manager. If there are no -h and -a flags, the trap will be sent to the AIX SNMP agent on the local host.

```
snmptrap  [-d] [-a hostname|-h dest] [-c community] -m message
```

-d Enable the debug facility.

-a hostname Connect to the snmp agent running on the specified AIX host.

-h dest Any destination hostname on which the trap will be received. This option makes it possible to send the

	message to any platform. Either -a hostname or -h dest can appear in the command line, but not both.
-c community	Use specified community name.
-m message	Define the message sent by the snmptrap command. the -m option must be the last option.

If you want to send a trap to the network manager running on a Linux platform and where the host name is myahn, use the following command:

```
# snmptrap -h myahn -m hello world
```

7.12 SMBFS enhancements

Server Message Block File system (SMBFS) allows access to shares on SMB servers as local file systems on AIX. In this file system, the user can create, delete, read, write, and modify the access times of files and directories. The owner or access mode of files and directories cannot be changed. SMBFS can be used to access files on an SMB server. The SMB server is a server running Samba; an AIX server running AIX Fast Connect; or a Windows® XP, Windows NT®, or Windows 2000 server or workstation. Each of these server types allows a directory to be exported as a share. This share can then be mounted on an AIX system using SMBFS. In AIX 5L Version 5.3, SMBs following newer protocol are used for file open, allowing the AIX file attributes to be mapped to a larger set of Windows file attributes. To install SMBFS on an AIX system, install the bos.cifs_fs package from AIX install CD-ROMs (Example 7-18). When the bos.cifs_fs package in installed, the device nsmb0 is created. This device allows the **mount** command to establish a connection between the SMB server and the client.

Example 7-18 Configuring SMB clients

```
# lslpp -L |grep cifs
  bos.cifs_fs.rte            5.3.0.0    C     F    Runtime for SMBFS
  bos.cifs_fs.smit           5.3.0.0    C     F    SMIT Interface for SMBFS

# lsdev -C|grep nsmb
nsmb0      Available               N/A
```

7.13 Ported 64-bit DES NFS kernel extension

The DES encryption kernel library used for the secure NFS is enabled for the 64-bit environment. This library is part of the DES LPP that is part of the AIX 5L Version 5.3 Expansion pack.

7.14 BIND domain search improvement

Before a user can use a nameserver to map hostname to IP address, and vice versa, on the client side the user must configure the /etc/resolv.conf file. In this file, the user specifies the nameserver by configuring sets of nameserver IP_address values for each nameserver he wants to use. The user also specifies how many domain names to use in the DNS queries by configuring the search DomainName ... statement. A typical file looks like the following:

```
nameserver  9.53.159.2
nameserver  9.3.149.2
search    ibm.com   austin.ibm.com   aoot.austin.ibm.com
```

A user can specify up to three nameserver lines in the /etc/resolv.conf file. In cases where there is only one nameserver, it opens a UDP socket, sends out query, receives the response, and then closes the socket. This is the sequence that takes place for every DNS query that the client makes. This can be very inefficient in cases where there are multiple queries.

Moreover, a user can specify a maximum of six domain names on the search line. This does not always meet user needs.

AIX 5L Version 5.3 has made enhancements in both these areas.

The DNS resolver uses either TCP socket or UDP socket for the hostname and IP address resolution. In any user application, if there is no sethostent(1) call which explicitly asks the resolver to use TCP socket, by default, the resolver will use UDP connection. There are cases when the resolver may send out multiple queries using UDP connection before it can resolve the name. For each query, it will do socket(), sendto(), poll(), recvfrom(), and close() in sequence. For multiple queries, there will be multiple socket() and close() calls. This will cause performance degradation.

In Version 5.3 the UDP socket is kept open for multiple queries against the same nameserver until it finishes the name resolution, then calls close() once to close the UDP socket. In general this results in performance improvement.

In cases where the answer returned from the nameserver is truncated, the UDP socket is closed and a TCP socket is opened to send out the query to the same nameserver.

As for the domain names, the limit of six on the search keyword has been replaced by the maximum number of characters in the search string to be 1024. The search keyword now can support any number of domain names as long as the number of characters do not exceed 1024.

7.15 Network API support for ARP

AIX 5L Version 5.3 provides new APIs for ARP: arpresolve_common and arpupdate. Prior to Version 5.3, whenever you change the arptab structure, you will break the interface (if) layer code for all the devices using ARP. Version 5.3 now provides a set of APIs to access the ARP table to avoid breaking the if code.

The arpresolve_common function

The **arpresolve_common** function is called by arpresolve from the if layer of the interface. It does the following, with the parameters provided in Table 7-12:

- If the arp entry is complete, then it will return the hwaddr and if_dependent data if the if_dependent is true.
- If entry is not complete, then add the mbuf to at->at_hold.
- If entry does not exist, then create a new entry by calling arptnew and adding the mbuf to at->at_hold.
- The arpresolve_common calls arpwhohas when it creates a new arp entry or when the timer for the incomplete arp entry (whose IP address is as per pointer dst) has expired.

```
int  arpresolve_common ( register struct arpcom *ac, struct mbuf *m, int
   (*arpwhohas)(register struct arpcom *ac, struct in_addr *addr,
   int skipbestif, void *extra), struct sockaddr_in *dst, u_char * hwaddr,
   int szhwaddr, void *extra, union if_dependent *if_dependent)
```

Table 7-12 Parameters for arpresolve_common

Parameter	Description
ac	Pointer to the arpcom structure.
m	Pointer to the mbuf which will be added to the list awaiting the arp table entry completion.
dst	Pointer to the sockaddr_in structure this structure has the destination IP address.
int (*arpwhohas) (....)	Function pointer to arpwhohas.
hwaddr	Pointer to the buffer. This buffer is filled with the hardware address if it finds a completed entry in the arp table for the destination IP address.
szhwaddr	Size of buffer hwaddr.

extra	A void pointer which can be used in future for IF layer to pass extra structures to arpwhohas.
if_dependent	Pointer to the if_dependent structure. The function arpresolve_common uses this to pass the if_dependent data which is part of arptab entry to the calling function.

The arpupdate function

The **arpupdate** function is called by arpinput functions from the if layer. The function does the following, with the parameters provided in Table 7-13:

- Looks at the arp table for h/w address for the given IP address.
- Updates the arp entry for a given IP address.
- Does reverse arp lookup.

```
int arpupdate (ac, m, hp, action, prm)
      struct arpcom *ac;
      struct mbuf *m;
       int action;
      struct arpupdate_parm     *prm;
```

Table 7-13 Parameters for arpupdate

Parameter	Description
ac	Pointer to the arpcom structure.
m	Pointer to the mbuf which contains the arp response packet which was received by the interface.
hp	Pointer to buffer.
action -	LOOK (1)Look for the IP address [isaddr] in the arp table and return h/w address & if_dependent structure. LKPUB (2) - Look for the IP address [isaddr] in the arp table and return h/w address & if_dependent structure only if flag set to ATF_PUBL. UPDT (3) - update the arp entry for an IP address [isaddr] or if no arp entry is there then create a new one and also update the if_dependent structure using function ptr passed in prm structure. REVARP (4) - Reverse arp request. hwaddr contains the h/w address, szhwaddr is size and the IP address will returned in saddr if an entry is found.
prm	Pointer to structure arpudpate_parm

7.16 Withdraw older versions of TCP applications

Starting with Version 5.3, the following services will not be supported:

- The previous versions of AIX support BIND Versions 4, 8 and 9. With Version 5.3, AIX does not support BIND Version 4 and only supports BIND Versions 8 and 9.
- TCPIP LDAP name resolution that used the.IBM SecureWay® Directory schema will no longer be supported in 5.3. In 5.3, TCPIP LDAP name resolution uses only RFC 2307 schema. See "TCP/IP Name Resolution" in the *System Management Guide: Communications and Networks* for detailed information.

8

Security

This chapter is dedicated to the latest security topics as they apply to AIX 5L. Topics include:

- LDAP user authentication enhancements
- Full PAM exploitation
- Prioritizing LDAP servers
- Rsh login control
- **chpasswd** command
- Initial login license limit increased
- NIS/NIS+ enhancements
- Support for passwd.adjunct NIS map

8.1 LDAP user authentication enhancements

The LDAP authentication package in AIX 5L Version 5.2 authenticates users by retrieving the user password from the LDAP server database and doing authentication locally. In order to retrieve a user password from the LDAP server, each client needs to have privileged access to the server. This privilege is also used by client systems for password change or user login activity logging. The privileged access is made available by storing the LDAP server privileged DN and password on each client system. In some environments, it may be a concern to allow any local root user of a configured client system to have privileged access to the LDAP database. AIX LDAP package in Version 5.3 now provides a mechanism to eliminate the need for privileged access rights from client systems for better security.

UNIX-type and LDAP-type authentication

The AIX LDAP authentication package was first implemented in AIX 4.3. In order to keep the same AIX user authentication and management behavior in the LDAP environment as using flat ASCII files (/etc/passwd, for example), client systems connect to the server using a privileged account (LDAP server administrator account). Such privileged connection serves the purpose of maintaining the AIX user management behavior, such as user creation and removal, password change, user login log, and so on. The only trade off is security: Each local root user of a client system has full control of the LDAP server database. This could be a major concern for LDAP deployment, especially if multiple clients are involved. An alternative user authentication mechanism is introduced in Version 5.3: LDAP server authentication, or simply LDAP-type authentication. LDAP-type authentication does not require privileged connections between client systems and the LDAP server. LDAP-type authentication involves constructing a distinguished name (DN) from the user account name, sending the constructed DN and the login password to the LDAP server, and requesting that the server determine if the user password matches the one stored in the server database.

Table 8-1 Difference between UNIX-type and LDAP-type authentication

UNIX-type authentication	LDAP-type authentication
Authentication mechanism of fetching user's encrypted password and authenticating user locally.	Authentication mechanism of sending user-entered password and constructed DN (from user name) to the LDAP server and having the server do the authentication.

Advantages of LDAP-type authentication

There are two advantages if you choose LDAP-type authentication:

- A more secure LDAP database. No privileged connection between server and client is required. It can be configured such that the local root cannot change password, create, delete, or modify user accounts. The current attribute-level access control feature of IBM Directory server (Version 4.1 and later) supports fine grained configuration with respect to what can be done by a local root user to the LDAP database.
- Multiple encryption mechanism support by the LDAP server. Prior to Version 5.3, only support for the *crypt* mechanism for user password encryption existed. With LDAP-type authentication, password encryption is done by the LDAP server, and the password encryption is hidden from users. The IBM Directory currently supports imask, crypt, sha, as well as clear text password.

