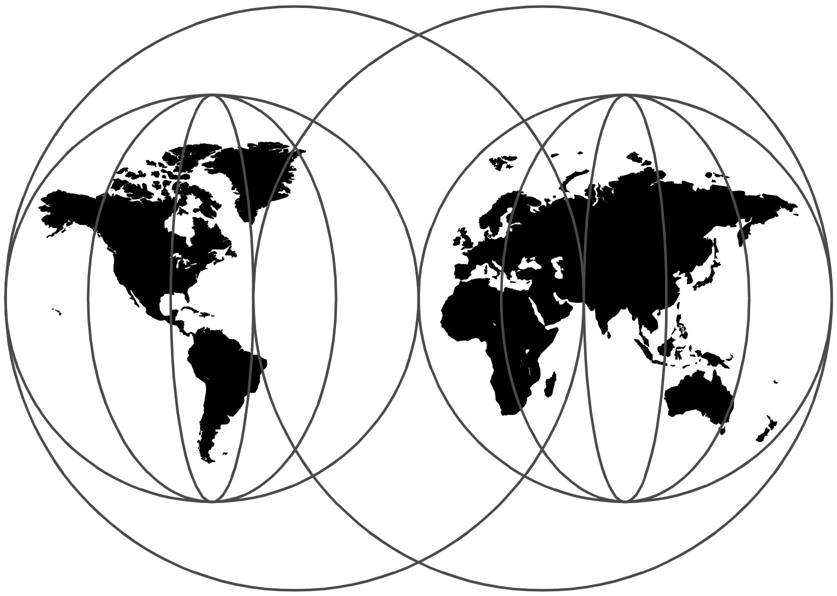
AIX Version 4.3 Differences Guide

Scott Vetter, Atsushi Baba, Robert Iacopetta, Federico Vagnini



International Technical Support Organization

http://www.redbooks.ibm.com

Draft Document for Review October 21, 1999 11:11 pm

SG24-2014-02

SG24-2014-02

International Technical Support Organization

AIX Version 4.3 Differences Guide

December 1999

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix B, “Special Notices” on page 695.

Third Edition (December 1999)

This edition applies to AIX Version 4 Release 3, program number 5765-C34 and subsequent releases.

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

Contents

Figures

Tables

Preface

This redbook focuses on the latest enhancements introduced in AIX Version 4.3 through 4.3.3. It is intended to help system administrators, developers, and users understand these enhancements and evaluate potential benefits in their own environments.

AIX Version 4.3 includes many new features, including 64-bit application support, IP Version 6, X11 Release 6, Lightweight Directory Access Protocol (LDAP), and improved scaling over a wider range of RS/6000 platforms. The availability of two new Web-based utilities, Web-based System Manager and a Web-based Documentation Search Service signal AIX's move towards a standard unified interface for system tools. There are many other enhancements available with AIX Version 4.3, and you can explore them all in this redbook.

This publication is an update to the previously published *AIX Version 4.3 Differences Guide*, Second Edition, which focused on the enhancements introduced in AIX Version 4.3.2. Certain sections of the First Edition and Second Edition have been removed, or edited as required, to reflect the fact that the online documentation provided with AIX Version 4.3 now covers many of the original topics in sufficient depth.

How this Redbook is Organized

Throughout this publication, each major section heading indicates which level of AIX 4.3 introduced the enhancement by including the maintenance level in parentheses. For example, the following section heading:

Multiple Concurrent Reads (4.3.1)

indicates that the feature was introduced in AIX Version 4.3.1. If no maintenance level is given, then the feature was included in the initial AIX Version 4.3.0 offering.

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.

Atsushi Baba is an I/T specialist at the Distribution Industry Systems Engineering lab at IBM Japan. Since he joined IBM in 1991, he has worked mainly as a field engineer for customers in the Travel and Transportation industry. His area of expertise include database, middleware design, Web server implementation, and airline systems.

Robert Iacopetta works in the IBM Software Support Centre in Sydney, Australia providing support for almost every AIX related product. After completing an Engineering degree, he joined IBM just before the RS/6000 was launched in 1990, and has worked with AIX since then. Most of his career is focused in Support Centre positions. He is an author of *AIX Storage Management*, GG24-4484.

Federico Vagnini is an I/T specialist at the Midrange Technical Support in IBM Italy. He joined IBM in 1995 and has four years of experience in the AIX field. He holds an university degree in Electronic Engineering from Politecnico di Milano. His areas of expertise include TCP/IP networking, network security, and SP administration. He has been author of the *Understanding and Using the SP Switch*, SG24-5161.

The authors of the First Edition are:

Colin Fearnley	IBM Johannesburg, South Africa
Andreas Gruber	IBM Munich, Germany
John Hance	IBM Melbourne, Australia
Kevin Murrell	IBM Basingstoke, UK
John Newman	IBM Basingstoke, UK

The authors of the Second Edition are:

Richard Cutler	IBM Austin, USA
Zhu Li	IBM Beijing, China
Armin Olaf Roell	IBM Hamburg, Germany

The project that produced this publication was managed by:

Scott Vetter	IBM Austin

Thanks to the following people for their invaluable contributions to this project. Without their help, this publication would have been impossible:

Andre L. Albot	IBM Austin
Greg Althaus	IBM Austin
Ron Arroyo	IBM Austin
David Babbitt	IBM Austin
Jim Babka	IBM Austin

Richard Belden	IBM Austin
Greg Birgen	IBM Austin
Larry Brenner	IBM Austin
Luke Browning	IBM Austin
Bill Bulko	IBM Austin
Chij-Mehn Chang	IBM Austin
Daisy Chang	IBM Austin
Iliese Chelstowski	IBM Austin
Julie Craft	IBM Austin
Zane Dodson	IBM Austin
Frank Dea	IBM Austin
John Emmons	IBM Austin
Kevin Fought	IBM Austin
Stan Gowen	IBM Austin
Maggie Gretta	IBM Austin
Mark Grubbs	IBM Austin
Lon Hall	IBM Austin
Emilia Hezari	IBM Austin
Dan Hinderliter	IBM Austin
Gary Hook	IBM Austin
Duen-wen Hsiao	IBM Austin
Ida Jackson	IBM Austin
Yoji Kumazawa	IBM Japan
Joy Latten	IBM Austin
Yantian Lu	IBM Austin
James Manon	IBM Austin
Dave Marquardt	IBM Austin
Brandon Mayfield	IBM Austin
Gerald McBrearty	IBM Austin
Mark McConaughy	IBM Austin
Dwayne McConnell	IBM Austin

Casey McCreary	IBM Austin
Hye-Young McCreary	IBM Austin
Bruce Mealey	IBM Austin
James Moody	IBM Austin
Kumar Nallapati	IBM Austin
Steve Nasypany	IBM Austin
Chris Nelson	IBM Austin
Grover Neuman	IBM Austin
Ram Pandiri	IBM Austin
Priya Paul	IBM Austin
Stephen Peckham	IBM Austin
Deanna Quigg	IBM Austin
Tony Ramirez	IBM Austin
Mark D. Rogers	IBM Austin
Ken Rozendal	IBM Austin
Ron Saint Pierre	IBM Austin
Jim Shaffer	IBM Austin
Greg Sharek	IBM Austin
Dave Sheffield	IBM Austin
Jeff A. Smith	IBM Austin
Luc Smolders	IBM Austin
Jeanne Sparlin	IBM Austin
Stephen Stair	IBM Austin
Marc Stephenson	IBM Austin
Masato Suzuki	IBM Japan
Randy Swanberg	IBM Austin
Takayuki Takitani	IBM Japan
Andrew Taylor	IBM Austin
Marvin Toungate	IBM Austin
Arthur Tysor	IBM Austin
Basu Vaidyanathan	IBM Austin

Wayne Wheeler	IBM Austin
Mike Wortman	IBM Austin
Seong Soo Yim	IBM Korea
James Young	IBM Austin
Dalal Younis	IBM Austin

Comments Welcome

Your comments are important to us!

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

- Fax the evaluation form found in “ITSO Redbook Evaluation” on page 729 to the fax number shown on the form.
- Use the online evaluation form found at `http://www.redbooks.ibm.com/`
- Send your comments in an Internet note to `redbook@us.ibm.com`

Chapter 1. Hardware Announcements

The following RS/6000 severs and adapters were companion announcements with AIX Version 4.3.3. AIX is exhaustively tested with every new hardware enhancement.

1.1 RS/6000 7017 Enterprise Server Model S80

The new RS/6000 7017 Model S80 belongs to the IBM's high-end enterprise server family, the S series, that provides the power, capacity, reliability, and growth capacity to help customers move towards the next generation of mission-critical commercial computing. The other members of the S series are the S70 and the S70 Advanced Models.

All S series models are nearly identical in appearance, consisting of a central electronic complex (CEC) and at least one I/O rack. In the CEC are located the CPUs and the central memory, while in the I/O rack are contained the PCI adapters and the disks. The CEC and I/O units are connected by the Remote I/O (RIO) and System Power Control Network (SPCN) cables. A new length cable is available to help reduce clutter.

The I/O racks may contain one or two I/O drawers each, up to a maximum of four drawers per system. The first I/O rack holds the primary I/O drawer, which contains:

- Service processor
- High-performance disk drive
- 32X maximum speed CD-ROM
- 1.44 MB, 3.5-inch diskette drive
- 2 PCI Ultra SCSI controllers

Up to three additional I/O drawers can be added. Each drawer provides fourteen PCI slots across four independent buses. Each drawer may contain two Ultra SCSI or SSA hot-pluggable disk 6-packs. Existing RS/6000 7015 Model R00 and 7014 Model S00 racks can also be used for additional storage and communication drawers.

The Model S80 may be attached to an RS/6000 SP to provide additional online transaction processing (OLTP) and database capability to the SP cluster. It may also be configured for high availability using IBM's High Availability Cluster Multi Processing (HACMP) software and redundant hardware components.

A Model S80 CEC (left) and one I/O rack is shown in Figure 1.

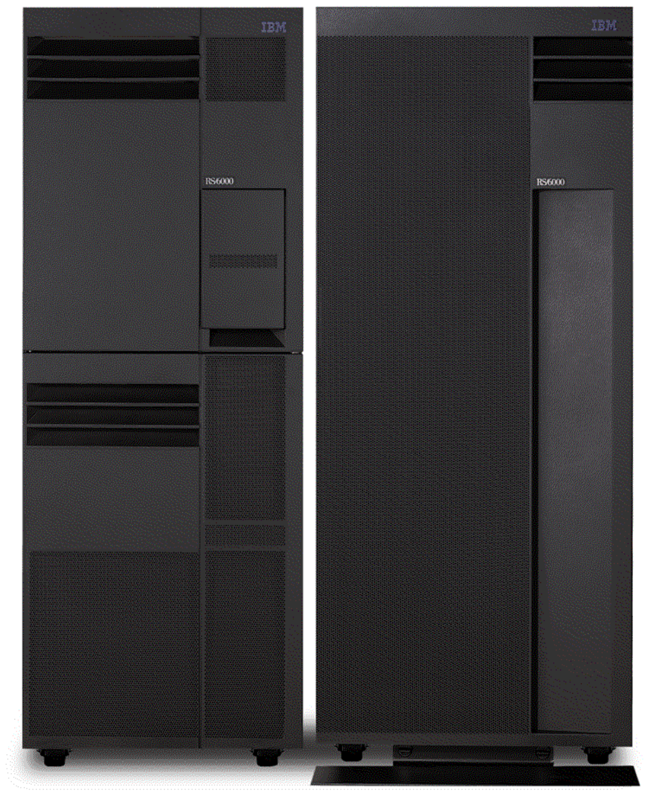


Figure 1. RS/6000 7017 Model S80

The Model S80 incorporates IBM’s state-of-the-art copper chip technology to provide faster, more reliable processors for commercial environments. The base configuration has a 6-way copper technology 450 MHz RS64-III 64 bit processor card. Each processor has 8 MB of L2 cache. Minimum system memory is 2 GB of SDRAM. Cache and system memory is capable of single-bit error correction and double-bit error detection (ECC).

A 6-way system can be expanded to a 12-, 18- or 24-way, with additional processor cards, containing six more processors each. Memory can be increased to 64 GB.

A fully configured system consists of:

- 24 processors
- 64 GB of system memory
- 53 available PCI adapter slots
- 48 hot-pluggable disk bays
- 7 available media bays.

The PCI adapters may be used for a wide range of communications and storage subsystems. Supported communications adapters include Gigabit Ethernet, 10/100 Mbps Ethernet, standard Ethernet, token ring, asynchronous transfer mode (ATM) and fiber distributed data interface (FDDI). Storage device protocols include Ultra SCSI, SSA, and fiber channel arbitrated loop (FCAL).

The Model S80 is shipped and delivered with internal adapters and devices already installed and configured. System power supplies are designed for maximum configurations. The Model S80 requires AIX Version 4.3.3 or later, which comes with every system ordered and which can be preinstalled, if desired.

1.2 RS/6000 7046 Model B50

The IBM RS/6000 7046 Model B50 is a rack mounted server offered at an affordable price. It enables high-density packaging of a variety of applications for the Internet and corporate intranets with a very small footprint, allowing up to 20 servers in one 19-inch rack.

The Model B50 is a uniprocessor system that provides enhanced performance by using a 375 MHz PowerPC 604e processor and an enhanced memory controller. With this memory controller, the Model B50 uses SDRAM memory and has an 83 MHz memory bus speed.

The Model B50 is shown in Figure 2.

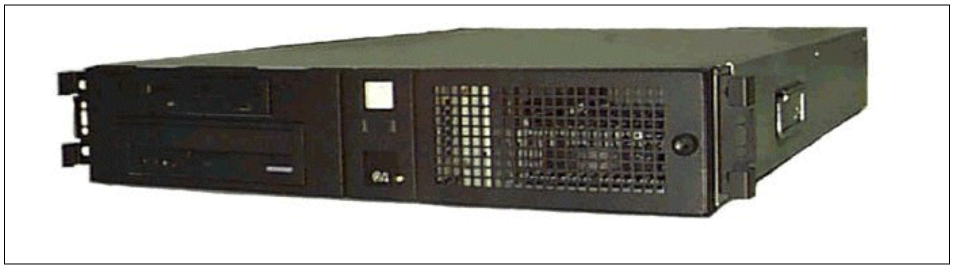


Figure 2. RS/6000 7046 Model B50

The key features of the Model B50 are:

- 375 MHz PowerPC 604e processor
- 1 MB L2 cache
- 128 MB to 1 GB ECC SDRAM memory using four memory slots
- Integrated 10/100 Mbps Ethernet controller (IEEE 802.3 compliant)
- Integrated Ultra SCSI controller
- Two 33 MHz PCI slots, one long and one half size
- Four bays. The two media bays are used by the 1.44 MB 3.5-inch diskette drive and the 32X max speed CD-ROM, and there are two disk bays that can be filled with 9.1 GB or 18.2 GB Ultra SCSI disks
- The GXT130 2D graphics adapter may be optionally installed
- Rack mountable format

For the operating system, the customer can specify to have AIX Version 4.3.3 installed by default, or obtain from the Web the Linux operating system.

1.3 Token Ring PCI Adapter 4/16 Mbps (#4959, 4.3.3)

The IBM Token Ring PCI Adapter for the RS/6000 is a single slot, short, 32-bit, PCI adapter supporting 4 and 16 Mbps data rates, in either half-duplex or full-duplex mode. Automatic ring-speed selection prevents *wrong speed* insertion into the ring, even when connected to speed sensing hubs.

This adapter operates with either unshielded twisted pair (UTP) Cat. 5 cable with RJ-45 connectors or shielded twisted pair (STP) Type 1A cabling with 9-pin D-shell connectors. The adapter provides network boot capability. Diagnostic support has also been provided. The PCI Token Ring device driver is shipped in the devices.pci.14103e00 fileset. This adapter provides the following:

- Can be obtained by specifying feature code 4959
- Autodetects connection types at all speeds
- Meets PCI 2.1 specification
- Fits in PCI Half size slots
- Operates in 64 bit PCI slots in 32 bit mode
- Operates at PCI bus speed from 16 MHz to 33 MHz
- Supports full duplex LAN operation at all speeds
- Consumes less than 2 W of power
- Includes adapter and ring status LEDs
- Supports field update of FLASH EEPROM
- Offers on card diagnostics in microcode
- Is FCC Class B and CISPR Class B certified for STP and UTP cabling
- Supports NIM installations

1.4 Dual Channel Ultra2 SCSI Adapter (#6205)

The PCI Dual Channel Ultra2 SCSI adapter is a 64-bit adapter and is an excellent solution for high performance SCSI applications. It provides two SCSI channels, each of them can either be internal or external and it supports a data rate of up to 80 MB/s, twice the maximum data transfer rate of previous Ultra SCSI adapters.

In order to achieve an Ultra2 SCSI bus data rate of up to 80 MB/s and also maintain a reasonable drive distance, the adapter utilizes Low Voltage Differential (LVD) drivers and receivers. In order to utilize this performance, all attaching devices or subsystems must also be Ultra2 LVD devices. If any device is not Ultra2 LVD, the adapter switches its SCSI bus to single-ended (SE) performance and its interface at the lower SE SCSI bus data rate of the device.

Two industry standard very high density cable interconnect (VHDCI) 68-pin connectors are mounted on the adapter's end bracket allowing attachment of various LVD and SE external subsystems. When using LVD drivers and receivers, drive distance from adapter may be up to 20 meters.

Any supported RS/6000 system can be set up to boot from the PCI Dual Channel Ultra2 SCSI Adapter. Running AIX 4.3.3 or later software, the boot support is part of AIX software. If the system uses previous AIX version, the operating system must be loaded the first time using Network Install Manager (NIM); the following boots will be made from the disk.

1.5 PCI 3-Channel Ultra2 SCSI RAID Adapter (#2494)

The RS/6000 PCI 3-Channel Ultra2 SCSI RAID Adapter (#2494) is a non bootable high-performance Ultra2 SCSI RAID Adapter providing RAID 0, 1 or 5 capability and can address up to forty five 16 bit SCSI2 physical disk drives on three independent SCSI buses. This adapter has one internal and two external independent Ultra2 SCSI buses. This adapter also supports connectivity to Ultra2 LVD devices.

Upgrading an existing PCI 3-Channel Ultra SCSI Adapter to the later version PCI 3-Channel Ultra2 SCSI RAID adapter is also available.

1.6 GXT130P Graphics Accelerator (#2830)

GXT130P Graphics Accelerator is an entry 2D graphics adapter that has been previously. AIX 4.3.3 introduces support for the adapter on following additional RS/6000 system models:

- 7013 Models S70 and S7A
- 7015 Models S70 and S7A
- 7017 Models S70 and S7A
- 7026 Models H10, H50, and H70

Chapter 2. AIX Kernel Enhancements

This chapter examines the changes in the AIX base kernel that are new with AIX Version 4.3.

2.1 Binary Compatibility

The AIX architecture and development teams place a very high priority on ensuring that binary compatibility exists for customers who want to migrate their applications the latest versions of AIX. The following sections explain the extent of this compatibility and the few areas where problems may arise.

2.1.1 Compatibility between AIX Version 4 Releases

Applications written using earlier releases of AIX Version 4 (Release 1 or Release 2) for RS/6000 POWER, POWER2, POWER3, and PowerPC-based models, can be executed on AIX Version 4 Release 3 without recompilation for the same and newer models in that processor family (POWER, POWER2, POWER3, or PowerPC). The exceptions to this statement are applications using:

- Non-shared compiles of AIX shared libraries
- Features explicitly described as non-portable by IBM in the AIX Version 4 reference manuals
- Undocumented AIX internal features
- X11R5 server extensions (AIX Version 4.3 Only)
- Applications compiled using POWER2-, POWER3-, or PowerPC-specific compiler options but executed on models other than POWER2, POWER3, or PowerPC.

Note: Applications compiled on a given release level of AIX Version 4 may not operate properly on systems running an earlier release of AIX Version 4.

Any program intended to run in all environments, POWER, POWER2, POWER3, and PowerPC (601 and newer PowerPC processors), *must* be compiled using the common mode option of the compiler. Programs compiled to exploit POWER2 technology must be run on POWER2-based processors. Programs compiled to exploit POWER3 technology must be run on POWER3-based processors. Programs compiled to exploit PowerPC-based technology must be run on PowerPC-based processors. Existing binaries do not need to be recompiled to operate on the target processors.

64-bit applications produced using AIX Version 4 Release 3 on any of the 32-bit processor models, or the 64-bit processor models, will execute without recompilation on the 64-bit processor models. 32-bit applications produced using AIX Version 4 Release 3 on either 32- or 64-bit processor models will execute without recompilation on both models.

2.1.2 X11 Compatibility

The AIX 4.3 X-server has been upgraded to the X Consortium Release 6 version of X (commonly known as X11R6). The libraries shipped by IBM with X11R6 are backward-compatible, and the client applications that access these libraries will work as on previous releases of AIX. As on earlier releases of AIX, IBM will also ship X11R3, X11R4, and X11R5 compatibility installation options for maximum customer flexibility. In this way, client applications will experience no problems with compatibility.

The majority of applications using X fall into this category and will not have any difficulties. However, a small number of X applications use the loadable extension facility provided by the X server.

The X-server allows for the addition of new function through its extension mechanism. For each extension, part of the extension is loaded into the X-server before it can be executed. X11R6 has modified this mechanism, and it is this part of the extension that must be made compatible with X11R6 to execute properly. All extensions supplied by IBM have been made compatible and will execute properly. In some circumstances, a customer may have an extension that will not work with X11R6, such as:

- Customers who have sample extensions downloaded from the X Consortium FTP site.
- Customers who have developed their own extensions.
- Customers using third-party extensions.

In these cases, the extensions must be made compatible with X11R6 before they will execute properly. Customer-developed extensions and sample X consortium extensions will need to be recompiled with the X11R6 environment. For third-party extensions, the customer should contact the vendor for a X11R6-compatible update.

Customers using non-IBM display adapters may also be using vendor-supplied software specific to those devices that use X-server capabilities. If so, this software must be made compatible with X11R6 to operate properly. The customer should contact the vendor of the display adapter for this software.

IBM provides a porting guide with AIX Version 4.3 that also appears on *The Developers Connection* CD to assist customers and vendors developing adapters or extensions for AIX. *The Developers Connection* can be found at the following URL:

`http://www.developer.ibm.com/devcon/`

2.1.3 AIX Version 3 Application Compatibility

All AIX applications correctly written using AIX Version 3 Release 2 or greater for POWER, POWER2, and PowerPC-based models will run on AIX Version 4 without recompilation for the same models. Exceptions are applications that use:

- Their own loadable kernel extensions
- Certain High Function Terminal (HFT) control interfaces
- X11R3 input device interfaces
- The CIO LAN device driver interface
- SCSI device configuration methods (IHVs)
- The nlist() interface
- DCE threads

Other exceptions include applications compiled using POWER2 or PowerPC-specific compiler options that run on models other than POWER2 or PowerPC.

Any program designed to run in all environments, that is, POWER, POWER2 and PowerPC (601 and above), *must* be compiled using the common mode option of the compiler. Programs compiled to exploit POWER2 technology must be run on POWER2-based processors. Programs compiled to exploit PowerPC-based technology must be run on PowerPC-based processors. Existing code does not need to be recompiled to run.

Note: Applications created on a system using AIX Version 4 may not function properly on a system using AIX Version 3.

For these statements to apply, applications must have been created using the AIX shared libraries.

2.1.4 Client/Server Application Compatibility

An RS/6000 system using AIX Version 3 Release 2 or greater can operate as a server system for client machines using AIX Version 4 with the following exceptions:

- Service of Version 4 diskless/dataless machines
- Network install of Version 4 clients
- Service SNA or X.25 to Version 4 clients
- Service HCON to Version 4 clients
- Service CGE extensions of PEX and PEX-PHIGS
- Use of AIX Version 4 client install formats

An AIX system using AIX Version 4 may operate as a server system for client machines using AIX Version 3 Release 2 or greater as long as the proper compatibility options are installed. All statements about binary compatibility apply in this case. Version 4 applications may not execute properly on Version 3 systems using remote network mounts of file systems.

In both cases, minor administrative changes must be made to the AIX Version 3 systems to support the new AIX Version 4 LFT terminal type.

2.1.5 IBM Licensed Program Products Compatibility

There are hundreds of Licensed Program Products (LPP) available for AIX Version 4. IBM LPPs currently sold for AIX Version 4 Release 1 or Release 2 will operate without change on AIX Version 4 Release 3 with certain exceptions, such as newer versions or releases of a product being available. For information about a specific LPP can be found at the following URL.

```
http://www.ibm.com/servers/aix/products/ibmsw/
```

This site contains information about latest LPP version levels, support, and AIX release compatibility.

For AIX systems using AIX Version 3 Release 2 or greater needing to migrate to AIX Version 4, the publication *A Holistic Approach to AIX V4 Migration, Planning Guide*, SG24-4651, contains information about LPPs and AIX release compatibility.