Note: The password is sent in clear text over the wire to the LDAP server for LDAP-type authentication. You may need to configure SSL when using LDAP-type authentication, if you are concerned about passwords being sent in clear text form.

Configuration file

Example 8-1 shows new controls added to the /etc/security/ldap/ldap.cfg file. The explanation for each of the controls is also added to this file. The old credential definition of ldapadmindn and ldapadmpwd will be removed from the file. However, the secldapclntd will keep its ability to read ldapadmindn and ldapadmpwd for backward compatibility. In case both binddn and ldapadmindn exist in the ldap.cfg file, secldapclntd will use ldapadmindn first.

Example 8-1 New controls added in /etc/security/ldap/ldap.cfg

```
# LDAP server bind DN.
#binddn: cn=admin

# LDAP server bind DN password
#bindpwd:
# Authentication type. Valid values are unix_auth and ldap_auth. Default is
unix_auth.
# unix_auth - retrieve user password and authenticate user locally.
# ldap_auth - bind to ldap server to authenticate user remotely by the LDAP
server.
#authtype: unix_auth
# Search mode. Valid values are ALL and OS. Default is ALL.
# ALL - returns all attributes of an entry.
# OS  - returns only the OS required attributes of an entry.
#       Non-OS attributes like telephone number, binary images,
#       will not be returned.
```

```
# Note: use OS only when user entry has many non-OS required
# attributes or attributes with large values, e.g., binary data,
# to reduce sorting effort by the LDAP server.
#
#searchmode: ALL
# Whether to use Kerberos for initial bind to the server.
# useKRB5       - valid value is either "yes" or "no". Default is "no".
# krbprincipal  - Kerberos principal used to bind to the server.
# krbkeypath    - Path to the kerberos keytab, default to
#                 /etc/security/ldap/krb5.keytab
# krbcmddir     - Directory that contains the Kerberos kinit command.
#                 Default to /usr/krb5/bin/.
#useKRB5:no
#krbprincipal:principal
#krbkeypath:/etc/security/ldap/krb5.keytab
#krbcmddir:/usr/krb5/bin/
#
# Default-entry location. Valid values are "ldap" and "local".
# The default is "ldap".
# ldap  - use the default entry in LDAP.
# local - use the default entry from /etc/security/user.
#defaultentrylocation: ldap

# Timeout period (seconds) for LDAP client requests to the server. This value
# determines how long the client will wait for a response from the LDAP server.
# Valid range is 0 - 3600 (1 hour).  Default is 60 seconds.  Set this
# value to 0 to disable the timeout.
#ldaptimeout:60
```

Proxyuser

This is the LDAP user account, the alternative of the LDAP administrator account, that a client system uses to bind to the LDAP server. The access right of this account to the LDAP server determines the behavior of the client systems. The **mksecldap** command will support the creation of this account at directory server setup. By default, this proxy user just has limited permissions and needs to be modified if necessary. The default ACL that is set for the proxy user during server setup can be configured by modifying the entries in /etc/security/ldap/proxy.ldif.template. The following shows new options to **mksecldap** command added in Version 5.3.

- For server setup (New options: -x, -X):

```
#mksecldap -s -a adminDN -p adminpasswd -S schematype [ -d baseDN ] [ -n
port ] [ -k SSLkeypath] [ -w SSLkeypasswd ] [ -u  NONE ] [ -U ] [ -x
proxyuserDN ] [ -X proxyuserPasswd ]
```

- For client setup (New options : -A, -D, -m):

```
#mksecldap -c -h serverlist -a bindDN -p bindPasswd [ -A AUTHTYPE ] [ -d
baseDN ] [ -n serverPort ] [ -k SSLKeyPath ] [ -w SSLKeyPasswd ] [ -t
cacheTimeout ] [ -C cachesize ] [ -P NumberofThreads ] [ -T heartBeatInt ]
[ -u userlist ] [ -U ] [ -m searchMode ] [ -D defaultEntryLocation ]
```

The **secldifconv** command was added in Version 5.3 to provide a utility to convert user and group entries between the schema types (AIX, RFC2307, RFC2307AIX) supported by AIX. Example 8-2 shows how to generate output from the LDAP database and convert the data to RFC2307 schema.

```
#secldifconv -S schematype -i infile [ -r ]
```

Example 8-2 Converting AIX schema to RFC2307

```
#To generate the input file for secldifconv
#db2ldif -o ldif.out

#To convert to RFC2307 schema type
#secldifconv -S RFC2307 -i ldif.out > newldif.out

#The converted data could then be added to an LDAP server through the following
command.
#ldif2db -i newldif.out
```

8.2 Full PAM exploitation

Over the recent past the Pluggable Authentication Module (PAM) framework became a standard method of authentication in the industry. PAM provides a means of separating authentication technologies from authentication services, allowing different services to potentially follow independent authentication paths.

The library /usr/lib/libpam.a was provided in AIX 5L Version 5.1 to allow PAM applications the ability to make use of the PAM framework. This solution did not allow existing AIX security services to utilize the PAM framework. Further PAM function was integrated into AIX 5L Version 5.2 through the use of the existing AIX loadable authentication module (LAM) scheme and creation of a PAM AIX LAM module /usr/lib/security/PAM. Users defined to use the PAM AIX LAM module for their registry could then be routed to PAM for authentication, effectively PAM-enabling all existing AIX security services. The module /usr/lib/security/pam_aix was also provided to allow existing PAM applications access to AIX security services through the PAM interface.

The enhancements to the AIX 5L Version 5.3 PAM implementation are focused on PAM application compatibility, better AIX base operating system integration, and improved configuration support.

The PAM implementation in AIX 5L Version 5.3 features the following:

- System-wide specification to use the PAM library for authentication.
- Default configuration file /etc/pam.conf shipped with AIX.
- PAM enablement of native AIX commands by direct calls to the PAM library.
- Additional PAM modules provided in /usr/lib/security.
- Additional control option and new APIs for the PAM library.
- Comprehensive documentation in the AIX standard publications covering the PAM application programming interface (PAM API), the PAM service programming interface (PAM SPI), all PAM modules, and PAM-enabled applications.

The AIX 5L Version 5.3 PAM implementation adheres to the PAM specification as defined in the Open Group standard X/Open Single Sign-on Service (XSSO) and in RFC 86.0.

8.2.1 AIX authentication through PAM library

In AIX 5L prior to Version 5.3 there are two separate paths of PAM integration. One path uses the typical PAM behavior of an application invoking a PAM library call and then correspondingly calling configured modules in /etc/pam.conf. The related process flow is similar to the following:

Application → PAM library → PAM module

The second path is unique to AIX and provides existing applications the ability to make use of PAM with minimal modification through use of the PAM AIX LAM module /usr/lib/security/PAM. This path requires users to be configured to utilize the PAM compound module database (PAMFiles) for their registry and SYSTEM attributes, with the additional restriction that the module used by the service could not be the pam_aix module to avoid loop conditions. The related process flow is similar to the following:

Application → AIX Security Library →

→ PAM AIX loadable authentication module → PAM library → PAM module

AIX 5L Version 5.3 obtains greater PAM compatibility and eliminates the need for the second path of integration by calling the PAM library directly from AIX

authentication applications rather than going through the AIX loadable authentication module framework.

This function requires the configuration of the new system-wide attribute named *auth_type* which determines whether to call the PAM or AIX security library for user authentication tasks. This attribute is configured in the usw stanza of the /etc/security/login.cfg file and has the following characteristics:

- auth_type: Determines whether PAM or the standard AIX authentication mechanism will be used. The two values to which auth_type can be set are:
  - PAM_AUTH - Use the PAM library for user authenticate tasks as configured by the /etc/pam.conf file.
  - STD_AUTH - Use the standard AIX security library for user authentication tasks. This is the default value.

System administrator can use the **chsec** command to enable PAM authentication for all users system wide:

```
# chsec -f /etc/security/login.cfg -s usw -a auth_type=PAM_AUTH
```

Setting auth_type = PAM_AUTH will configure PAM-enabled commands to invoke the PAM library functions directly for authentication rather than use the standard AIX authentication mechanism. This configuration is a run-time decision and does not require a reboot of the system to take affect. Note, users will no longer need their SYSTEM and registry attributes set to use PAM because the configuration is now a system setting.

The **lssec** command will show the current setting for the auth_type parameter:

```
# lssec -f /etc/security/login.cfg -s usw -a auth_type
usw auth_type=PAM_AUTH
```

The following native AIX commands and applications have been changed to recognize the auth_type attribute and to directly call the PAM library if the system is configured to use PAM for authentication:

- **login**
- **rlogind**
- **telnetd**
- **su**
- **passwd**
- **ftpd**
- **rexecd**
- **rshd**

- **snappd**
- **imapd**

Conversation functions were also added to each modified command and application in order to handle prompting and messaging.

In order to enable the PAM exploitation, the PAM stacks for all PAM-enabled services have to be configured in the /etc/pam.conf file. AIX 5L Version 5.3 delivers a pre-configured default PAM configuration file which defines the PAM stacks for all previously listed PAM-enabled applications and commands to closely resemble the behavior of the standard AIX authentication mechanism. The default /etc/pam.conf file on an AIX 5L Version 5.3 system lists the PAM stack definitions by module type. By default, PAM-enabled services not explicitly defined will fail authentication because the OTHER service name entry at the end of each module type stack is configured to use the /usr/lib/security/pam_prohibit module.

The following sections show the PAM stack configurations consolidated by the service name:

login service

```
login auth      required     /usr/lib/security/pam_aix
login account   required     /usr/lib/security/pam_aix
login password  required     /usr/lib/security/pam_aix
login session   required     /usr/lib/security/pam_aix
```

rlogin service

```
rlogin auth      sufficient   /usr/lib/security/pam_rhosts_auth
rlogin auth      required     /usr/lib/security/pam_aix
rlogin account   required     /usr/lib/security/pam_aix
rlogin password  required     /usr/lib/security/pam_aix
rlogin session   required     /usr/lib/security/pam_aix
```

telnet service

```
telnet auth      required     /usr/lib/security/pam_aix
telnet account   required     /usr/lib/security/pam_aix
telnet password  required     /usr/lib/security/pam_aix
telnet session   required     /usr/lib/security/pam_aix
```

su service

```
su auth      sufficient   /usr/lib/security/pam_allowroot
su auth      required     /usr/lib/security/pam_aix
su account   sufficient   /usr/lib/security/pam_allowroot
su account   required     /usr/lib/security/pam_aix
su password  required     /usr/lib/security/pam_aix
su session   required     /usr/lib/security/pam_aix
```

passwd service

```
passwd password  required     /usr/lib/security/pam_aix
```

rsh service

```
rsh auth      required     /usr/lib/security/pam_rhosts_auth
rsh account   required     /usr/lib/security/pam_aix
rsh session   required     /usr/lib/security/pam_aix
```

rexec service

```
rexec auth      required     /usr/lib/security/pam_aix
rexec account   required     /usr/lib/security/pam_aix
rexec session   required     /usr/lib/security/pam_aix
```

ftp service

```
ftp auth      required     /usr/lib/security/pam_aix
ftp account   required     /usr/lib/security/pam_aix
ftp session   required     /usr/lib/security/pam_aix
```

snapp service

```
snapp auth    required        /usr/lib/security/pam_aix
snapp session required        /usr/lib/security/pam_aix
```

imap service

```
imap auth    required        /usr/lib/security/pam_aix
imap session required        /usr/lib/security/pam_aix
```

Stacking is implemented in the configuration file by creating multiple entries for a service with the same module type field. The modules are invoked in the order in which they are listed in the file for a given service, with the final result determined by the control flag field specified for each entry. Valid values for the control flag field are: required, requisite, sufficient, and optional. The *requisite* control option for the PAM library was introduced with AIX 5L Version 5.3 and the corresponding behavior in the stack is as follows:

requisite If a requisite module fails, the PAM framework immediately returns to the application with the error returned by the requisite module and stops processing the module stack. If all requisite and required modules in the stack succeed, then success is returned. If none of the service modules in the stack are designated as required or requisite, then the PAM framework requires that at least one optional or sufficient module succeed.

Most PAM modules accept a debug option to be specified in the module options field of the related entry in the /etc/pam.conf file. Consult the module specification in the AIX product documentation for valid options. Debug syslog messages from the PAM library will be logged as soon as the system

administrator creates an empty /etc/pam_debug file (**touch /etc/pam_debug** command), configured /etc/syslog.conf file to contain the following line:

```
auth.debug      /tmp/syslog_auth.log
```

and refreshes the syslogd daemon configuration (**refresh -s syslogd** command). You can specify any other syslog output file name in /etc/syslog.conf as desired.

Note: The AIX loadable authentication module for PAM /etc/security/security/PAM is no longer the recommended means to provide PAM authentication to native AIX applications. System administrators should use the system-wide configuration parameter auth_type=PAM_AUTH in /etc/security/login.cfg to configure authentication applications to directly access the PAM library for user authentication tasks. This configuration provides full PAM compliancy, enhanced module compatibility, and improved configuration support.

8.2.2 Additional PAM modules

In previous AIX releases, only the pam_aix PAM module was shipped to support the configuration of PAM services. The pam_aix module provides PAM-enabled applications access to AIX security services by providing interfaces that call the equivalent AIX services where they exist.

AIX 5L Version 5.3 adds six new PAM modules in the /usr/lib/security/ directory to simplify PAM configuration and setup:

pam_allow	Always returns PAM_SUCCESS
pam_allowroot	Returns success if the user ID (UID) is 0 (root user)
pam_ckfile	Performs a file check, similar to that done for /etc/nologin
pam_permission	Checks the user name or group name against a control list in a file, similar to the /etc/ftpusers check
pam_prohibit	Always returns a PAM failure code
pam_rhost_auth	Performs rhosts authentication

The modifications and additions made in AIX 5L Version 5.3 allow the pam_aix module to be used by default on the system, thereby providing a fully integrated and pre-configured PAM solution.

The following sections give a brief description of each of the new PAM modules and also provide some information about the pam_aix module for completeness.

pam_aix module

The pam_aix module allows the authentication path to be routed back to the AIX security services from PAM through use of the AIX security library routines. This allows PAM to closely mimic the behavior of many existing applications as well as allowing new applications to use AIX authentication. This module supports all user types (except for PAMfiles) including: local, NIS, NIS+, DCE, and LDAP users. Users configured to use the PAM AIX LAM module (registry=PAMfiles) supplied in Version 5.2 will fail authentication when using pam_aix due to loop detection performed by the /usr/lib/security/PAM module. pam_aix accepts the debug and nowarn options for all SPIs, and additionally accepts try_first_pass and use_first_pass for the pam_sm_authenticate SPI.

AIX 5L Version 5.3 adds one new option use_new_state to the pam_aix module:

use_new_state — AIX builds and maintains state information when authenticating a user. By default, the pam_aix module will use the same state information throughout a PAM session. This can produce results that are correct in terms of AIX authentication but are unexpected within the PAM framework. For example, pam_authenticate requests may fail due to access restrictions. If this behavior is not desired for a given module type, specify the use_new_state option to use new state information for each invocation.

Typical usage for the pam_aix module is as a backup, or OTHER service. This way, if a specific authentication stack is not defined for a service, local AIX authentication is used.

pam_allow module

This module returns PAM_SUCCESS for all invocations. Support for authentication, account management, password, and session module types is provided. Use of this module is intended for administrator testing and for disabling the checks performed by a given PAM module type. This module only accepts debug and nowarn as specified options in the configuration file.

This module should be used with caution and should often only be used for PAM debugging purposes. Placing this module in the PAM stack for a service could potentially grant access to all users.

pam_allowroot

The pam_allowroot module checks the real user ID (UID) under which the PAM application was run. If the UID of the authenticating user is 0 (zero), then it is the root user and PAM_SUCCESS is returned. Otherwise the module returns a failure code. Support for authentication and account management module types

is provided. This module only accepts debug and nowarn as specified options in the configuration file.

The pam_allowroot module only checks the real user ID. Many applications that require root access will set the effective user ID to 0. For this reason, the effective ID is not used in determining whether or not the user executing the authenticating application is a root user.

It is recommended that pam_allowroot be used as sufficient in conjunction with other modules. This allows the root user to bypass the rest of the modules in the stack and for a failure not to impact the result of other authenticating users. An example authentication stack configuration which mimics the historic behavior of the **su** command follows:

```
#
# The PAM configuration for standard su behavior.
#
su auth sufficient /usr/lib/security/pam_allowroot
su auth required   /usr/lib/security/pam_aix
```

pam_ckfile

The pam_ckfile module allows or denies authentication, based on the existence of a file. The file checked for existence can be set with the file module option. If not specified the file defaults to /etc/nologin.

If the specified file exists, only root users (those with a user ID of 0) may authenticate. All other users are denied access for the service, and pam_ckfile will echo the contents (if any) of that file. If the specified file does not exist, the module returns PAM_IGNORE. System administrators should ensure that success or failure of the module stack for a service does not depend solely on the result of this module.

It is recommended that pam_ckfile is used with the required or requisite control flag in conjunction with other modules.

pam_permission

The pam_permission module is an authentication and account-service PAM module that uses an access-control list to determine whether or not to permit or deny authentication requests. The file to use for the control list is configured using a module option and defaults to /etc/ftpusers if not specified.

If the access-control file exists, the pam_permission module will scan the file using the authenticating user name and group(s). The first match will then be

used to determine the result. The general syntax for an entry in the access-control file is as follows:

```
[+|-][@]<name>
```

The optional first character controls whether to allow(+) or deny(-) the request for the user or group specified by <name>. If a + or - is not the first character in an entry, then the value of the found module option determines the behavior.

Preceding a name with an @ symbol designates the entry as a group. Otherwise the entry is used as a user name. The first match found to a user name or group entry is used to determine access.

All spaces in an entry are ignored. Comments may be added to the file using the # character as the first character in the line. Only one entry or comment is allowed per line and the entries are processed one at a time, sequentially, starting at the top of the file.

Using the keyword ALL will match all users. Since the file is parsed sequentially, use of the ALL keyword should be reserved for the end of the file since any entries after it are ignored.

Upon reaching the end of the access-control file, if a match to a user name or group has not been made, the result will be the opposite value of the found module option. For example, if found=prohibit is set and the user is not found within the file, then the result for that user would be allow.

If the specified access control file does not exist, the module will return PAM_IGNORE and have no affect on the module stack. It is not recommended that the overall success or failure of the module stack depend solely on pam_permission.

It is recommended that pam_permission be used with the control flag required or requisite in conjunction with other modules.

pam_prohibit

The pam_prohibit module returns a failure for all PAM module types. If used as a required or requisite module for a service, the stack that this module is incorporated into will always fail. It is recommended that individual services be explicitly configured in /etc/pam.conf and then the pam_prohibit module used for the OTHER service entries. Configuring the system in this way ensures that only known PAM-enabled applications are capable of successfully authenticating users. Following is an example of how to configure the OTHER service keyword in /etc/pam.conf to use the pam_prohibit module:

```
#
# Fail for all PAM services not explicitly configured
```

```
#
OTHER auth     required /usr/lib/security/pam_prohibit
OTHER account  required /usr/lib/security/pam_prohibit
OTHER password required /usr/lib/security/pam_prohibit
OTHER session  required /usr/lib/security/pam_prohibit
```

pam_rhost_auth

The pam_rhosts_auth module provides rhost authentication services similar to the **rlogin**, **rsh**, and **rcp** commands. The module queries the PAM handle for the remote user name, remote host, and the local user name. This information is then compared to the rules in /etc/hosts.equiv and $HOME/.rhosts.

For a typical user, the module first checks /etc/hosts.equiv. If a match is not found for the user name and host name, the module will continue on to check the $HOME/.rhosts file. If a user name and host name match is still not found, the module returns the PAM_AUTH_ERR failure code. Otherwise, the result depends on the first rule found matching the specified user name and host name.

When authenticating to the root user (user with the UID of 0), the first check of the /etc/hosts.equiv file is skipped. Success of the rhosts authentication is based solely on the contents of the root user's $HOME/.rhosts file.