2.2 AIX 4.3 for 24-Way SMP Performance (4.3.3)

AIX Version 4.3 has evolved to provide better scalability and increased performance. Certain kernel areas identified by the AIX performance team were investigated, tuned, and redesigned, when necessary, to eliminate situations that impede SMP performance.

SMP support is enhanced to support up to 24-way processor configurations in AIX 4.3.3. A list of AIX versions and their supported number of CPUs are provided in Table 1.

Table 1. Maximum Supported Number of CPU

AIX Version	Maximum CPUs supported
AIX 4.1.5	8 CPU
AIX 4.3.0	12 CPU
AIX 4.3.3	24 CPU

2.3 64 GB Real Memory Support (4.3.3)

AIX now supports up to 64 GB of real memory and has been enabled to support larger memory sizes as hardware grows in capacity.

Common Hardware Reference Platform (CHRP) and later RS/6000 Platform Architecture (RPA) are the system architecture bases for systems with large physical memory or any memory above 32-bit real addresses. A list of AIX versions and their supported maximum memories are provided in Table 2.

Table 2. Maximum Supported Memory Sizes

AIX Version	Maximum memory supported
AIX 4.1.5	2 GB
AIX 4.2.1	4 GB
AIX 4.3.0	16 GB
AIX 4.3.2	32 GB
AIX 4.3.3	64 GB

2.4 Real Memory Driver Design (4.3.3)

A new driver /dev/pmem has been added in AIX 4.3.3 to access *real* memory. It is a requirement for the new command kdb in order to support VMM debugging and to read memory without interfering with VMM. Only a root user is allowed to access the driver and this access is limited to read-only mode.

The /dev/pmem special file provides access to the real memory address space, as it is seen by the kernel. The seek offset, set by the lseek() subroutine, is used to specify the real address targeted for the read. Only the open(), close(), read(), and readx() subroutines are supported. Before issuing a read operation, the lseek() subroutine must be used to designate the relevant starting address in real memory.

The lseek() subroutine limits the offset to OFF_MAX (2 GB). The llseek() and lseek64() subroutine limits the offset to DEV_OFF_MAX (1 TB). If the address is within the first terabyte of real memory, then the read() subroutine call can be used. If memory addresses higher than 1 TB are to be accessed, the readx() form of the subroutine call must be used. In this case, the ext (extension) parameter must be set to specify which terabyte range of real memory is referenced. This causes the lseek offset to be interpreted relative to this terabyte range of real memory. No more than 2^{64} bytes of real memory can be accessed.

2.5 Lock-Based Dumping

In AIX Version 4.1 and 4.2, the AIX dump routines always dumped the same data areas. This generic policy meant that certain key data areas were kept out of system dumps because their inclusion would greatly increase the size of the dump. For AIX Version 4.3, dump routines have been added that gather additional information for inclusion in a dump (based on the status of certain locks or flags in the kernel) when the system dump is initiated.

If a lock protecting a structure is held at the time of the dump, then almost certainly, that structure must have been in the process of being updated and should be included in the dump. The primary area where this information is of use is in the Virtual Memory Manager (VMM).

With these additional routines, the need to inconvenience customers with debug kernels or reproduce the problem on test systems with the kernel debugger is greatly reduced.

2.5.1 Dump Support

In AIX Version 4.3 you can use the `dump` interface, through a dump table, to dump specified memory using a real address without requiring the real address to have virtual address mapping.

To support dumping real memory using a real address, a new data structure and a new magic number (`DMP_MAGIC_REAL`) has been defined. The following are modified to check for the magic number and handle the new table format:

- savecore.c
- copydump.c

savecore() and copycore() check for both DMP_MAGIC and DMP_MAGIC_REAL and are able to process either of the dump table formats.

When displaying the data for a dump table, `crash` and `dmpfmt` use an address format that distinguishes real addresses from virtual addresses. A real address can be entered on a `crash` subcommand as *r:address*. For example:

```
> od r:10012
```

When `dump` is initialized, it allocates one page from the pinned heap. It also gets the real address for this page. When dumping memory referenced by a virtual address, dump will perform the following steps for each page (or page segment) to be copied:

1. Turn data translation off
2. Copy the data to the buffer at the real address
3. Turn data translation back on
4. Dump the data

This process retrieves the data in real mode while calling the device driver code in virtual mode.

Note: Only one page will be dumped at a time.

2.5.2 Programming Interface

The external dump interfaces are found in /usr/include/sys/dump.h, and new structures have been defined in this file. In the dumpinfo structure, dm_hostip becomes __ulong32_t. The structures dump_read and dumpio_stat are defined for the kernel and extensions only.

2.6 Bad Block Relocation during System Dump (4.3.1)

In previous versions of AIX, if the LVM detected a bad block and received an I/O error while processing a system dump, the dump was ended if no secondary dump device was available. In AIX Version 4.3.1, the LVM will now try to relocate the bad block so that processing of the system dump can continue and information is not lost.

2.7 Kernel Protection

Memory overlays are extremely destructive and, in certain cases, can destroy the kernel’s ability to respond to interrupts, making it impossible to obtain a dump. A destructive overlay can be caught when it occurs if the kernel code is protected from overwrites. The AIX kernel has therefore been enhanced to provide some protection against these types of errors. The first page of memory is now protected from writes by setting page protection bits in virtual page tables. A similar scheme has been implemented for other pages in the kernel that contain nothing but code (since code should never be altered). Any attempt to overlay protected pages now results in dumps that point directly to the program that tried to do the overwriting. This cuts out the most expensive and time-consuming part of memory overlay debugging for a large number of overlay cases.

For kernel text, enough symbol information has been added to the kernel space so that the kernel text is protected during system initialization. Note that pages containing a mixture of data and text, or data only, cannot be protected, so some kernel text remains writable.

Kernel extension text areas are optionally protected. A run-time check enables the system loader to protect kernel extension text areas. If xmdbg is set by the `bosdebug` or `bosboot` commands, text pages are protected at kernel extension load time. Pages that share text and data are not protected.

Note: This change has impacted kernel and kernel extension code that attempts to modify text. Self-modifying kernel extensions will cause the system to crash unless those extensions also modify the protection of the text pages.

This design protects as many pages in the kernel space as is practical without extensive kernel modifications or increasing the working set needed to run the kernel.

2.7.1 Storage Protection Macro

The STORE_PROTECT macro has been added to store-protect whole pages that reside between two symbols (x and y). This macro is defined as follows:

```
#define STORE_PROTECT(x,y) if (STARTOFPAGE(y) > NEXTPAGE(x))  \
        vm_protect(NEXTPAGE(x),STARTOFPAGE(y)-NEXTPAGE(x),RDONLY)
```

The STORE_PROTECT macro has the effect of protecting all pages starting with the next page boundary beyond x to the last page boundary before y. This macro is used during system initialization for the various regions in the kernel and conditionally by the loader during kernel extension load time.

During system initialization, k_protect() is called to protect the regions marked by the bind steps. k_protect() is called from main() in the following sequence:

```
debugger init();    /* start the kernel debugger */
kmem init();        /* initialize kernel memory heaps */
k_protect();        /* store protect kernel text areas */
strtdisp();         /* start up the dispatcher */
```

When called, k_protect() does the following:

- Store protects low.o areas, at least the first three pages
- Store protects pinned text sub binds
- Store protects paged text sub binds

2.7.2 Debug Modifications

Since the debugger cannot store to some areas in virtual mode due to kernel protection, the debugger has been altered so that all stores to virtual memory addresses are first translated and then performed in real mode. This operation is transparent to the debug user. It requires a modification to the get_put_data_aligned() routine so that virtual operations are translated and performed in real mode. I/O space has not been affected.

2.7.3 Stack Overflow Protection

A stack overflow detection mechanism has been implemented. i_poll() and i_poll_soft() check the MST save-area located lower in memory to see if the csa_prev values, that would be used if the interrupt is interrupted, are valid. If this location contains incorrect data, it is repaired if possible, and the code logs an error.

2.8 SMP TTY Handling

Currently on J30 and J40 SMP systems equipped with 128-port adapters, when the system is under load, CPU 0 spends a great deal of time off-level polling the various 128-port adapters for incoming events.

To alleviate this problem, instead of CPU 0 being used as a timer handler, the load has been passed to other CPUs that are available, thereby improving the overall SMP performance.

2.9 Faster Per-Thread Data

In previous versions of AIX, all threads shared an identical address space. When per-thread data needed to be accessed, a fairly expensive lookup had to be performed by the get_thread_specific() routine.

In a non-threaded version of the OpenGL API (which is very call intensive), tests show that you can expect to spend roughly 150 cycles per call (on average) in a routine. Using the existing get-thread-specific() routine would add approximately 70 cycles (or 50 percent overhead) to enable a multithreaded OpenGL API. A much faster mechanism to access per-thread data for 32-bit systems is therefore required. For 32-bit systems, a separate segment 0 for each processor is now provided. This segment contains a page of thread-specific data that is modified as each thread is swapped in. Faster access to private memory should also provide benefits to the thread libraries.

2.10 Expanded Limits on Open Files (4.3.1)

In previous versions of AIX, a single process was limited to a maximum of 2000 open files at any one time. There was also a total system-wide limit of 200,000 open files. This number was entirely arbitrary, and although it was perfectly adequate for most processes, it was not enough for some. AIX Version 4.3.1 increases these limits to the following.

- Maximum of 1,048,576 open files system wide.
- Maximum of 32,767 open files per process.

The maximum number of file descriptors per process is defined by the constant OPEN_MAX. In AIX 4.3.1 it is 32767.

However, this change can create certain compatibility problems with programs that were compiled with the old OPEN_MAX value of 2000. So there must be a way to enforce the old OPEN_MAX value for existing programs, yet allow new programs to exploit the new capability. This has been done with the existing resource limit functions. There was already a limit for number of available file descriptors, but it has always been set to RLIM_INFINITY. In AIX 4.3.1, the setrlimit() and getrlimit() system calls can be used to maintain specific values for RLIMIT_NOFILE. By default, the soft limit will be the old value of OPEN_MAX, 2000. The default and maximum hard limit will be the new OPEN_MAX value, 32767. With these limits, everything should continue to work as before with no user intervention. If a user increases the soft limit, then programs written to exploit the increased table size can be used.

In addition to the system calls for managing limits, the user can change their limit for number of file descriptors with the `ulimit -n` command. Because the hard limit is set to OPEN_MAX, any user can increase the limit. Privileged access is not required.

2.11 Multiple Concurrent JFS Reads (4.3.1)

AIX uses a simple lock type to serialize access to the in-core inode of a file or directory on a JFS. This is to ensure data integrity, particularly on MP systems, where multiple threads can be accessing an inode simultaneously. When reading a file, the lock is used to serialize updates to the last access time stamp in the inode. This lock has been identified as a potential performance bottleneck in the situation where multiple threads are attempting to read the same file, particularly when migrating from UP to MP systems.

This type of problem affects customers who use databases on a JFS and do not have a choice because their database application does not support raw partitions. Examples include Progress, and Universe, to name two. There are also some large customers who use a JFS for their databases.

The problem stems from the length of time the lock is held. A thread would obtain the lock and then initiate the I/O to read the required data before updating the access time field in the inode and releasing the lock. During this time, other threads would be blocked from accessing the file.

To alleviate this problem, AIX 4.3.1 has changed the mechanism for reads from a JFS to minimize the length of time the inode lock is held by a thread.

When only one thread is reading a file, no change has been made. The reading thread obtains the inode lock and sets a flag in the inode to indicate that a read operation is in place. When the I/O for the read is complete, the thread updates the access time field in the inode and releases the lock.

When a thread attempts to get the inode lock and determines that a read is in progress, instead of blocking on the inode lock, it calls a kernel service to pre-read the data it requires. Once the pre-read has completed, the thread will attempt to obtain the inode lock to update the access time field. This solution reduces the amount of time a thread is required to hold the inode lock when performing a read operation, therefore allowing greater throughput on multiple concurrent reads.

If a thread attempting to read from a file cannot get the inode lock and there is a write operation in progress, then the thread blocks on the lock waiting for the write operation to complete.

2.12 Increase in the Upper Limit of Trace Buffer (4.3.1)

The current upper limit for a trace buffer produced by the `trace` command with -T option is around 55 MB on SMP systems. This only allows a few seconds of a performance benchmark execution to be recorded. As a consequence, only a fraction of the data needed can be collected, since the benchmarks can take up to several minutes.

For AIX Version 4.3.1, the upper limit is increased to the size of a segment and two segments when double buffering is used or as close to that as possible, allowing the amount of information collected to be more complete and useful.