This module requires that a PAM application, before making the call to pam_authenticate, call pam_set_item and at least set the values of PAM_RHOST and PAM_RUSER. If the PAM_USER item is not set, the module will prompt for the user name through the conversation function provided in the PAM handle.

Further description on how rhosts authentication works can be found in the documentation for the ruserok() subroutine. Information regarding the syntax of rhost configuration files can be found in the $HOME/.rhosts or /etc/hosts.equiv files description.

For expected behavior, pam_rhosts_auth should be used as one of the first authentication modules in the stack and designated as sufficient.

```
#
# PAM authentication stack for typical rlogin behavior.
#
rlogin auth sufficient /usr/lib/security/pam_rhosts_auth
rlogin auth required   /usr/lib/security/pam_unix
```

8.2.3 PAM application programming interface changes

Support for the requisite control option in a PAM configuration file was added to the PAM library in AIX 5L Version 5.3.

In addition to the new control option, the PAM library was further enhanced by three new PAM environment APIs in accordance with the PAM specification as defined in the Open Group standard X/Open Single Sign-on Service (XSSO) and in RFC 86.0:

pam_getenv	Retrieve the value of a defined PAM environment variable.
pam_getenvlist	Retrieve a list of all of the defined PAM environment variables and their values.
pam_putenv	Set a PAM environment variable.

8.3 Prioritizing LDAP servers

AIX supports multiple LDAP servers and offers a failover mechanism. When the current server it is talking to becomes unavailable, it automatically talks to the next available server in the list. Since Version 5.3, AIX is also capable of detecting when a higher priority server is back on line, and reconnects to it.

The list of the servers as defined in the /etc/security/ldap/ldap.cfg configuration file is prioritized. The first server in the list has the highest priority. The connection is managed by the secldapclntd daemon that reads the configuration file. An example of a section of the /etc/security/ldap/ldap.cfg file follows:

```
/etc/security/ldap/ldap.cfg:
# Comma separated list of ldap servers this client talks to
ldapservers:ldapsrv1.ibm.com,ldapsrv2.ibm.com
```

8.4 Rsh login control

The classic way of disabling execution of r-commands has been to set the ttys user attribute for specific users:

```
chuser 'ttys=ALL, !RSH' username
```

In AIX 5L Version 5.3, a new user attribute *rcmds* is introduced. This attribute controls the r-command execution. It accepts the following values:

allow	Allows the user to execute r-commands.

deny	Denies the user to execute r-commands. This is the same as “tty=ALL, !RSH”.
hostlogincontrol	Specifies that the ability of remote command execution is under the control of the hostsallowedlogin and hostsdeniedlogin attributes. Those attributes control the user login.

An example of use of the rcmds user attribute follows:

```
s4 # chuser rcmds=hostlogincontrol adrian
s4 # chuser hostsdeniedlogin=server4 adrian
s4 # su - adrian
$ rsh server4 date
3004-306 Remote logins are not allowed for this account.
$ exit
s4 # chuser hostsdeniedlogin='' adrian
s4 # chuser hostsallowedlogin=server4 adrian
s4 # su - adrian
$ rsh server4 date
Thu Jul  8 15:47:59 CDT 2004
```

8.5 chpasswd command

The **chpasswd** command is introduced that allows the user to change the password in non-interactive form, for example from a script.

An example on how to change the password of user adrian to passwd follows:

```
# chpasswd -?
chpasswd: Not a recognized flag: ?
Usage:
        chpasswd [-R load_module] [-e] [ -f flags | -c ]
# chpasswd
adrian:passwd
^D
#
```

8.6 Initial login license limit increased

The license policy at the time of writing is based on per CPU licensing. The per user licensing is irrelevant in this case.

The initial login license limit has been changed to 32,767, which is the largest signed 16-bit number. This change is also included in the later versions of AIX 5L Version 5.2. The previous initial login license limit was set to 2. Changes to this attribute do not affect the system performance.

The login license limit is defined in the maxlogins attribute in the /etc/security/login.cfg file.

8.7 NIS/NIS+ enhancements

The following enhancements have been made in AIX 5L Version 5.3:

- AIX security support of netgroups in LDAP
- Client support for shadow passwords (encrypted passwords in the /etc/security/passwd file)

A popular abstraction known as *netgroups* is used to name sets of users, hosts, and domain names. As a concept they are very useful for accessing files that require permissions. For example, used in /etc/exports, they can specify which hosts may mount certain file systems. Used in /etc/hosts.equiv or in user's .rhosts file, they can grant or revoke login permissions for groups of users or machines.

Prior versions of AIX supported netgroups only through NIS. AIX 5L Version 5.3 has extended this support to LDAP, and any customizable load modules.

Version 5.3 has made access control more secure by tightening the shadow passwords and authentication maps through use of the passwd.adjunct map.

8.8 Support for passwd.adjunct NIS map

The previous versions of AIX do not support the passwd.adjunct map file in NIS. Starting with Version 5.3, if you have a NIS server with the passwd.adjunct file, you are now able to be authenticated through this map file. Example 8-3 shows the steps to test the passwd.adjunct map support.

Example 8-3 Authentication through the passwd.adjunct NIS map file

```
# Changes the current domain name of the system to your domain. (“nis-dom”, for
example)
    # chypdom -B nis-dom

# Configure a NIS client using your NIS server address (nismaster.itso.ibm.com,
in this example)
```

```
    # mkclient -B -S nismaster.itso.ibm.com

# After configuring the NIS client, use ypcat to display the NIS passwd.adjunct
database:
    # ypcat passwd.adjunct.byname
    adjtest:AgRjp3PiYTFHc:::::

# telnet to the NIS client (your AIX box) using
    login_name=adjtest, passwd=adjtest. You should get
    authenticated.
```

9

Clustering technologies

This chapter focuses on the enhancements to the clustering technologies introduced by AIX 5L Version 5.3, namely:

- RSCT support for Micro-Partitioning technology
- Least privilege resource manager
- Cluster Systems Management Version 1.4 for AIX

9.1 RSCT support for Micro-Partitioning technology

The host resource manager is a component of the RSCT that provides the capability to monitor resources that are tied to an individual machine. This is similar to the function of the aixos resource monitor that is available in event management. However, in this component the values to be monitored are implemented within the new resource management model that allows us to extend these capabilities to configure and control these resources in the future.

The types of values that will be provided include those related to the operating system load and status (paging space, memory usage, processor load, and so on). In addition, this resource manager includes the capability to monitor programs for creation, death, and other characteristics.

AIX 5L Version 5.3 uses RSCT version 2.4. The following resource classes are updated in RSCT version 2.4:

- IBM.Host
- IBM.Processor

9.1.1 Updated Host resource class - IBM.Host

The following three persistent resource attributes are added to the IBM.Host class in RSCT version 2.4.0:

NumOnVProcessors Indicates the number of virtual processors currently online in this partition. This attribute is available only on partitions using Micro-Partitioning technology.

EntProcCapacity Indicates the number of processor units this LPAR is entitled to receive. This attribute represents the partition's percentage of shared physical processors. This attribute is available only on partitions using Micro-Partitioning technology.

NumActPProcessors Indicates the current number of physical CPUs in the system that contains this partition. This attribute is available only on partitions using Micro-Partitioning technology.

9.1.2 Updated Processor resource class - IBM.Processors

The following persistent resource attribute is added to the IBM.Processors class in RSCT 2.4.0:

LogicalId Identifies SMT threads ID. On SMT-enabled system, one physical or virtual processor can support multiple thread

contexts and execute instructions for them in parallel, thereby improving the utilization of the functional units and otherwise idle processor cycles due to cache delays or other wait states imposed on an individual thread of execution. This attribute refers to this SMT thread's ID.

9.2 Least privilege resource manager

The *Least privilege resource manager* (LPRM) provides a way through which non-root users and processes can access least privilege commands and executable scripts based on their authenticated identity and authorizations in the access control lists. Least privilege commands/scripts are root-only executables to which a normal user, who otherwise cannot run such commands, is given access in the most restricted sense for certain purposes only. Such a facility reduces the need for users and processes to run with total root authority when, in reality, they need only execute one or two commands that require root authority.

The LPRM is modeled after PSSP's sysctl facility, which grants non-root users and processes access to root commands by wrapping the root commands in sysctl procedures. These procedures, in turn, become the actual commands seen and issued by users or processes. This is accomplished in the LPRM by modeling a resource class, the resources of which are such root only commands. Using CT security services for authentication and RSCT RMC ACL file for authorization users are granted access to these resources.

LPRM functions as an authenticated client server system, enabling remote and parallel execution of root commands. A command line interface, that is the LPRM client, is provided which can perform operations on the commands such as creation, execution, deletion, modification, listing, history of previous commands. The LPRM daemon maintains and manages the resource class and its instances.

LPRM supports IW, Distributed management, and Peer modes of operation. Not only does the LPRM give the most restricted access to root-only commands, it also logs the detailed usage information of such commands using AuditLog. Also, when RMC's Full ACL support is available, LPRM will be able to give least privilege access to users at a command (resource) level.

9.2.1 LPRM commands

The LPRM is related to the rsct.core.lprm LPP file set. This section describes the basic commands of the LPRM environment.

mklpcmd command

The **mklpcmd** command defines a new least-privilege root command or script to the resource monitoring and control (RMC) subsystem.

To define the /mycommand script as the LP1 LPRM command locally on your node, run the **mklpcmd** command as follows:

```
# mklpcmd LP1 /mycommand
```

lslpcmd command

The **lslpcmd** command lists the least-privilege resources on one or all nodes in a domain.

To list the LPRM commands locally on your node with all their attributes, run the **lslpcmd -A** command as follows:

```
# lslpcmd -A
 Name = LP1
 ActivePeerDomain =
 CheckSum = 1723032027
 CommandPath = /mycommand
 ControlFlags = 1
 Description =
 Lock = 0
 NodeNameList = {server4}
```

runlpcmd command

The **runlpcmd** command runs the least-privilege resource.

The following example shows how to run the LP1 command and substitute an argument to it:

```
# cat /mycommand
#!/usr/bin/ksh
echo "$0: Starting my program"
echo "$0: My program says $1"
echo "$0: My program finished"
exit 0

# runlpcmd LP1 "Hello"
Stdout::/mycommand: Starting my program
/mycommand: My program says Hello
/mycommand: My program finished
```

chlpcmd command

The **chlpcmd** command changes the read/write attribute values of a least privilege root command or script.

The following example shows how to use the **chlpcmd** command to change the command path attribute:

```
# chlpcmd LP1 CommandPath=/mycommand2
# lslpcmd -A
 Name = LP1
 ActivePeerDomain =
 CheckSum = 1512195727
 CommandPath = /mycommand2
 ControlFlags = 1
 Description =
 Lock = 0
 NodeNameList = {server4}
# runlpcmd LP1 "Hello"
Stdout::/mycommand2: Starting my program
/mycommand2: My program says Hello
/mycommand2: My program finished
```

lphistory command

The **lphistory** command lists or clears the history of a certain number of least-privilege commands that were previously issued during the current resource monitoring and control (RMC) session.

The following example shows how to use the **lphistory** command to show the history of commands and the **lphistory -c** to clear the history:

```
# lphistory
/mycommand
/mycommand
/mycommand
/mycommand Hello
/mycommand Hello
/mycommand2 Hello
# lphistory -c
```

rmlpcmd command

The **rmlpcmd** command removes one or more least-privilege resources from the resource monitoring and control (RMC) subsystem.

An example of removing the LP1 command from the LPRM environment follows:

```
# rmlpcmd LP1
```

9.2.2 LPRM security

The security to execute or modify the LPRM commands is based on the RMC ACLs. To enable the execution or creation of the commands configured to the LPRM you need to modify the /var/ct/cfg/ctrmc.acls file and edit the IBM.LPCommands class. If it does not exist then create a new stanza. To enable the user to create, modify, and execute the commands defined to the LPRM, add the following line to the IBM.LPCommands stanza:

```
user@LOCALHOST  *  rwx
```

To enable the user only to execute the commands already defined to the LPRM, add the following line to the IBM.LPCommands stanza:

```
user@LOCALHOST  R  x
```

An example that enables user adrian to run the LPRM command, but does not allow this user to create new ones or modify existing ones, follows.

```
# more /var/ct/cfg/ctrmc.acls
...
IBM.LPCommands
        root@LOCALHOST          *       rwx
        adrian@LOCALHOST        R       x

# stopsrc -s ctrmc
# startsrc -s ctrmc
# su - adrian
# su - adrian
$ runlpcmd LP1 "Hello"
Stdout::/mycommand: Starting my program
/mycommand: My program says Hello
/mycommand: My program finished
$ mklpcmd LP2 /mycommand2
2610-418 Permission is denied to access the resources or resource class
specified in this command.
$ exit
# su - guest
$ runlpcmd LP1 Hello
2610-418 Permission is denied to access the resources or resource class
specified in this command.
```

In the example, user adrian started the /mycommand through the LPRM environment and the command is executed under the root user.

9.3 Cluster Systems Management Version 1.4 for AIX

AIX 5L Version 5.3 introduces a new version of the Cluster Systems Management (CSM) software. The version of CSM that comes with AIX 5L Version 5.3 is CSM Version 1.4.

CSM Version 1.4 for AIX requires, and ships with, AIX 5L Version 5.3 or AIX 5L Version 5.2 Maintenance Level 04. If the system is currently at a lower level, then the upgrade procedure described later in this section should be followed.

Note: Prerequisites for CSM are AIX installp file sets and also RPM packages. All prerequisite installp file sets and RPM packages are available on the basic AIX installation CD.

The CSM that comes with the AIX 5L Version 5.3 does not contain the full license of the product. It contains only a time-limited license. The full license key is delivered on separate CD media.

9.3.1 CSM 1.4 new features

CSM Version 1.4 has the following new features:

- Support for AIX 5L Version 5.3.
- HW support of the new POWER5 systems.
- Support for JS20 Blade servers running Linux.
- Enhanced cluster scaling. POWER4 HMC scales to 32 physical systems and 64 operating system images.
- Utilities and documentation for the transition from PSSP to CSM.
- New version of OpenCIMOM is used. Version 0.8-1 is required to support new hardware and is necessary for CSM Version 1.4.
- The **rpower** command is updated with new function.
- Java 1.4.1 is used by CSM as it is the default on AIX 5L Version 5.3.

Hardware management console power method

The **rpower** command is updated to have ability to wait for on, off, reboot, cec_on, cec_off and open_firmware operations to complete before returning. The **rpower -w** will wait for all these commands to complete, but only when used with the newly introduced -w option. The command completes means that it returns when the operation is accepted by the HMC CIMOM, but it does not wait until the node changes state.

In previous versions, the command returned before the operation was accepted by the HMC CIMOM infrastructure and it was even before the CIMOM operation has ever begun. With the new -w option, the command returns when the operation is accepted by the HMC CIMOM.

The **rpower** command executes the operations in parallel for nodes on multiple HMCs.

9.3.2 CSM migrations to Version 1.4

Supported migration to CSM Version 1.4 is from CSM Version 1.3. Customers with lower versions of CSM must upgrade to CSM Version 1.3 at least.

Use the following procedure to upgrade to CSM Version 1.4:

1. Verify that the environment before upgrade corresponds to the requirements.
2. Back up the CSM data. The migration preserves the CSM database, but it is prudent to create a backup, just in case. Run the **csmbackup** to save the CSM environment and archive the file outside the server that is being upgraded.
3. Perform a regular system migration of the AIX as documented in the *AIX 5L Version 5.3 Installation Guide and Reference*, in the section titled “Migration Installation.”

   As part of the migration, all the CSM LPP file sets will be updated. If NIM is configured, all the NIM data and definitions will be preserved.
4. Update the open source prerequisites. The AIX migration does not automatically update the RPM packages. The following CSM prerequisite RPM packages need to be upgraded:
   - tk
   - tcl
   - expect
   - conserver

   The installation packages of the listed packages are available on one of the AIX 5L Version 5.3 installation CD media. For example, you can use the AIX **geninstall** command to install the RPMs as follows:

   ```
   geninstall -laX -d /dev/cd0 R:expect R:tcl R:tk R:conserver
   ```
5. Accept the CSM license. The license key is shipped separately on the CSM license CD media. If you are migrating the operating system, you will be getting a new version of CSM and you will have to accept the new license.

To accept the license and activate the license key, use the mount and **csmconfig** commands as follows:

```
mount -v cdrfs -o ro /dev/cd0 /mnt
csmconfig -L /mnt/csmlum.full
```

At the prompt, follow the instructions to accept the CSM license. Check the success of the **csmconfig** command by running it with no flags, and then checking the output.

6. Copy the CSM files into the /csminstall subdirectories. Use the following command to do this:

   ```
   csmconfig -c
   ```

   These files are primarily used when CSM system management scripts are run on the nodes of the cluster.

7. Verify the installation. To verify that the management server has installed correctly and is ready for use, you can run the ibm.csm.ms probe, which is shipped with CSM. To run the probe issue the following command:

   ```
   probemgr -p ibm.csm.ms -l 0
   ```

8. Migrate the nodes of the cluster. The operating system migration of the cluster nodes is accomplished using standard AIX support. This includes the AIX Network Installation Manager (NIM).

Migrating the CSM management server does not necessarily mean that the nodes must also be migrated. The management server will still be able to manage the nodes even if they are not upgraded, if the nodes are at compatible levels with the CSM management server.

When the nodes are migrated to a new operating system level, keep in mind that there will also be a new level of CSM file sets installed on the nodes. You should therefore update the InstallCSMVersion and InstallDistributionVersion node attributes to be consistent with what is installed. This is done by using the **chnode** command. For example, if you would like to update all the node definitions to have a CSM version of 1.4.0 and an operation system version of 5.3.0 you could issue the following command:

```
chnode -a InstallCSMVersion=1.4.0 InstallDistributionVersion=5.3.0
```

10

National language support

This chapter discusses the enhancements made in AIX 5L Version 5.3 for national language support, including the following topics:

- Gujarati NLS enablement
- Tamil NLS enablement
- Kazakh NLS enablement
- Telegu NLS enablement
- Unicode extension
- Update AGFA TrueType font rasterizer
- Unicode 4.0 support

10.1 Gujarati NLS enablement

AIX 5L Version 5.3 provides full locale support within the base operating system for the Gujarati language as spoken in western India. Hindi is the country's national language spoken by nearly 500 million people. However, there are 17 other officially recognized languages, of which Gujarati is one. The Gujarati script is written from left to right and top to bottom in the same fashion as English. However, the language takes on a greater complexity because of the various ways in which Gujarati characters combine to form connected ligatures and syllables. The rules for rendering Gujarati are very similar to those for Hindi, although some differences exist primarily due to differences in the available glyphs for rendering the language.

10.2 Tamil NLS enablement

AIX 5L Version 5.3 provides full locale support within the base operating system for the Tamil language as spoken in India. Hindi is the country's national language spoken by nearly 500 million people. However, there are 17 other officially recognized languages, of which Tamil is one. Tamil is the main language spoken in the state of Tamil Nadu and the union territory of Pondicherry. Tamil language is a member of the Dravidian/South Indian family of languages. Tamilians alone number about 64 million people. The Tamil script is written from left to right and top to bottom in the same fashion as English. However, proper rendering of the language is somewhat difficult due to the various ways in which Tamil letters can change shape and position depending on their context.

10.3 Kazakh NLS enablement

AIX 5L Version 5.3 provides full locale support within the AIX base operating system for the Kazakh language as spoken in Kazakhstan. Kazakh is the official language and principle native language of the Republic of Kazakhstan. It is also spoken in southern Siberia, northwestern China (Sinkiang-Uighur) and northwestern Mongolia. It is one of the most widely spoken languages in central Asia. An estimated 8 million people speak Kazakh.

10.4 Telegu NLS enablement

AIX 5L Version 5.3 provides full locale support within the base operating system for the Telegu language as spoken in India. Hindi is the country's national language spoken by nearly 500 million people. There are 17 other officially

recognized languages, of which Telegu is one. It is the main language spoken in the state of Andhra Pradesh and is the second most widely spoken language in southern India. Telegu language is a member of the Dravidian/South Indian family of languages. It is written using the Telegu script and is written from left to right and top to bottom in the same fashion as English. However, proper rendering of the language is somewhat difficult due to the various ways in which Telegu letters can change shape and position depending on their context.