2.13 Kernel Scaling Enhancements

With increasing demands being placed on machines acting as busy network servers, it is possible that in certain situations some kernel resources may become exhausted. As machines supporting larger amounts of physical memory, adapters, and devices are introduced, it makes sense for the kernel to be able to use larger resource pools when required. Therefore, the `crash` utility, used for examining system images and the running kernel, is enhanced to understand the new increased system resources.

The following sections describe the major enhancements.

2.13.1 Network Memory Buffer Pool (4.3.2)

The kernel allocates memory from the network memory buffer pool, commonly called the *mbuf* pool, to be used as buffers by the networking subsystem. The size of the mbuf pool is a tunable parameter and is changed using the `thewall` option of the `no` command.

2.13.1.1 Network Memory Buffer Pool Size Increase

The maximum size of the mbuf pool is now hardware dependent. Previous versions of AIX allocated the mbuf pool from the kernel heap. AIX 4.3.2 now uses a dedicated memory segment for the mbuf pool on most machines, and four contiguous memory segments on CHRP machines. This allows a maximum mbuf pool of 256 MB on most machines and 1 GB on CHRP hardware. The kernel will allocate an amount of virtual memory equal to one half the amount of physical memory, or the maximum value allowed for the hardware type, whichever is smaller. For example, on a machine with 128 MB of memory, the default value of `thewall` will be 64 MB. On a CHRP machine with 16 GB of memory, the default value will be 1 GB.

The larger mbuf pool will allow greater network throughput on large SMP systems.

2.13.1.2 Network Memory Buffer Pool Allocation Algorithm

The algorithm used by the net_malloc kernel service for allocating buffers of various sizes has been changed. The high-water marks for various buffer sizes have been increased, particularly the 2 KB size used by the Ethernet and token ring device drivers.

Requests for memory buffers between 16 KB and 128 KB are now rounded to the nearest power of 2. Allocations in this range were rounded to the nearest page size on prior versions of AIX. This change reduces the number of different sizes of buffers available, which in turn reduces the number of free lists the algorithm is required to manage.

The method used by net_malloc to maintain free list information has been changed. On prior versions of AIX, each CPU maintained a free list for each different size of buffer. There is now one central free list for each buffer size between 16 KB and 128 KB in size. Each CPU still maintains a free list for each of the smaller buffer sizes. This change minimizes the amount of memory used by unallocated large buffers, while at the same time still allowing each CPU to maintain a list of smaller buffers for faster allocation.

2.13.1.3 Network Memory Buffer Pool Statistics

The kernel maintains usage statistics for the buffers allocated from the network memory buffer pool. The information contains details of the number of buffers of each size, and for each size, information on the number of buffers in use and the number of failed requests. This information can be displayed using the `netstat -m` command. In addition to maintaining information indexed by buffer size, the kernel also maintains information indexed by the purpose the buffer is being used for. The type indexed information can be difficult to maintain, so AIX Version 4.3.2 has introduced a method to disable it.

The new extendednetstats network variable, which is altered using the `no` command, determines whether the by-type statistical information should be collected by the kernel. The variable has a default value of 1, which means that the kernel will collect the information. In order to improve networking performance, the default AIX installation disables collection of this information by changing the value of extendednetstats to 0. This is performed in the file /etc/rc.net that is run during system start up. If you want to view the by-type statistics, you should comment out the section of /etc/rc.net that changes the value of extendednetstats.

The following fragment is from the end of the file and appears as:

```
##################################################################
# This disables extended netstat statistics for performance
# reasons.  To have extended netstat statistics enabled on
# future reboots, comment out the following three lines.
##################################################################
if [ -f /usr/sbin/no ] ; then
        /usr/sbin/no -o extendednetstats=0 >>/dev/null 2>&1
fi
```

The information collected when extendednetstats is set to a value of 1 is displayed near the end of the output produced by the `netstat -m` command. An example of the output is shown as followings.

```
# netstat -m
61 mbufs in use:
32 mbuf cluster pages in use
143 Kbytes allocated to mbufs
0 requests for mbufs denied
0 calls to protocol drain routines
0 sockets not created because sockthresh was reached

Kernel malloc statistics:

******* CPU 0 *******
By size       inuse     calls failed    free   hiwat   freed
32              201       749      0      55     640       0
64               98       325      0      30     320       0
128              61       257      0      35     160       0
256             135      7324      0      25     384       0
512             110       937      0       2      40       0
1024             35       276      0       5     100       0
2048              0       482      0       6     100       0
4096             34        68      0       2     120       0
16384             1         1      0      18      24       7
32768             1         1      0       0     511       0

By type       inuse     calls failed  memuse  memmax  mapb
mbuf             61      6952      0   15616   21760     0
mcluster         32      1123      0  131072  141312     0
```

```
socket          212      1034      0   76480   77536     0
pcb              77       457      0   14144   14400     0
routetbl         19        21      0    3008    3648     0
fragtbl           0       132      0       0      32     0
ifaddr            6         6      0    1472    1472     0
mblk             23       196      0    4992    5504     0
mblkdata          2       125      0     512    2816     0
strhead          11        19      0    3232    3680     0
strqueue         18        38      0    9216   10752     0
strmodsw         22        22      0    1408    1408     0
strosr            0        17      0       0     256     0
strsyncq         26        88      0    2752    3200     0
streams         138       153      0   15520   16096     0
file              1         1      0     128     128     0
kernel table     14        14      0   50016   50016     0
locking           3         3      0     384     384     0
temp              9        15      0    5568   10112     0
mcast opts        0         2      0       0     128     0
mcast addrs       2         2      0     128     128     0

Streams mblk statistic failures:
0 high priority mblk failures
0 medium priority mblk failures
0 low priority mblk failures
#
```

When extendednetstats is set to a value of 0, the by-type information is not displayed.

2.13.2 Expanded Kernel Heap (4.3.2)

AIX Version 4.3.2 has added a dedicated memory segment for the kernel heap. The kernel heap is a general memory allocation area and is used to store the dynamic data structures created and used by the kernel, kernel extensions, and device drivers.

This enhancement has increased the maximum size of the heap by an additional 256 MB. The kernel heap was previously located in the upper part of segment 0 that was not used for kernel text pages. The maximum size of the heap depended on the size of the kernel image that was running. The original location is now used as an overflow buffer and is only used if the dedicated 256 MB kernel heap segment becomes exhausted.

2.13.3 Network Water Marks Scaling with thewall (4.3.3)

The network memory allocator (net_malloc) is based on the 4.3 BSD UNIX general purpose memory allocator. It was designed before machines with gigabytes of memory were available and little improvements were introduced since the original design.

In brief, net_malloc is a power of 2 memory allocator and has a set of buckets, each of size 2^n. There is a set of these buckets for each CPU in the system. Each of these buckets maintains of list of free memory pieces of the specific size. This bucket structure has a high water mark: when the length of the bucket free list is greater than the bucket high water mark, net_malloc attempts to return memory from the bucket free list to the system by coalescing and unpinning pages. When the high water mark is set too low for a particular workload, net_malloc may spend much time coalescing and unpinning free memory, only to pin the memory again a short time later.

The network memory allocator also has a low water mark for the MAXALLOCSAVE (16384 bytes) sized buckets. When net_malloc detects shortages in buckets of size less than MAXALLOCSAVE, it attempts to steal memory from buckets whose size is between PAGESIZE (4096 bytes) and MAXALLOCSAVE. By keeping a low water mark for the MAXALLOCSAVE sized buckets, net_malloc ensures there is always memory available to be divided into smaller pieces when necessary. This low water mark is maintained by the netm thread and if we run low on MAXALLOCSAVE sized buffers on some CPU it may take some time before the thread gets around to replenishing the MAXALLOCSAVE bucket on that CPU.

Water marks have been tuned for some of the buckets, but this tuning is static and does not scale with thewall, which is the maximum amount of memory, in kilobytes, that net_malloc is allowed to pin.

The default (and maximum) value of thewall is represented in Table 3:

Table 3. Default and Maximum Values of thewall

Platform	**Maximum (and default) thewall value**
CHRP	$\mathrm{Min}\left(1048576\ ;\ \frac{\text{real memory size}}{2048}\right)$
non CHRP	$\mathrm{Min}\left(262144\ ;\ \frac{\text{real memory size}}{2048}\right)$

Starting from AIX 4.3.3 the two watermarks are set according to the *default* value of thewall parameter, that is the physical amount of memory installed, when the machine boots. In this way, systems with large amounts of memory experience fewer dropped packets and fewer dropped connections, causing the system to handle fewer retransmissions.

This new feature causes the network memory allocator subsystem to have more pinned pages on its free lists. However, this is the trade off of memory against CPU time. By having more memory available on free lists, the are two advantages:

- Less CPU time is spent in the main code paths pinning pages
- Less CPU time is spent processing dropped packets and retransmissions

2.13.4 Larger Pipe Buffer Pool (4.3.2)

The increased size of the kernel heap means that more space can be used for pipe buffers by the kernel, therefore increasing the number of simultaneously open pipes. As with the mbuf pool, the amount of kernel virtual memory reserved for the pipe buffer pool depends on the total amount of physical memory. The system will allocate an amount of virtual memory equivalent to one eighth of the physical memory, or 64 MB, whichever is smaller, with a minimum allocation of 16 MB. Of the memory reserved for use as pipe buffers, 1 MB is pinned in physical memory for faster initial buffer allocation. The size of each individual pipe buffer remains the same as on previous versions of AIX, at 4 KB.

2.13.5 Inter-Process Communication Identifier Enhancement (4.3.2)

The limits of the maximum number of IPC identifiers have been increased, as provided in Table 4.

Table 4. IPC Identifier Limits

Value	Previous Limit	New Limit
Message queue IDs	4096	131072
Semaphore set IDs	4096	131072
Shared memory IDs	4096	131072

In addition to increasing the number of identifiers available, AIX Version 4.3.2 has also implemented a new algorithm to handle the ipcget() routine.

The previous implementation used a sequential search algorithm for traversing the list of IPC identifiers. For a table size of N, the algorithm resulted in N operations for a search miss and N/2 operations for a search hit. Although this is very simple, it does not scale very well. The algorithm has been replaced with a hash table implementation that is better matched to the larger number of IPC identifiers now available.

The IPC support commands, such as `ipcrm` and `ipcs`, have also been changed to take account of the increased limits.

2.13.6 Boot Logical Volume Scaling (4.3.2)

In AIX 4.3.2, the boot logical volume is expanded to enable system configurations with up to 1,000 devices.

As more and more devices are added to a system, the ODM object classes containing device configuration data will grow larger and larger. It is possible that the RAM file system used in the initial stages of booting will not be large enough for the larger ODM files. The existing boot process accounts for this when booting from disk by dynamically expanding the RAM file system based on the amount of system memory. The increased savebase area will not fit on the boot logical volume when adding large amount of devices.

2.13.7 Kernel Services Locks (4.3.3)

Locks are kernel services that are used to serialize and co-ordinate work by multiple threads. For example, if two executing threads both have simultaneous write access to critical data areas, then data corruption and confusion may be the end result. There was usually no need to use locks on a uniprocessor system since interrupt control was sufficient. However, the growing number of CPUs that you can use with a single AIX system image means that more flexible co-ordination mechanisms are required. Not only does your application need to use a locking mechanism to organize data access, but also different parts of the AIX operating system must also use locks.

AIX Version 4.3.3 includes lock enhancements that enable it to support extra CPUs. The AIX documentation includes information about locks, such as the article *Locking Kernel Services* in the publication *AIX Version 4.3 Kernel Extensions and Device Support Programming Concepts*, SC23-4125. This article describes two major types of locks that are used:

- Simple Locks
- Complex Locks

2.13.7.1 Some Specific Lock Enhancements

As well as enhancements affecting how specific locks are used in AIX, the lock code itself has been improved. Specifically, there are front-ends to the complex lock code optimized for the following cases:

- Take read lock with no one holding the lock
- Take write lock with no one holding the lock

xmalloc

The xmalloc kernel service allocates an area of memory out of the heap. In the past, the kernel heap has been serialized with a single lock for all allocations.

This memory allocation process has been redesigned so that it consists of a front end and a back end. The back end code behind xmalloc uses many more locks, so that its much less likely that multiple CPUs will suffer lock contention related delays if they cannot have their xmalloc request satisfied by the front end code.

The fast front end code to xmalloc eliminates contention since it provides each processor with its own set of lists from which it can allocate memory. If the lists do not satisfy the allocation, then the back end is called to get memory from the kernel heap. When the processor is finished with this memory from the kernel heap, the xmfree routine may return that memory to the processor’s list of memory it can allocate, rather than to the kernel heap. Much of kernel memory use from the heap is short term, and many of the lists will remain relatively short.

This combination of the front end and back end components of xmalloc is thus effectively contention free, so that xmalloc can now scale linearly with the addition of any supported number of CPUs. That is, no matter how many CPUs, there is never any measurable contention for any of the kernel heap related locks.

Other Lock Changes

The vfs_list_lock has been redesigned to reduce lock contention. The tod_lock has been converted from a simple_lock to a complex_lock.

2.13.7.2 FIFO Files, or Pipes

A *FIFO* is first in first out special file that is more commonly known as a pipe. In the previous AIX release, FIFO buffers were allocated from a pool of buffers. When a FIFO finished with a buffer it was returned to the pool. Therefore, throughout the life of the FIFO, it had to allocate and free buffers as needed. A global lock had to be taken when allocating and freeing buffs from the pool. When the number of FIFO's reading and writing became significant enough, contention on the global lock (FIFOBUF_LOCK) sometimes rose to unacceptable levels.

Now AIX has eliminated this contention by allocating a slot within a segment for the fifonode buffer. Even after the fifonode has been released and returned to the fifonode pool, the fifonode maintains the same slot.These buffer slots exist within their own segment(s). New segments are created once all slots in the current segment have been allocated. A possible side effect of this change is that it may cause an application foot print to become larger. This is due to the fact that previously a buffer was allocated and released during the read/write operation. The buffer was not held. In this case, the pages are not released after the read or write operation. The buffer is held onto for the life of the fifonode. This would be most obvious with an application that opened a large number of FIFOs, held them open and did occasional read or writes. For additional information, refer to the article *File Creation and Removal* in the publication *General Programming Concepts: Writing and Debugging Programs*, SC23-4128.

2.13.8 Networking Enhancements (4.3.3)

Some of the changes affecting kernel networking include:

- Further reduction of INIFADDR_LOCK contention by the elimination of the need to take this lock for every incoming and outgoing packet.
- NFS performance over UDP has been enhanced by the improvement of the socket connections.
- KRPC_XPRT_LOCK contention for the NFS server has been reduced. The nfsd thread startup mechanism has been redesigned so that the lock does not need to be taken every time a request comes in.

2.13.9 VMM Enhancements (4.3.3)

As the amount of AIX memory increases, systems have increased addressability requirements. The VMM continues to have internal improvements, such as a reduction in lock contention. The following sections 2.13.9.2, “Memory Pools” on page 53 and 2.13.9.3, “Segments” on page 54 are mainly of interest if you work with memory analysis tools.

2.13.9.1 VMM Background Information

Virtual memory is a mechanism where the real memory available for use appears larger than its true size. The virtual memory system is composed of physical disk space where portions of a file that are not currently in use are stored, as well as the system’s real memory. The physical disk part of virtual memory is divided into three types of segments that reflect where the data is being stored:

- *Local persistent segments* from a local file system
- *Working segments* in the paging space
- *Client persistent segments* from CD-ROM or remote file systems

One of the basic building blocks of the AIX memory system is the *segment*, which is a 256 MB (2^{28} bytes) piece of the virtual address space. Each segment is further divided into 4096 byte *page*s of information. Each page sits in a 4 KB partition of the disk known as a slot. Similarly, real memory is divided into 4096 byte *page frame*s. A *frame* (or page frame) usually means a physical memory page as opposed to a virtual page; the context usually of its use usually indicates which one is intended. When a page is needed from its disk location, it is loaded into a frame in real memory.

The Virtual Memory Manager (VMM) coordinates and manages all the activities associated with the virtual memory system. The VMM is responsible for allocating real memory page frames and resolving references to pages that are not currently in real memory.

2.13.9.2 Memory Pools

Previous releases of AIX managed all of a system's real memory as one large resource that was available for the programs executing in one or more CPUs to address and use through the VMM. During this time, there was only one list of free memory frames, and one page replacement daemon that would help ensure that the required pages could be located in system RAM.

Since systems continue to grow (for example, 64 GB is possible on an RS6000 Model S80), AIX 4.3.3 has improved memory management through the use of multiple free frame lists, and multiple page replacement daemons. This increases the VMM concurrency since contention has been reduced in the serialization mechanisms and processes with lower latencies can now service the memory requests.

A *memory pool* is a range of memory on which operates a single memory replacement operation; that is, only one least recently used (LRU) manages this pool of memory frames. A *memory frame* (or page frame) appears in one, and only one memory pool. A *frame set* is a set of memory frames; the frames in a set belong exclusively to that set.

Using this terminology, previous releases of AIX can be considered to consist of one memory pool and one frame set. In AIX 4.3.3, all of the system wide memory frames are managed with multiple memory pools. Each pool has roughly the same number of frames (so that the system is balanced), and the frames in a pool are organized in multiple frame sets. The number of frame sets, and the number of memory pools, depends on the number of CPUs and the amount of real memory in the system. In the AIX Version 4.3.3 implementation, for each memory pool, a LRU daemon called *lrud* (least recently used daemon) is created and started at the end of the VMM initialization. To see the current implementation of the VMM memory pool mechanism has been implemented on an MP system, perform the following:

```
# echo thread|crash|grep lrud
>  14 s     e1d      e1c  unbound    FIFO  10   0              lrud
 17 s    1123      e1c  unbound    FIFO  10   0              lrud
> # lsattr -El sys0|grep realmem
realmem      1048576         Amount of usable physical memory in Kbytes
False
# lsdev -Cc processor|wc -l
      12
```

As previously described, the number of memory pools depends on the number of CPUs and the amount of real memory on your system. For an MP kernel (packaged in the bos.mp fileset), there should be at least one lrud daemon even if it is running on a single CPU system. With the UP kernel (in the bos.up fileset), there is only one memory pool and one frameset. There will never be a lrud when the UP kernel is active.

Now, if a thread needs some page frames, it is no longer constrained by having only one lrud available to check memory in an MP environment. Relevant VMM locks have also been enhanced.

Debugger Enhancements

The KDB kernel debugger has been modified in order to provide visibility of memory pools and frame sets. (Note that `crash` and `lldb` do not have these changes.) The following commands have been modified:

- ppda
- pft 1,2,3,7,8
- rmap
- vmlocks
- pfhdata

The debugger has been enhanced with the following new commands that display all the new data structures:

- vmpool
- frameset [* | frs_id]
- mempool [* | memp_id]

2.13.9.3 Segments

Information about segments is stored in segment control blocks (SCBs). Previous releases of AIX were limited to 2^{20} SCBs. The PowerPC architecture allows for 24 bits to be used for the segment id, but the 24 bits in the SCB were used as follows:

- 1 bit to specify a page is in I/O state (IOSID)
- 3 bits to support big files up to 2 GB (8 segments of 256 MB each, without requiring 8 unique segment IDs)
- 20 bits to specify the segment ID

There is now another mechanism for handling big files up to 64 GB, so all files bigger than the segment size of 256 MB can now be supported with this mechanism. This means that there are now 3 bits that can be used for segment IDs. However, these SCBs themselves are stored in a segment; this is segment 0xB, the VMM data segment. Thus all 3 bits cannot be used for the segment ID to get an eight-fold increase in the total number of segments, since the complete information about all these segments would not fit in the VMM data segment. To solve this problem, extended SCBs, (or mini-SCBs) are now used. Since the 23 index bits allow for up to eight million segments, and two million are used for big SCBs, there are six million left for mini-SCBs.

2.13.10 Threads and Processes (4.3.3)

With up to 24 CPUs available on the Model S80, a single AIX system potentially manages an increased number of processes and threads. Consequently, support of an increased number of threads and processes has been enhanced. Useful background reading includes the article *SMP Scheduling* in the publication *AIX Versions 3.2 and 4 Performance Tuning Guide*, SC23-2365.

2.13.10.1 Threads and Process Limits

With previous releases of AIX, the following limits applied:

- Maximum number of processes was 128K
- Maximum number of threads was 128K

The kernel’s 15th segment, segment 0xE, holds global kernel data, specifically the process and thread tables. The contents of this segment was reorganized so that these limits have been increased, as shown in Figure 3 on page 56.

You can better understand these changes by looking at the system header files, and comparing them to those for the previous 4.3.2 Release of AIX. For example, `/usr/include/sys/thread.h` has increased THREADSHIFT so that the line now reads as:

```
#define THREADSHIFT     19             /* number of bits in thread index */
```

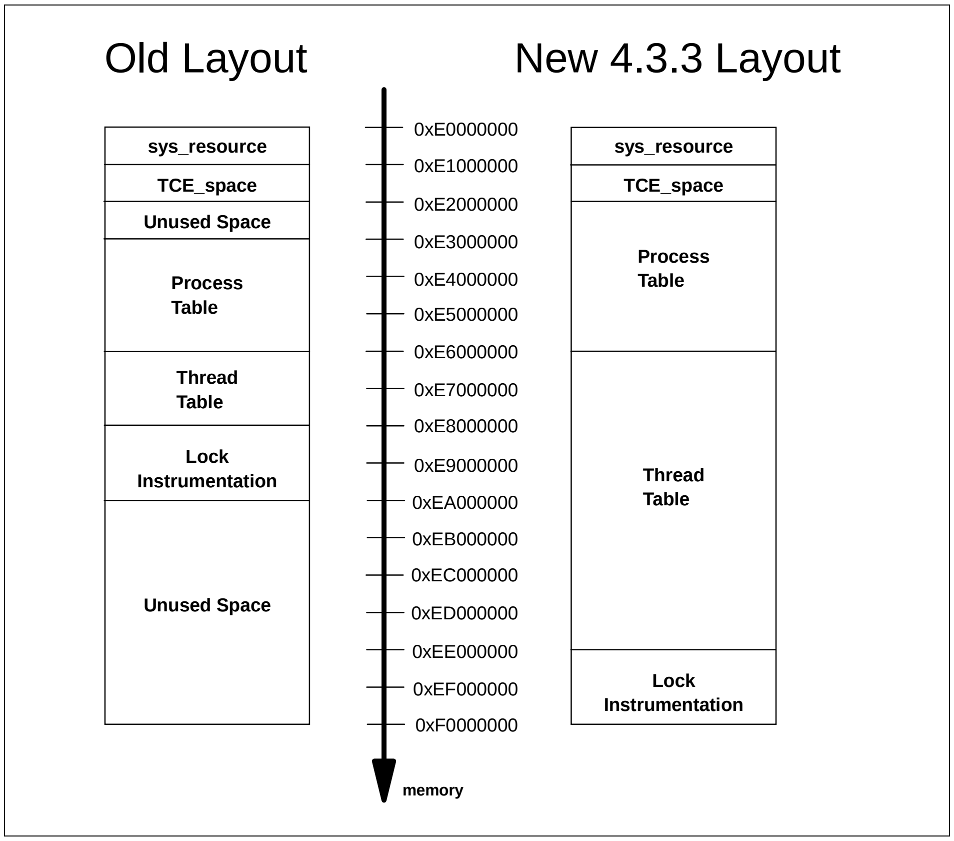


Figure 3. Kernel Segment 0xE

Threads

Before AIX 4.3.3, the thread table was in the address range of 0xE6000000 to 0xE8000000, so there was 0x2000000 bytes of space. Since each thread table entry requires 256 bytes, which is 0x100, then there could only be 0x20000 table entries, which is a maximum of 128K threads.

The new address range allows for 0xEE000000 - 0xE6000000 = 0x8000000 bytes of space. Since the table entries are still 256 bytes each, and there has also been an increase of the THREADSHIFT index variable, then you can have up to 0x80000 thread table entries, which is 512K threads. The change in the thread table address use is shown in Figure 3.

Processes

As was the case with threads, the available space in kernel segment 0xE, and the PROCSHIFT variable, limited you to 128K processes. Each process entry in the proc table requires 384 bytes, which equals 0x180.

Since the new address range provides 0x4000000 bytes of space, then you now can have up to 0x2AAAA proc table entries, or 174,762 processes.

Lock Enhancements

With the increase in the number of processes in the system, the maximum limit of processes per user using the p_uidl pointer in the proc structure was causing contention on proc_tbl_lock. The p_uidl pointer has hence been replaced with pointer to a common user specific information structure (uidinfo), which is globally visible to all processes in the system.

2.13.10.2 Run Queues for CPU Scheduling

AIX Version 4.3.3 offers improved cache affinity through the use of multiple run queues. The new kernel scheduler implements a single global run queue along with a set of local run queues, where each processor has a dedicated local run queue. With multiple run queues, there are multiple locks that eliminates a major source of lock contention.