10.5 Unicode extension

AIX 5L Version 5.3 provides full locale support within the base operating system for the GB18030-2000 codeset standard. GB 18030-2000 is a new Chinese standard that specifies an extended codepage and a mapping table to Unicode. GB 18030 was first published on March 17, 2000. After feedback from the worldwide software industry, the codepage was changed, and a new mapping table was released.

This codepage standard is important for the software industry because China has mandated that any software application that is released for the Chinese market must support GB 18030.

10.6 Update AGFA TrueType font rasterizer

The UFST (Universal Font Scaling Technology) font rasterizer from AGFA/Monotype has been updated from version 4.4 to version 4.6 for AIX 5L Version 5.3.

UFST version 4.6 includes support for 4-byte font indexing. Previous versions only supported 2-byte font indexing, so were limited to fonts that contain no more than 65536 characters. New international standards such as Chinese GB18030 require thousands of additional characters, thus the bigger font index is needed.

10.7 Unicode 4.0 support

AIX 5L Version 5.3 includes enhancements to the existing Unicode locales in order to bring them up to compliance with the latest published version of the standard, which is the 4.0 version. This version adds additional characters and scripts to the standard, bringing the total number of defined Unicode characters to 96,382. Some of these characters are required in order to achieve support for Japanese Industrial Standard JISX0213.

A

Web-based System Manager enhancements

AIX 5L Version 5.3 supports the following features in Web-based System Manager, most of which are already available with Version 5.2 ML3:

- Now using Java 1.4.2 (Version 5.3) instead of 1.4.1 (Version 5.2 ML3)
- Support for Java Web Start (Version 5.2 ML3)
- Voicing support on Windows (Version 5.2 ML3)
- Improved keyboard navigation (Version 5.2 ML3)
- Support for Windows Native theme on Windows (Version 5.2 ML3)

In addition to these topics, enhancements in the base operating system have been reflected by modifications in Web-based System Manager in the areas of:

- Advanced Accounting
- Partition Load Manager
- NFS Version 4
- SMT, job scheduling, resource sets
- Performance tuning and monitoring
- Configuration assistant
- Remote client for firewall configuration

For a discussion of these topics, consult the product documentation.

A.1 Java Web Start

Users of the Linux or Windows client have the choice of using Java Web Start instead of installing the client using InstallShield. The URL for downloading the remote client is:

`http://<hostname>/remote_client.html`

The page displayed will allow the following choices:

InstallShield	This remote client is installed using an InstallShield wizard and it must be re-installed to obtain updates. This client is useful when running the Web-based System Manager over a broadband connection (cable modem or DSL), because updates to the console are not automatically downloaded.
Java Web Start	This remote client is loaded by Java Web Start, which must be installed on the client system prior to installing the remote client. This version of the remote client will check for updates on the server every time it is invoked and download updates automatically.

A.2 Voicing support on Windows

Installation of the remote client now automatically installs the IBM Accessibility Speech Interface V1.2 and IBM ViaVoice® TTS Runtime V6.740. This technology preview provides for a limited voicing capability of Web-based System Manager. In order to run with Voicing enabled, the startup file **wsmsvk.bat** sets the required environment variables and launches Web-based System Manager.

The startup file wsm.bat that starts Web-based System Manager without voicing turned on and **wsmsvk.bat** are located in the bin sub-directory of the Web-based System Manager install directory (normally C:\Program Files\websm\bin).

The desktop shortcut that was created above launches Web-based System Manager without voicing turned on. You can create a copy of that shortcut and change the startup file to **wsmsvk.bat** if you would like to be able to launch the voicing version of Web-based System Manager using a short-cut.

A.3 Keyboard accessibility

The goal of keyboard accessibility is for the user to be able to use the Web-based System Manager without having to use a mouse (Figure A-1). The following keyboard accessibility features are available:

Menu mnemonics

All menu choices can be selected from the keyboard by typing the letter indicated in the menu title. To open the menu, type the underlined letter while pressing the Alt key on the keyboard. This is true only for opening the menu. Once the menu is open, release the Alt key. For example, to select the Properties option in the Selected menu, open the menu by typing **s** while holding the **Alt key**, then release the Alt key and type **r** to select the Properties option. When using mnemonics of the Web-based System Manager menu bar, be sure to move the mouse cursor into the console frame.

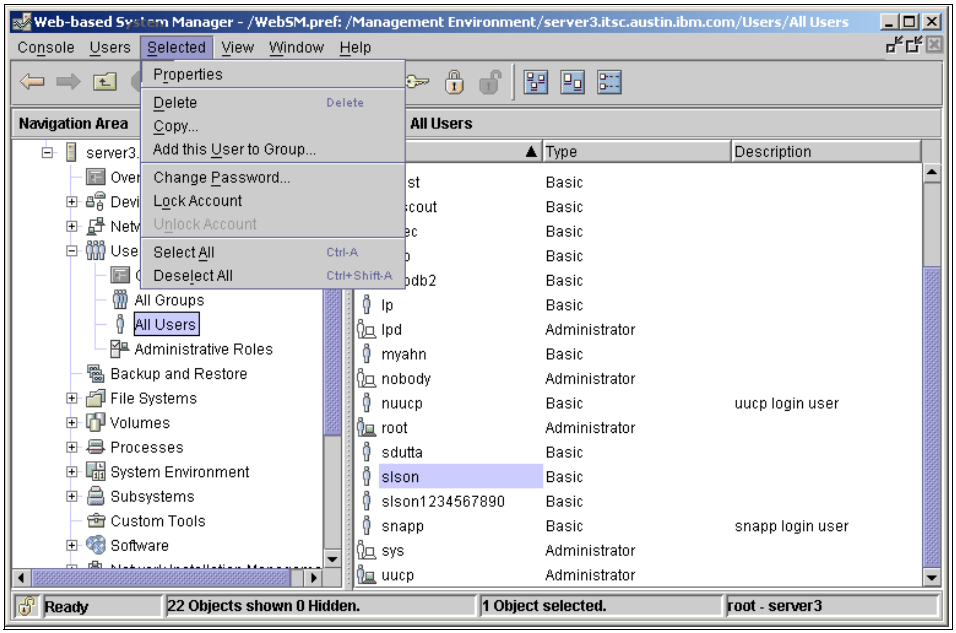

Figure A-1 Improved keyboard navigation in Web-based System Manager

Menu accelerators or shortcut keys

Key combinations are available for common actions, for example, Ctrl + Q to quit and F9 for Key Help.

A.4 Inherit desktop theme on Windows remote client

Web-based System Manager supports the Windows Native theme on the Windows client (Figure A-2).

The Native theme causes Web-based System Manager to inherit a number of desktop theme properties. When you set your theme type to Native and you subsequently run Web-based System Manager on a non-Windows client, your theme will be set to Classic. As long as you do not change your theme (and save your preferences), Web-based System Manager will remember to set the theme back to Native when you run again from a Windows client. If you run an older version of the Web-based System Manager client and save your preferences, then any Native theme setting is lost and will not be remembered the next time you launch Web-based System Manager using that preference file.

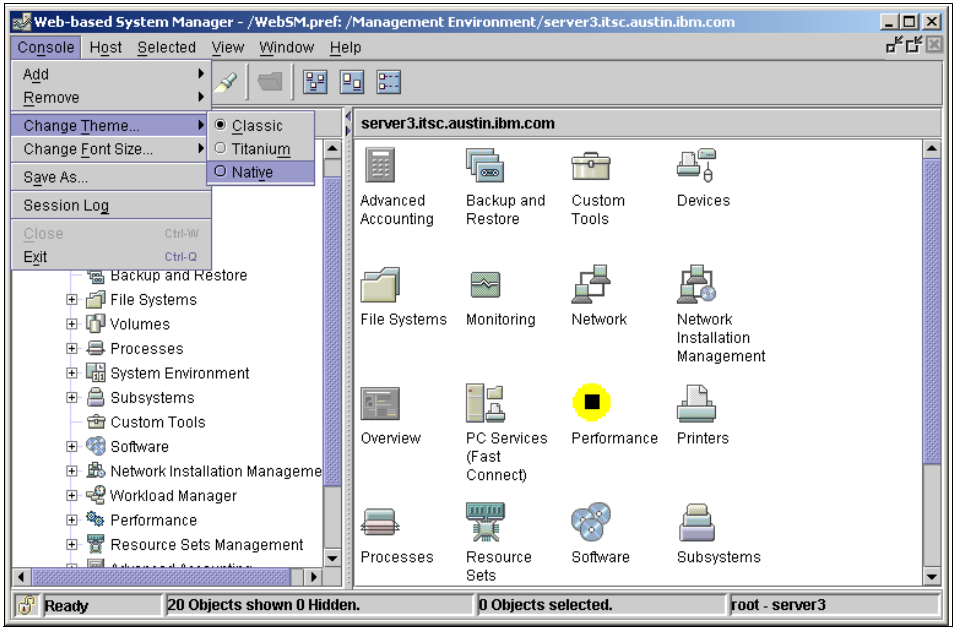

Figure A-2 Native theme support in Web-based System Manager

Abbreviations and acronyms

ABI	Application Binary Interface
AC	Alternating Current
ACL	Access Control List
ADSM	ADSTAR Distributed Storage Manager
ADSTAR	Advanced Storage and Retrieval
AFPA	Adaptive Fast Path Architecture
AFS®	Andrew File System
AH	Authentication Header
AIO	Asynchronous I/O
AIX	Advanced Interactive Executive
ANSI	American National Standards Institute
APAR	Authorized Program Analysis Report
API	Application Programming Interface
AppA	Application Audio
AppV	Application Video
ARP	Address Resolution Protocol
ASCI	Accelerated Strategic Computing Initiative
ASCII	American National Standards Code for Information Interchange
ASR	Address Space Register
ATA	Advanced Technology Attachment
ATM	Asynchronous Transfer Mode
AuditRM	Audit Log Resource Manager
AUI	Attached Unit Interface
AWT	Abstract Window Toolkit
BCT	Branch on CounT
BFF	Backup File Format
BI	Business Intelligence
BIND	Berkeley Internet Name Domain
BIST	Built-In Self-Test
BLAS	Basic Linear Algebra Subprograms
BLOB	Binary Large Object
BLV	Boot Logical Volume
BOOTP	Boot Protocol
BOS	Base Operating System
BPF	Berkeley Packet Filter
BRX	Branch Execution Unit
BSC	Binary Synchronous Communications
BSD	Berkeley Software Distribution
CA	Certificate Authority
CAD	Computer-Aided Design
CAE	Computer-Aided Engineering
CAM	Computer-Aided Manufacturing
CATE	Certified Advanced Technical Expert
CATIA	Computer-Graphics Aided Three-Dimensional Interactive Application
CCM	Common Character Mode
CD	Compact Disk
CDE	Common Desktop Environment
CDLI	Common Data Link Interface
CD-R	CD Recordable