Once a thread is placed on a local run queue, it generally stays there until an imbalance is detected. Thresholds are used to limit the amount of load balancing that takes place. In general, threads have better processor affinity than in previous versions of AIX, since the set of runable threads that have affinity with a particular processor is strongly held in its local run queue. In the previous version of the scheduler, the scheduler looked at a number of threads on its single run queue and was sometimes able to protect threads' affinities while scheduling.

Load Balancing

Load Balancing refers to a number of distinct algorithms designed to keep the various run queues of a system approximately equally utilized.

The load balancing algorithms include:

- *Initial load balancing* attempts to spread new work evenly across the processors of the system, using the global run queue if there are no idle CPUs. All CPUs can serve the global run queue.
- *Idle load balancing* attempts to find work for processors that have gone idle, but with a bias to maintain a certain degree of processor affinity as well.

- *Busy load balancing* is performed during periods of extreme load when no processors go idle.
- *Starvation load balancing* detects threads that have been waiting in local run queues for a long time, and pushes them onto the global run queue so they may be rescheduled.

The global run queue is used for fixed priority POSIX compliant unbound threads (those with XPG_SUS_ENV=ON exported when they are executed.) This preserves the POSIX requirement for running fixed priority threads in strict priority order.

2.13.11 I/O Wait Time Calculation on SMP Systems (4.3.3)

The calculation of I/O wait time on SMP systems has been modified to provide a more accurate accounting of CPU utilization in tools like `vmstat` and `iostat`.

2.13.11.1 Previous Releases of AIX

The high I/O wait on AIX SMPs was a statistical anomaly of the way AIX counted CPU time. The commands `vmstat` and `iostat` simply reported the CPU break down into the four categories of usr/sys/wio/idle as tabulated within the kernel.

At each clock interrupt on each processor (100 times a second in AIX), a determination is made as to which of the four categories to place the last 10 ms of time. If the CPU is busy in user mode at the time of the clock interrupt, then usr gets the clock tick added into its category. If the CPU was busy in kernel mode at the time of the clock interrupt, then sys category gets the tick. If the CPU was not busy, then a check is made to see if any I/O to disk is in progress. If any disk I/O is in progress, then the wio category is incremented. If no disk I/O was or is in progress and the CPU is not busy, then the idle category gets the tick.

Notice in the prior discussion that it does not matter which processor starts the I/O. This fact leads to higher wio times on SMP systems compared to UP systems in some situations. Using a `cp` copy command example, one processor will do all the work (assume the `cp` process is bound) and so that processor would show some combination of usr, sys and wio percentages, but no idle. If this were a UP, there would be no miscalculation. On an SMP system, however, the other processors (three others in the case of a four-way SMP) would show some combination of wio and idl, but no usr or sys percentage. Table 5 on page 59 shows a hypothetical example of the impact of the `cp` command on a 4 way system, as reported by vmstat or iostat (cp runs on processor 0). Note that this is a hypothetical example for a total of 400 ticks:

Table 5. Per Second Ticks in each Category

Processor	usr	sys	wio	idl
0	30	40	30	0
1	0	0	90	10
2	0	0	90	10
3	0	0	90	10
Average	7.5	10	75	7.5

Any disk I/O in progress would cause all idle processors to show wio instead of idle regardless of which processor initiated the I/O activity. There was no track kept of which one started the I/O. I/O wait goes up compared to UP systems and an eight-way SMP would show even higher I/O wait compared to the four-way in theTable 5 example (82.5%).

2.13.11.2 AIX 4.3.3

The time attributed to I/O wait is no longer inflated; all CPUs are no longer attributed wait time when a disk is busy and the CPU is idle. The decision is based on whether a thread is awaiting an I/O on the CPU being measured.

This method can report much lower wio times when just a few threads are doing I/O and the system is otherwise idle. For example, an RS/6000 with four CPUs and one thread doing I/O will report a maximum of 25% wio time. An RS/6000 with 12 CPUs and one thread doing I/O will report a maximum of 8.3% wio time.

2.14 Scheduler Enhancements (4.3.2)

The scheduler on AIX Version 4.3.2 has been enhanced to increase the impact of using the `nice` command to alter the priority of a thread. The following sections explain how the changes have been implemented and show sample results by comparing the new scheduler with the previous version.

2.14.1 Thread Priority Calculation Changes

All threads on a system have a priority value between 0 and 127, with 60 being the default initial value. As a thread runs and consumes CPU time, the priority value changes numerically as a function of CPU time recently used. A numerically higher number represents a less favored priority. A thread that started with the default priority of 60 may have an instantaneous priority in the range 60 to 126. The value of 127 is reserved for the wait process. The scheduler runs all threads at priority N that are marked as runable before it runs any threads at priority N+1, thus favoring threads using less CPU time.

The `nice` and `renice` commands, and the setpriority system call, can be used to change the initial priority of a thread by a given delta. The delta can be in the range -20 to 19. Thus a thread can have an initial absolute priority in the range 40 to 79. The absolute initial priority, or *nice* value, is included in the calculation of a threads priority. This introduces the idea of relative priority between threads.

In addition to the nice value, the `schedtune` command can be used to fine tune the method used to calculate the new priority. The calculation also has parameters that scale the relative importance of recent CPU utilization (the -r option to schedtune, shown as sched_R) and historical CPU utilization (the -d option to schedtune, shown as sched_D). Both the sched_R and sched_D parameters have a default value of 16.

On versions of AIX prior to 4.3.2, thread priority is calculated using the following algorithm:

- Once per clock tick: cpu = cpu + 1 for the currently running thread, limited to a maximum of 120
- Priority calculation: (cpu * sched_R) / (2 * 16) + nice, limited to a maximum of 126
- Once per second ageing of all threads: cpu = cpu * sched_D / 32

With the default values in place, this equates to:

- Priority calculation: cpu / 2 + 60
- Once per second ageing of all threads: cpu = cpu / 2

The scheduler on AIX 4.3.2 now uses the following algorithm to calculate thread priorities.

- Once per clock tick: cpu = cpu + 1 for the currently running thread, limited to a maximum of 120
- Priority calculation part 1:
  - xnice = (nice > DEFAULT_NICE) ? (2*nice) - 60 : nice
- Priority calculation part 2 (limited to a maximum of 126):
  - p = (cpu * sched_R * (xnice + 4))/(32*(DEFAULT_NICE + 4)) + xnice

- Once per second ageing of all threads: cpu = cpu * sched_D / 32

The `nice` value has a much greater impact on the priority of a thread. It is now included in the calculation as a multiplier of the recent CPU usage in addition to the use as a constant factor.

With the default values of 16 for sched_R and sched_D, and 60 for `nice` and DEFAULT_NICE, the priority calculation equates to:

- Priority calculation: cpu / 2 + 60
- Once per second ageing of all threads: cpu = cpu / 2

Although the algorithm has changed, the default values provide an identical function.

2.14.2 Sample Results of Altering Nice Value

The following tables list the results of changing the nice value of a thread on two identical machines; one running the old algorithm, and the other running the new algorithm. In each case, the tables list the percentage of CPU time delivered to one thread that has been *niced* by the indicated delta, which is in competition with varying numbers of default priority threads. All threads were running the same CPU bound application.

From comparison of the values in Table 6 and Table 7, it can be seen that the effect of a positive nice delta on a thread has been enhanced. Take, for example, a thread running with a nice delta of 19 in competition with one default thread. Previously, the niced thread would receive 41 percent of the CPU, with the default thread receiving the remaining 59 percent. With the new algorithm, the niced thread has been reduced to 15 percent, with the default thread increasing to 85 percent.

In addition, the effect of a negative nice delta has been increased. A thread running with a nice delta of -20 competing against 31 default threads now receives 32 percent of the CPU, compared with 23 percent under the previous algorithm. Correspondingly, the CPU delivered to each of the remaining 31 default threads has decreased from 2.5 percent to 2.2 percent.

Table 6. Old Priority Algorithm, sched_R and sched_D Defaulted to 16

nice delta	**Number of threads running (1 niced, others default)**							
	1	2	3	4	5	8	16	32
-20	100	60	47	40	36	30	25	23
-15	100	58	43	36	32	26	21	17
-10	100	55	40	32	28	21	15	13
-5	100	53	37	29	24	17	11	8
0	100	50	33	25	20	13	6	3
5	100	48	30	21	16	7	2	0
10	100	45	27	17	12	4	0	0
15	100	43	23	14	8	0	0	0
19	100	41	21	11	4	0	0	0

Table 7. New Priority Algorithm, sched_R and sched_D Defaulted to 16

nice delta	**Number of threads running (1 niced, others default)**							
	1	2	3	4	5	8	16	32
-20	100	78	60	52	48	42	36	32
-15	100	69	51	43	39	33	26	24
-10	100	60	45	37	32	25	19	16
-5	100	55	39	31	25	19	12	9
0	100	50	33	25	20	13	6	3
5	100	42	24	16	11	4	0	0
10	100	32	17	8	4	0	0	0
15	100	22	11	3	0	0	0	0
19	100	15	6	0	0	0	0	0

It can be seen from the values in Table 8 and Table 9 that decreasing the value of the sched_R parameter makes the effect of the nice delta even greater. By reducing the value of sched_R, the priority algorithm places less emphasis on recently used CPU time. Consequently, the thread with a negative nice delta receives even more CPU time.

Table 8. Old Priority Algorithm, sched_R=8

nice delta	Number of threads running (1 niced, others default)							
	1	2	3	4	5	8	16	32
-20	100	78	60	55	52	47	43	41
-15	100	68	54	48	44	39	34	32
-10	100	60	47	40	36	30	25	23
-5	100	55	40	32	28	21	15	13
0	100	50	33	25	20	13	6	3
5	100	45	27	18	12	3	0	0
10	100	40	20	10	4	0	0	0
15	32	14	2	0	0	0	0	0
19	100	24	8	0	0	0	0	0

Table 9. New Priority Algorithm, sched_R=8

nice delta	Number of threads running (1 niced, others default)							
	1	2	3	4	5	8	16	32
-20	100	98	96	94	92	86	70	59
-15	100	84	68	58	53	50	43	41
-10	100	68	52	46	41	36	28	27
-5	100	57	42	35	31	23	18	13
0	100	50	33	25	20	13	6	3
5	100	37	18	9	4	0	0	0
10	100	17	6	0	0	0	0	0
15	100	2	0	0	0	0	0	0
19	100	0	0	0	0	0	0	0

Note

Changing system scheduling parameters using the `schedtune` command can cause unexpected results, particularly if more than one system parameter is changed. See *RS/6000 Performance Tools In Focus*, SG24-4989 for more information on the `schedtune` command, its parameters, and other performance monitoring and tuning tools.

2.15 Fast Device Configuration (4.3.3)

A new device configuration methodology has been introduced in AIX 4.3.3 in order to reduce the time needed to detect and configure all the devices attached to the system. In particular, the `cfgmgr` command has been changed so that it can run device configuration methods in parallel rather than sequentially, one at a time. A new locking mechanism has also been introduced to serialize critical sections of the configuration code.

In general, the more devices that are attached to a system, the longer it takes to configure them. With systems having thousands of disk drives or thousands of TTY devices, it can take AIX an hour or more just configuring them into the kernel at boot time.

Many of the asynchronous I/O adapters take two to five minutes to configure. In a system with 16 asynchronous I/O adapters this means that from 32 to 80 minutes of boot time is required just to configure these adapters. Most of this time is spent waiting on I/O which could be done in parallel if the configure methods were executed in parallel instead of one at a time. Thus, in theory, 16 asynchronous I/O adapters could be configured in the same time it takes to configure one.

2.15.1 The cfgmgr Command

The `cfgmgr` command executes configuration rules from the Config_Rules Object class. They are executed one at a time in the order specified by sequence numbers in the Config_Rules objects. If two or more configuration rules have the same sequence number, they are executed in the order they are returned by the Object Data Manager (ODM).

If a configuration rule is associated with the root node of a device tree in the ODM, it may write device names and device package names to stdout. The `cfgmgr` command parses the stdout for the device names and adds them to a list it maintains for keeping track of the devices for which it must execute device configure methods. The names are added to the list in the same order they were returned in stdout. The `cfgmgr` command then processes this list by looking up each device’s configure method in the ODM and executing them one at a time. The methods may return additional device names and package names to the `cfgmgr` command using stdout, and those are added to the end of the list devices to be configured.

When new devices are discovered, named, and added to the ODM, the `cfgmgr` command ensures that this is done in a repeatable fashion: two identical systems with identical devices all cabled and connected identically always have their devices named exactly in the same way.

The new version of the `cfgmgr` command is able to have the same behavior of previous releases, though starting some configuration processes in parallel. A full parallelization is not possible since some methods must be serialized and devices must be always named in the same order.

At boot time, the configuration process uses the RS/6000’s LEDs to display an indication of the type of device being configured. The device configuration methods were responsible for displaying the proper LED value. Now, with the `cfgmgr` command invoking several configure methods in parallel, this scheme would leave the LEDs displaying the value for the last device whose configure method writes the LEDs.

To avoid any confusion, the `cfgmgr` command now is responsible for setting the LEDs for a device. It looks up the proper LED value from the ODM (PdDv and CuAt objects) for the device, and displays this value just prior to running the configure method for the device. It displays the LED value for the most recently invoked configure method. Each time the configure method whose LED value was last displayed completes, the `cfgmgr` command changes the LED value to that for the next most recently invoked configure method that has not yet completed. If there is a configure method that is taking an exceptionally long time to complete or is hung, it will eventually be the only configure method that has not yet completed and hence its LED value will be displayed.

If the `cfgmgr` command is unable to determine the LED value to be displayed for a device, it displays a value of 538. If there is not another configure method running when one completes, the value 539 is shown.

Some systems are equipped with a two line LCD display. On those machines the `cfgmgr` displays the device's physical location codes on the second line. Physical location codes are not the location codes from the CuDv objects in the ODM. They can be obtained from the Open Firmware device tree for many devices, from the CuVPD object class for PCI and built-in adapters. If the device does not have a physical location code in the CuVPD object class but it has an AIX location code in the CuDv object class, then the AIX location code is displayed. If the device has neither a physical location code nor an AIX location code in the ODM, then the device name is displayed.

If it is a configuration rule that is being executed, then the `cfgmgr` command displays the sequence number of the rule on the second line. This can be used to determine which configuration rule is being run.

The `cfgmgr` command's verbose output, produced using the -v option, indicates the name of a configuration method that is invoked and the name of the device it is configuring. This is followed by a display of the stdout and stderr produced by the configure method. When running configure methods in parallel, the invoking of a configure method and its completion are disjoint events. Therefore, the verbose output has been changed to explicitly identify when a configure method is being started and the device being configured. Later, when the configure method completes, the verbose output explicitly indicates that the configure method has completed for the device and then shows the stdout and stderr. Information indicating the number of configure methods currently running, time stamps showing when methods are invoked and when they complete, and the elapsed time are also given.

The following example shows the `cfgmgr -v` command output:

```
cfgmgr is running in phase 2
----------------
Time: 0LEDS: 0x538
invoking top level program -- "/etc/methods/cfgprobe -c /etc/drivers/coreprobe.ext"
Time: 0LEDS: 0x539
return code = 0
****************** no stdout ***********
****************** no stderr ***********
----------------
Time: 0LEDS: 0x538
invoking top level program -- "/etc/methods/defsys"
Time: 0LEDS: 0x539
return code = 0
****************** stdout ***********
sys0

****************** no stderr ***********
----------------
attempting to configure device 'sys0'
Time: 0LEDS: 0x811
invoking /usr/lib/methods/cfgsys_chrp -l sys0
Number of running methods: 1
----------------
Completed method for: sys0, Elapsed time = 0
return code = 0
****************** stdout ***********
pmc0
pci0
pci1
pci2
```

```
****************** no stderr ***********

                   ....

                   lines skipped

                   ....

----------------
Time: 1LEDS: 0x746 for scsi1
Number of running methods: 1
----------------
attempting to configure device ’sa0’
Time: 1LEDS: 0x826
invoking /usr/lib/methods/cfgasync_rspc -l sa0
Number of running methods: 2
----------------
attempting to configure device ’sa1’
Time: 1LEDS: 0x826
invoking /usr/lib/methods/cfgasync_rspc -l sa1
Number of running methods: 3
----------------
attempting to configure device ’sa2’
Time: 1LEDS: 0x826
invoking /usr/lib/methods/cfgasync_rspc -l sa2
Number of running methods: 4
----------------
attempting to configure device ’scsi0’
Time: 1LEDS: 0x746
invoking /usr/lib/methods/cfgncr_scsi -l scsi0
Number of running methods: 5
----------------
Completed method for: sa0, Elapsed time = 0
return code = 0
****************** no stdout ***********
****************** no stderr ***********

                   ....

                   lines skipped

                   ....

----------------
calling savebase
return code = 0
****************** no stdout ***********
****************** no stderr ***********
Configuration time: 15 seconds
```

Chapter 3. 64-Bit Enablement

This chapter covers the introduction of 64-bit systems and the support provided in AIX Version 4.3 for these systems. In the first section, an introduction to 64-bit architectures and its benefits is provided, including the hardware and software aspects of 64-bit implementations. The design chosen for RS/6000 systems and the AIX operating system is also explained.

The second section describes the changes made to the core operating system that are necessary to run AIX on 64-bit hardware and enable 64-bit applications. These changes include modifications to the basic application development tools like compiler, linker, and debugger, and other tools that operate on object files.

3.1 Introduction to 64-Bit Computing

The following sections describe some of the features of the new 64-bit environment.

3.1.1 64-Bit Architecture and Benefits

From an operational point of view, an architecture is said to be 64-bit when:

- It can handle 64-bit-long data; in other words, a contiguous block of 64 bits (8 bytes) in memory is defined as one of the elementary units that the CPU can handle. This means that the instruction set includes instructions for moving 64-bit-long data and instructions for performing arithmetic operations on 64-bit-long integers.
- It generates 64-bit-long addresses, both as effective addresses (the addresses generated and used by machine instructions) and as physical addresses (those that address the memory cards plugged into the machine memory slots). Individual processor implementations may generate shorter physical addresses, but the architecture must support 64-bit addresses.

The benefits of 64-bit architectures can be summarized as follows:

- Extended-precision arithmetic. The ability to use very long integers in computations is a feature that can be very useful in specialized applications.
- Access to large data sets. The ability to create and maintain very large file systems is increasingly important for many users. In particular, data

warehousing applications, scientific, and multimedia applications frequently require these features.

- Large address spaces. A 64-bit architecture has the capability of addressing huge address spaces. You should realize that the step to 64-bits is much more than just a doubling of 32-bits. In terms of addressability, it represents a four billion-fold increase. With clever exploitation, these large address spaces can result in spectacular performance improvements or gains in productivity through simplified programming of very large technical problems.

The ability to handle large address spaces is considered the greatest potential benefit for users since, in the future, complex applications such as large databases, large numeric applications, and multimedia environments will need to manage and operate on larger data sets. Since internal memory is much faster than most storage devices, the ability to fetch and keep more data in memory, where it can be directly manipulated, should provide dramatic performance improvements. Table 10 provides the size of the address spaces that can be managed as a function of the length of the address that the CPU generates.

Table 10. Size of Address Space as a Function of Address Length

Address Length	**Flat Address Space**
8-bit	256 Bytes
16-bit	64 Kilobytes
32-bit	4 Gigabytes
52-bit	4000 Terabytes
64-bit	16 Million Terabytes

3.1.2 64-Bit Challenges

As previously mentioned, 64-bit architectures can prove advantageous in many areas of application. It should be noted, however, that these advantages can come at a cost. Extra addressability must be accompanied by very large amounts of system memory to work effectively. Applications compiled in 64-bit mode also consume more disk space than their 32-bit equivalents, adding to the cost of system storage. It is also important to remember that 32-bit applications that cannot or do not take advantage of the features previously mentioned should remain as 32-bit binaries. If compiled in 64-bit mode without change, they will probably not see any performance improvement. It is possible that the application will run slightly slower due to cache effects and longer code-path length.

Although the vast majority of current applications will not fully utilize the functions and capabilities of a 64-bit architecture, the applications of the near future will increasingly view 32-bit technology as a limiting factor.

3.1.3 64-Bit PowerPC Design

The PowerPC architecture is, by its nature, an open, extendable design. There is nothing in the chip architecture itself that would affect binary compatibility as you migrate across different PowerPC implementations. The PowerPC processor architecture was defined from the start as a 64-bit architecture that is a superset of the 32-bit architecture implemented in the 601, 603, and 604 processors.

An important aspect of the 64-bit version of PowerPC is its binary compatibility with the previous PowerPC processors. From the standpoint of the 32-bit and 64-bit specifications, there are a few differences, as shown in Figure 4. The number of CPU registers (the basic storage cell where the CPU stores the data on which it performs its computations) remains the same, but these registers are now 64 bits long instead of 32 bits. A few other control registers also move from 32 to 64 bits in length. Note that the floating point registers do not change in size, since they already conform to industry standards for floating-point that require 32- or 64-bit-long data.

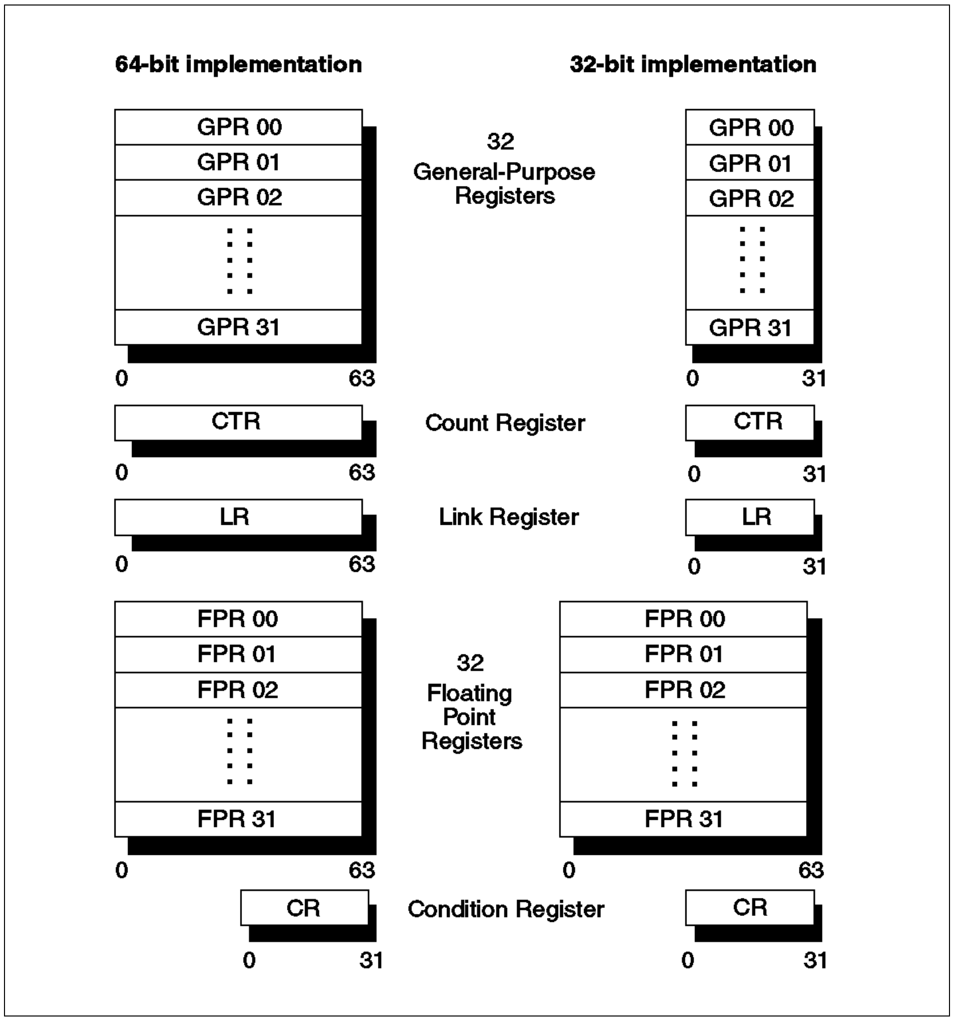


Figure 4. Register Implementation of 32-Bit and 64-Bit PowerPC Processors

In the 64-bit implementation of PowerPC, existing machine instructions do not change drastically. Many instructions simply work the same in 64-bit mode. That is, they can manage 64-bit long data and use/generate 64-bit-long addresses. New instructions, that were not implemented in the previous PowerPC chips, are included for the handling of 64-bit data.

A 64-bit PowerPC can also work in 32-bit mode. In this way, any application that currently runs on the 32-bit PowerPCs can run unchanged. For example, arithmetic instructions running in 32-bit mode operate only on the lower-half of the CPU register involved and consider only that half of the register in the result. 32-bit addresses are handled in the same way.

The virtual address space is the amount of virtual memory that an application can address independent of the size of the physical memory actually installed in the machine on which it is running. Figure 5 shows a simplified representation of the virtual address space that the PowerPC architecture can manage in 32-bit and in 64-bit mode. As shown, the 32-bit implementation is already capable of addressing a very large (2^{52} bytes, refer also to Table 10) address space. The 64-bit implementation goes up to 2^{80} bytes (a huge number that signifies nearly one trillion terabytes). Other 64-bit architectures currently on the market mainly address a 2^{64} bytes-wide virtual address space.