CD-ROM Compact Disk-Read Only Memory

CE Customer Engineer

CEC Central Electronics Complex

CFD Computational Fluid Dynamics

CFM Configuration File Manager

CGE Common Graphics Environment

CHRP Common Hardware Reference Platform

CIM Common Information Model

CISPR International Special Committee on Radio Interference

CIU Core Interface Unit

CLI Command Line Interface

CLIO/S Client Input/Output Sockets

CLVM Concurrent LVM

CMOS Complimentary Metal-Oxide Semiconductor

CMP Certificate Management Protocol

COFF Common Object File Format

COLD Computer Output to Laser Disk

CPU Central Processing Unit

CRC Cyclic Redundancy Check

CRL Certificate Revocation List

CSID Character Set ID

CSM Cluster Systems Management

CSR Customer Service Representative

CSS Communication Subsystems Support

CSU Customer Set-Up

CUoD Capacity Upgrade on Demand

CWS Control Workstation

DAD Duplicate Address Detection

DAS Dual Attach Station

DASD Direct Access Storage Device

DAT Digital Audio Tape

DBCS Double Byte Character Set

DBE Double Buffer Extension

DC Direct Current

DCE Distributed Computing Environment

DCM Dual Chip Module

DCUoD Dynamic Capacity Upgrade on Demand

DDC Display Data Channel

DDS Digital Data Storage

DE Dual-Ended

DES Data Encryption Standard

DFL Divide Float

DFP Dynamic Feedback Protocol

DFS Distributed File System

DGD Dead Gateway Detection

DH Diffie-Hellman

DHCP Dynamic Host Configuration Protocol

DIMM Dual In-Line Memory Module

DIP Direct Insertion Probe

DIT Directory Information Tree

DIVA Digital Inquiry Voice Answer

DLPAR Dynamic LPAR

DLT Digital Linear Tape

DMA Direct Memory Access

DMT Directory Management Tool

DMTF Distributed Management Task Force

DN Distinguished Name

DNLC Dynamic Name Lookup Cache

DNS	Domain Naming System
DOE	Department of Energy
DOI	Domain of Interpretation
DOM	Document Object Model
DOS	Disk Operating System
DPCL	Dynamic Probe Class Library
DRAM	Dynamic Random Access Memory
DRM	Dynamic Reconfiguration Manager
DS	Differentiated Service
DSA	Dynamic Segment Allocation
DSE	Diagnostic System Exerciser
DSMIT	Distributed SMIT
DSU	Data Service Unit
DTD	Document Type Definition
DTE	Data Terminating Equipment
DVD	Digital Versatile Disk
DW	Data Warehouse
DWA	Direct Window Access
EA	Effective Address
EC	Engineering Change
ECC	Error Checking and Correcting
ECN	Explicit Congestion Notification
EEH	Extended Error Handling
EEPROM	Electrically Erasable Programmable Read Only Memory
EFI	Extensible Firmware Interface
EHD	Extended Hardware Drivers
EIA	Electronic Industries Association
EIM	Enterprise Identity Mapping
EISA	Extended Industry Standard Architecture
ELA	Error Log Analysis
ELF	Executable and Linking Format
EMU	European Monetary Union
EOF	End of File
EPOW	Environmental and Power Warning
ERRM	Event Response resource manager
ESID	Effective Segment ID
ESP	Encapsulating Security Payload
ESS	Enterprise Storage Server®
ESSL	Engineering and Scientific Subroutine Library
ETML	Extract, Transformation, Movement, and Loading
F/C	Feature Code
F/W	Fast and Wide
FC	Fibre Channel
FCAL	Fibre Channel Arbitrated Loop
FCC	Federal Communication Commission
FCP	Fibre Channel Protocol
FDDI	Fiber Distributed Data Interface
FDPR	Feedback Directed Program Restructuring
FDX	Full Duplex
FIFO	First In/First Out
FLASH EPROM	Flash Erasable Programmable Read-Only Memory
FLIH	First Level Interrupt Handler
FLOP	Floating Point Operation
FMA	Floating point Multiply Add operation
FP	Fixed Point
FPR	Floating Point Register
FPU	Floating Point Unit

FRCA	Fast Response Cache Architecture
FRU	Field Replaceable Unit
FSRM	File System Resource Manager
FTP	File Transfer Protocol
FTP	File Transfer Protocol
GAI	Graphic Adapter Interface
GAMESS	General Atomic and Molecular Electronic Structure System
GID	Group ID
GPFS	General Parallel File System
GPR	General-Purpose Register
GUI	Graphical User Interface
GUID	Globally Unique Identifier
HACMP	High Availability Cluster Multi Processing
HACWS	High Availability Control Workstation
HBA	Host Bus Adapters
HCON	IBM AIX Host Connection Program/6000
HDX	Half Duplex
HFT	High Function Terminal
HIPPI	High Performance Parallel Interface
HiPS	High Performance Switch
HiPS LC-8	Low-Cost Eight-Port High Performance Switch
HMC	Hardware Management Console
HMT	Hardware Multithreading
HostRM	Host Resource Manager
HP	Hewlett-Packard
HPF	High Performance FORTRAN
HPSSDL	High Performance Supercomputer Systems Development Laboratory
HP-UX	Hewlett-Packard UNIX
HTML	Hyper-text Markup Language
HTTP	Hypertext Transfer Protocol
Hz	Hertz
I/O	Input/Output
I^2C	Inter Integrated-Circuit Communications
IAR	Instruction Address Register
IBM	International Business Machines
ICCCM	Inter-Client Communications Conventions Manual
ICE	Inter-Client Exchange
ICElib	Inter-Client Exchange library
ICMP	Internet Control Message Protocol
ID	Identification
IDE	Integrated Device Electronics
IDL	Interface Definition Language
IDS	Intelligent Decision Server
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IHS	IBM HTTP Server
IHV	Independent Hardware Vendor
IIOP	Internet Inter-ORB Protocol
IJG	Independent JPEG Group
IKE	Internet Key Exchange
ILMI	Integrated Local Management Interface
ILS	International Language Support
IM	Input Method
INRIA	Institut National de Recherche en Informatique et en Automatique

IOCTL	I/O Control
IP	Internetwork Protocol (OSI)
IPL	Initial Program Load
IPSec	IP Security
IrDA	Infrared Data Association (which sets standards for infrared support including protocols for data interchange)
IRQ	Interrupt Request
IS	Integrated Service
ISA	Industry Standard Architecture, Instruction Set Architecture
ISAKMP	Internet Security Association Management Protocol
ISB	Intermediate Switch Board
ISDN	Integrated-Services Digital Network
ISMP	InstallSheild Multi-Platform
ISNO	Interface Specific Network Options
ISO	International Organization for Standardization
ISV	Independent Software Vendor
ITSO	International Technical Support Organization
IXFR	Incremental Zone Transfer
JBOD	Just a Bunch of Disks
JCE	Java Cryptography Extension
JDBC	Java Database Connectivity
JFC	Java Foundation Classes
JFS	Journaled File System
JSSE	Java Secure Sockets Extension
JTAG	Joint Test Action Group
JVMPI	Java Machine Profiling Interface
KDC	Key Distribution Center
L1	Level 1
L2	Level 2
L3	Level 3
LAM	Loadable Authentication Module
LAN	Local Area Network
LANE	Local Area Network Emulation
LAPI	Low-Level Application Programming Interface
LDAP	Lightweight Directory Access Protocol
LDIF	LDAP Directory Interchange Format
LED	Light Emitting Diode
LFD	Load Float Double
LFT	Low Function Terminal
LID	Load ID
LLNL	Lawrence Livermore National Laboratory
LMB	Logical Memory Block
LP	Logical Partition
LPAR	Logical Partition
LP64	Long-Pointer 64
LPI	Lines Per Inch
LPP	Licensed Program Product
LPR/LPD	Line Printer/Line Printer Daemon
LRU	Least Recently Used
LTG	Logical Track Group
LUN	Logical Unit Number
LV	Logical Volume
LVCB	Logical Volume Control Block
LVD	Low Voltage Differential
LVM	Logical Volume Manager
MAP	Maintenance Analysis Procedure

MASS Mathematical Acceleration Subsystem

MAU Multiple Access Unit

MBCS Multi-Byte Character Support

Mbps Megabits Per Second

MBps Megabytes Per Second

MCA Micro Channel® Architecture

MCAD Mechanical Computer-Aided Design

MCM Multichip Module

MDF Managed Object Format

MDI Media Dependent Interface

MES Miscellaneous Equipment Specification

MFLOPS Million of FLoating point Operations Per Second

MII Media Independent Interface

MIB Management Information Base

MIP Mixed-Integer Programming

MLR1 Multi-Channel Linear Recording 1

MMF Multi-Mode Fibre

MODS Memory Overlay Detection Subsystem

MP Multiprocessor

MPC-3 Multimedia PC-3

MPI Message Passing Interface

MPIO Multipath I/O

MPOA Multiprotocol over ATM

MPP Massively Parallel Processing

MPS Mathematical Programming System

MSS Maximum Segment Size

MST Machine State

MTU Maximum Transmission Unit

MWCC Mirror Write Consistency Check

MX Mezzanine Bus

NBC Network Buffer Cache

NCP Network Control Point

ND Neighbor Discovery

NDP Neighbor Discovery Protocol

NDS Novell Directory Services

NFB No Frame Buffer

NFS Network File System

NHRP Next Hop Resolution Protocol

NIM Network Installation Management

NIMOL NIM on Linux

NIS Network Information Service

NL National Language

NLS National Language Support

NT-1 Network Terminator-1

NTF No Trouble Found

NTP Network Time Protocol

NUMA Non-Uniform Memory Access

NUS Numerical Aerodynamic Simulation

NVRAM Non-Volatile Random Access Memory

NWP Numerical Weather Prediction

OACK Option Acknowledgment

OCS Online Customer Support

ODBC Open DataBase Connectivity

ODM Object Data Manager

OEM Original Equipment Manufacturer

OLAP Online Analytical Processing

OLTP Online Transaction Processing

ONC+ Open Network Computing

OOUI Object-Oriented User Interface

OSF	Open Software Foundation, Inc.
OSL	Optimization Subroutine Library
OSLp	Parallel Optimization Subroutine Library
P2SC	POWER2™ Single/Super Chip
PAG	Process Authentication Group
PAM	Pluggable Authentication Mechanism
PAP	Privileged Access Password
PBLAS	Parallel Basic Linear Algebra Subprograms
PCB	Protocol Control Block
PCI	Peripheral Component Interconnect
PDT	Paging Device Table
PDU	Power Distribution Unit
PE	Parallel Environment
PEDB	Parallel Environment Debugging
PEX	PHIGS Extension to X
PFS	Perfect Forward Security
PGID	Process Group ID
PHB	Processor Host Bridges
PHY	Physical Layer
PID	Process ID
PID	Process ID
PIOFS	Parallel Input Output File System
PKCS	Public-Key Cryptography Standards
PKI	Public Key Infrastructure
PKR	Protection Key Registers
PMTU	Path MTU
POE	Parallel Operating Environment
POP	Power-On Password
POSIX	Portable Operating Interface for Computing Environments
POST	Power-On Self-test
POWER	Performance Optimization with Enhanced Risc (Architecture)
PPC	PowerPC®
PPM	Piecewise Parabolic Method
PPP	Point-to-Point Protocol
PREP	PowerPC Reference Platform
PRNG	Pseudo-Random Number Generator
PSE	Portable Streams Environment
PSSP	Parallel System Support Program
PTF	Program Temporary Fix
PTPE	Performance Toolbox Parallel Extensions
PTX®	Performance Toolbox
PV	Physical Volume
PVC	Permanent Virtual Circuit
PVID	Physical Volume Identifier
QMF™	Query Management Facility
QoS	Quality of Service
QP	Quadratic Programming
RAID	Redundant Array of Independent Disks
RAM	Random Access Memory
RAN	Remote Asynchronous Node
RAS	Reliability, Availability, and Serviceability
RDB	Relational DataBase
RDBMS	Relational Database Management System
RDF	Resource Description Framework

RDISC	ICMP Router Discovery
RDN™	Relative Distinguished Name
RDP	Router Discovery Protocol
RFC	Request for Comments
RIO	Remote I/O
RIP	Routing Information Protocol
RIPL	Remote Initial Program Load
RISC	Reduced Instruction-Set Computer
RMC	Resource Monitoring and Control
ROLTP	Relative Online Transaction Processing
RPA	RS/6000 Platform Architecture
RPC	Remote Procedure Call
RPL	Remote Program Loader
RPM	Redhat Package Manager
RSC	RISC Single Chip
RSCT	Reliable Scalable Cluster Technology
RSE	Register Stack Engine
RSVP	Resource Reservation Protocol
RTC	Real-Time Clock
RVSD	Recoverable Virtual Shared Disk
SA	Secure Association
SACK	Selective Acknowledgments
SAN	Storage Area Network
SAR	Solutions Assurance Review
SAS	Single Attach Station
SASL	Simple Authentication and Security Layer
SBCS	Single-Byte Character Support
ScaLAPACK	Scalable Linear Algebra Package
SCB	Segment Control Block
SCSI	Small Computer System Interface
SCSI-SE	SCSI-Single Ended
SDK	Software Development Kit
SDLC	Synchronous Data Link Control
SDR	System Data Repository
SDRAM	Synchronous Dynamic Random Access Memory
SE	Single Ended
SEPBU	Scalable Electrical Power Base Unit
SGI	Silicon Graphics Incorporated
SGID	Set Group ID
SHLAP	Shared Library Assistant Process
SID	Segment ID
SIT	Simple Internet Transition
SKIP	Simple Key Management for IP
SLB	Segment Lookaside Buffer
SLIH	Second Level Interrupt Handler
SLIP	Serial Line Internet Protocol
SLR1	Single-Channel Linear Recording 1
SM	Session Management
SMB	Server Message Block
SMIT	System Management Interface Tool
SMP	Symmetric Multiprocessor
SMS	System Management Services
SNG	Secured Network Gateway
SNIA	Storage Networking Industry Association
SNMP	Simple Network Management Protocol

SOI	Silicon-on-Insulator
SP	IBM RS/6000 Scalable POWER parallel Systems
SP	Service Processor
SPCN	System Power Control Network
SPEC	System Performance Evaluation Cooperative
SPI	Security Parameter Index
SPM	System Performance Measurement
SPOT	Shared Product Object Tree
SPS	SP Switch
SPS-8	Eight-Port SP Switch
SRC	System Resource Controller
SRN	Service Request Number
SSA	Serial Storage Architecture
SSC	System Support Controller
SSL	Secure Socket Layer
STFDU	Store Float Double with Update
STP	Shielded Twisted Pair
SUID	Set User ID
SUP	Software Update Protocol
SVC	Switch Virtual Circuit
SVC	Supervisor or System Call
SWVPD	Software Vital Product Data
SYNC	Synchronization
TCB	Trusted Computing Base
TCE	Translate Control Entry
Tcl	Tool Command Language
TCP/IP	Transmission Control Protocol/Internet Protocol
TCQ	Tagged Command Queuing
TGT	Ticket Granting Ticket
TLB	Translation Lookaside Buffer
TLS	Transport Layer Security
TOS	Type Of Service
TPC	Transaction Processing Council
TPP	Toward Peak Performance
TSE	Text Search Engine
TSE	Text Search Engine
TTL	Time To Live
UCS	Universal Coded Character Set
UDB EEE	Universal Database and Enterprise Extended Edition
UDF	Universal Disk Format
UDI	Uniform Device Interface
UIL	User Interface Language
ULS	Universal Language Support
UNI	Universal Network Interface
UP	Uniprocessor
USB	Universal Serial Bus
USLA	User-Space Loader Assistant
UTF	UCS Transformation Format
UTM	Uniform Transfer Model
UTP	Unshielded Twisted Pair
UUCP	UNIX-to-UNIX Communication Protocol
VACM	View-based Access Control Model
VESA	Video Electronics Standards Association
VFB	Virtual Frame Buffer
VG	Volume Group
VGDA	Volume Group Descriptor Area
VGSA	Volume Group Status Area
VHDCI	Very High Density Cable Interconnect
VIPA	Virtual IP Address
VLAN	Virtual Local Area Network
VMM	Virtual Memory Manager

VP	Virtual Processor
VPD	Vital Product Data
VPN	Virtual Private Network
VSD	Virtual Shared Disk
VSM	Visual System Manager
VSS	Versatile Storage Server™
VT	Visualization Tool
WAN	Wide Area Network
WBEM	Web-based Enterprise Management
WLM	Workload Manager
WTE	Web Traffic Express
XCOFF	Extended Common Object File Format
XIE	X Image Extension
XIM	X Input Method
XKB	X Keyboard Extension
XLF	XL Fortran
XML	Extended Markup Language
XOM	X Output Method
XPM	X Pixmap
XSSO	Open Single Sign-on Service
XTF	Extended Distance Feature
XVFB	X Virtual Frame Buffer

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 374. Note that some of the documents referenced here may be available in softcopy only.

- *Advanced POWER Virtualization on IBM eServer p5 Servers Architecture and Performance Considerations*, SG24-5768 available in December 2004
- *Advanced POWER Virtualization on IBM eServer p5 Servers: Introduction and Basic Configuration*, SG24-7940
- *A Practical Guide for Resource Monitoring and Control*, SG24-6615
- *Linux Applications on pSeries*, SG24-6033
- *Managing AIX Server Farms*, SG24-6606
- *The Complete Partitioning Guide for IBM @server pSeries Servers*, SG24-7039
- *Effective System Management Using the IBM Hardware Management Console for pSeries*, SG24-7038
- *Problem Solving and Troubleshooting in AIX 5L*, SG24-5496
- *Understanding IBM @server pSeries Performance and Sizing*, SG24-4810
- *AIX 5L Workload Manager (WLM)*, SG24-5977
- *AIX Reference for Sun Solaris Administrators*, SG24-6584
- *AIX Version 5.2 Differences Guide*, SG24-2014
- *Introducing VERITAS Foundation Suite for AIX*, SG24-6619

Other publications

These publications are also relevant as further information sources:

- The following types of documentation are located through the Internet at the following URL:

http://www.ibm.com/servers/eserver/pseries/library

- User guides
- System management guides
- Application programmer guides
- All commands reference volumes
- Files reference
- Technical reference volumes used by application programmers

Online resources

These Web sites and URLs are also relevant as further information sources:

- A location to download and explore the Mozilla browser for AIX

  http://www.ibm.com/servers/aix/browsers

- Information on RFCs for Internet standards

  http://www.ietf.org

- How to find product documentation

  http://publib16.boulder.ibm.com/pseries/en_US/infocenter/base/

- What is new with BSD UNIX

  http://www.openbsd.org

- Inoformation on large page support for AIX

  http://www-1.ibm.com/servers/aix/whitepapers/large_page.html

- The OpenGroups home page

  http://www.opengroup.org

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

AIX 5L Differences Guide Version 5.3 Edition

(0.5” spine)
0.475”<->0.875”
250 <-> 459 pages

AIX 5L Differences Guide Version 5.3 Edition

AIX - The industrial strength UNIX operating system

An expert's guide to the new release

AIX 5L Version 5.3 enhancements explained

This IBM Redbook focuses on the differences introduced in AIX 5L Version 5.3 when compared to AIX 5L Version 5.2. It is intended to help system administrators, developers, and users understand these enhancements and evaluate potential benefits in their own environments.

AIX 5L Version 5.3 introduces many new features, including NFS Version 4 and Advanced Accounting, and exploits the advanced capabilities of POWER5 equipped severs, such as Virtual SCSI, Virtual Ethernet SMT, Micro-Partitioning, and others. There are many other enhancements available with AIX 5L Version 5.3, and you can explore them in this redbook.

For customers who are not familiar with the enhancements of AIX 5L through Version 5.2, a companion redbook SG24-5765 is available.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-7463-00 ISBN 0738491535