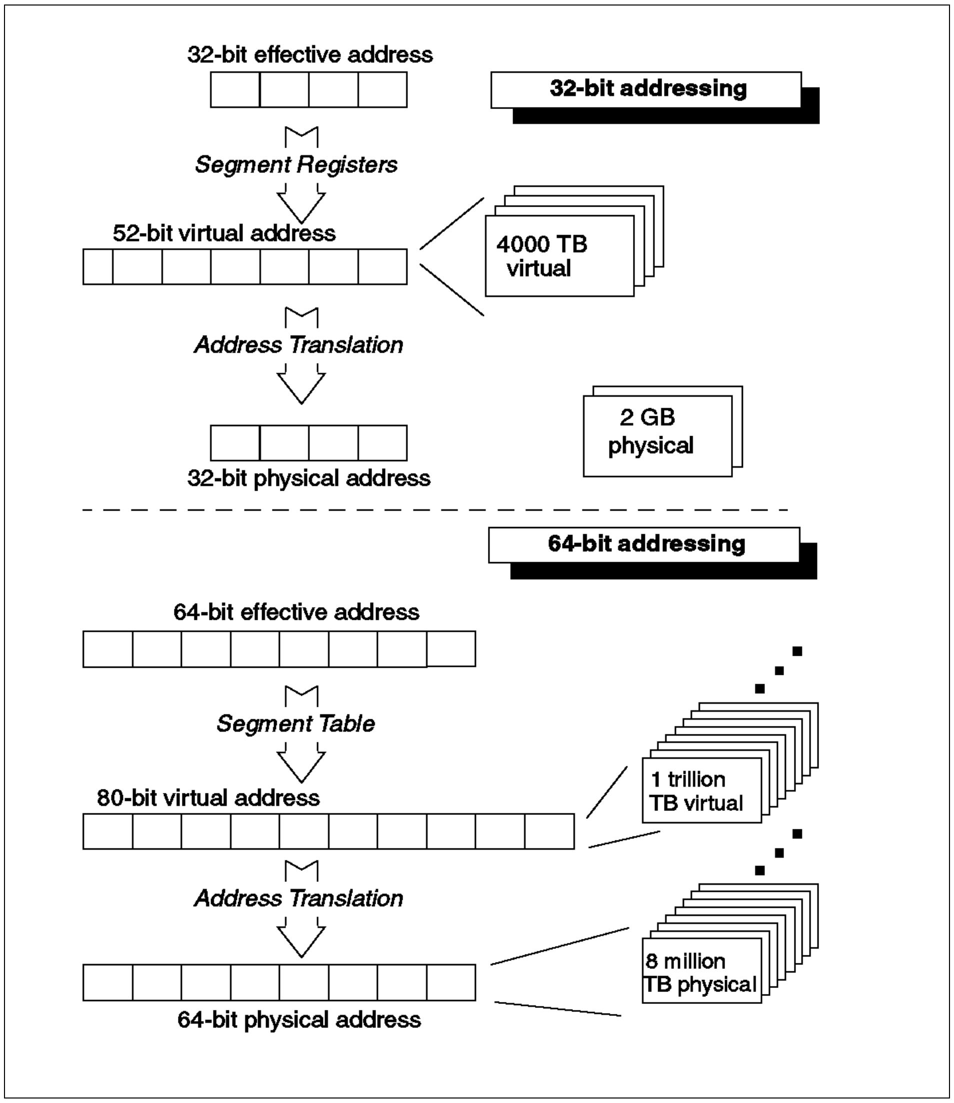


Figure 5. Comparison of Address Translation in 32-Bit and 64-Bit Mode

3.1.4 64-Bit AIX Design Criteria

It is important to note that the 64-bit execution environment for application processes is an upward-compatible addition to AIX capability, not a replacement for the existing 32-bit function. The design IBM chose for 64-bit AIX allows existing 32-bit applications to run with 64-bit applications with no changes, thus protecting the investment users have made in their current applications. Users can take advantage of the features of 64-bit AIX when business needs dictate.

AIX 64-bit support is intended to be run on 64-bit hardware systems, not simply any system containing a 64-bit PowerPC processor. A 64-bit hardware system is one that is based on the RS/6000 architecture and identifies 64-bit properties for the processor(s) and processor host bridges (PHBs) in configurations with memory addressing greater than 32 bits.

For AIX, and the applications that run on AIX, the 64-bit PowerPCs have two important attributes. They are very fast when running as 32-bit processors, and they offer the opportunity of running a 64-bit environment. AIX 4.3 exploits these attributes separately. There are two different execution environments in AIX 4.3, the 32-bit execution environment and the 64-bit execution environment. These two environments are only available for 64-bit hardware. There is no 64-bit execution environment on 32-bit hardware.

Generally, the AIX 4.3 kernel remains 32-bit, and only selected parts, such as the Virtual Memory Manager, are upgraded to be aware of the 64-bit address space. This means that the number of AIX kernels remains two (uniprocessor and multiprocessor).

Although there were a number of choices regarding the basic data types of 64-bit processes, the choice that was made by the Aspen working group, formed by XOPEN, and a consortium of hardware vendors is called LP64, short for Long-Pointer 64. This is commonly also called the 4/8/8 model, which stands for the Integer/Long/Pointer type sizes. The benefit of this configuration is that it provides 64-bit addressing with 8-byte pointers, large object support (8-byte longs), and backward compatibility (4-byte integers). Other alternatives included an 8/8/8 solution, called ILP64, and the LLP64 that were not adapted. Obviously, the choice for AIX was LP64.

3.2 64-Bit Core Design

As stated, the kernel of the AIX operating systems remains 32-bit. To allow 64-bit applications to run and use memory in the 64-bit address space, extensions to the kernel were introduced. This is discussed in Section 3.2.2, “System Calls” on page 80. A fundamental change was also introduced in the object module format that basically enables executables to overcome the size limit of 2^{32} bytes. The subsection on XCOFF addresses these changes.

The implications for device drivers of 64-bit and 32-bit devices in a 64-bit execution environment are explained in Section 3.2.4, “Device Drivers” on page 96.

The loading of 64-bit modules is shared between kernel-space code and user-space code. The changes to the loader are discussed in 3.2.5, “Loader” on page 99.

The Virtual Memory Manager is one of the key parts of the operating system when it comes to large address space and mapping of virtual to real memory. The consequences are summarized in the last part of this section.

3.2.1 Segment Register Mapping

Since the 64-bit hardware uses 64-bit effective addresses while operating in 64-bit execution mode, keeping sixteen segment registers would have required increasing the size of segments proportionally to allow for the increased address space. This is less useful to applications than increasing the number of segments in a process' address space. Therefore, segmentation on the 64-bit hardware remains in units of 256 megabytes. In 64-bit execution mode, the sixteen segment registers used for 32-bit addressing are replaced with a segment table (analogous to the hardware page table) that contains up to 256 of the most recently used mappings of segment numbers to segment ID for the process' user address space.

The architecture also allows a Segment Lookaside Buffer (SLB) to hold a smaller number of recently used segment number to segment ID mappings for the process' user address space. For 32-bit mode, part of the SLB can be used by the hardware to represent the sixteen segment registers.

64-bit processes are limited to 60-bit effective addresses. This is a convenient number for the VMM since 2^{60} represents effective segment IDs up to 32-bits, which fits into one register in the kernel. The 60-bit effective address space will be sparsely instantiated up to the rlimit values for the process or up to some limitation imposed by overall system resources. The choice of address space layout was made to reduce the number of constraints on the size of each area and to allow for future expansion and new uses for address spaces. At some future date, the address space may be expanded to more than 60 bits.

The first sixteen segment numbers are freed as much as possible from default system use and left for application use. This has many advantages. It makes finding incorrect use of 32-bit addresses in 64-bit processes easier; it allows 32-bit and 64-bit applications to share memory at the same addresses in each process, and it allows 64-bit applications to be specially designed to work with interfaces that do not understand 64-bit addresses. For example ioctl().

Segment number 0 is used for the first kernel segment in user mode. It is read-protected in user mode, except for the system call (SVC) tables, svc_instruction code, and system configuration structure.

Segment number 1 is used for the second kernel segment in user mode. It contains the 64-bit system call (SVC) tables.

Segment number 2 is still used for the process private segment. No program-allocatable data is contained in this segment. The address of the user block is at the same location in 32-bit and 64-bit processes.

The segment numbers from 3 to 12 and segment number 14 are reserved for application use by the shmat() and mmap() interfaces.

Segment numbers 13 and 15 are reserved for the new user space loader (see 3.2.5, “Loader” on page 99) used at exec() and load() time. These segments are global, read-only, and are always mapped in.

The address space from segment numbers 0xA0000000 to 0xEFFFFFFF are reserved for future system use. These segments numbers are not given out in response to user requests.

Table 11. Address Space Layout in User Mode

Segment Number	Use in 64-Bit Mode	Use in 32-Bit Mode
0	Kernel	Kernel
1	Kernel	User
2	Process Private	Process Private
3	shmat/mmap	Available for User
4-0xC	shmat/mmap	shmat/mmap
0xD	Loader use	Shared libraries
0xE	shmat/mmap	shmat/mmap
0xF	Loader use	Shared lib data
0x10-0x6FFFFFFF	Application text, data, Bss, heap	N/A
0x70000000-0x7FFFFFFF	Default shmat/mmap	N/A
0x80000000-0x8FFFFFFF	Private load	N/A
0x90000000-0x9FFFFFFF	Shared library text and data	N/A

Segment Number	Use in 64-Bit Mode	Use in 32-Bit Mode
0xA0000000-0xEFFFFFFF	Reserved for system use	N/A
0xF0000000-0xFFFFFFFF	Application stack	N/A

The following list provides more detail on the use of the various segments in 32-bit and 64-bit mode:

- Process private data

  Segment number 2 continues to contain the process private segment. This segment is substantially different for 32-bit and 64-bit processes. The process private segment continues to contain errno, errnop, environ, the top_of_stack structure, and the exec() arguments. It also contains the user structure, primary user thread structure, and the primary kernel thread stack.

  It does not contain the user thread stack or any user data. The user thread structures (other than the primary) is moved to the kernel thread stack segments for both 32-bit and 64-bit processes. The errno, errnop, and environ locations are different in 32-bit and 64-bit mode. The top_of_stack structure is reformatted for the 64-bit values for 64-bit processes.

  The errno, errnop, environ, the top_of_stack structure, and the exec arguments (all the user accessible data) are kept in the lowest one megabyte of this segment and are user-modifiable. All data above this in the segment is read-protected from user access. The segment table, adspace, and segstate structures for the process are allocated from a region above the first megabyte in the segment. The segment table is pinned. The region above these adspace structures and below the primary kernel thread stack is used for the overflow heap for the per-process loader heap.

- Executable text area

  The text area starts in segment number 16. All segments from the start of the executable, through, and including, the loader section of the executable, are mapped in. No segments beyond the segment containing the loader section of the executable are mapped in the address space. The text segments are mapped read-only if the underlying file system supports mapping (JFS, NFS, CD-ROM file system, AFS, and DFS support mapping). Otherwise, the text section and the loader section are copied to working storage segments.

- Executable data and BSS

  Following the text segments are working segments containing the executable's initialized data and BSS (uninitialized data) areas. The data area is vm_map()ed from the executable and relocated.

- Heap

  The break value for the process is initialized just above the BSS area. As the heap is grown (with brk() and sbrk() calls), new segments are allocated above the current segment containing the current break value up to the segment containing the new break value. The heap is not allowed to grow into segment number 0x70000000 or beyond any shmat or mmap segment.

- shmat and mmap segments

  Starting at segment number 0x70000000, shmat and mmap segments are allocated if no address is specified. Segments are allocated in increasing order in the first available segment at, or after, segment number 0x70000000. The shmat and mmap segments are placed where requested if the segment number is available and less than segment number 0x80000000.

- Explicitly-loaded modules

  Explicitly-loaded modules (using the load() system call) were previously loaded into the heap of the process. This creates complexity in dealing with heap expansion/contraction and explicit loads.

  Explicitly-loaded objects are now loaded into separate segments starting at segment number 0x80000000. Segment numbers are allocated in increasing order in the first available segment number at, or after, segment number 0x80000000.

  Explicitly-loaded objects are limited to segment numbers 0x80000000 to 0x8FFFFFFF. To reduce segment table faults, multiple loaded modules are vm_map()ed into these working storage segments. The data for the loaded modules is loaded into these segments (and relocated) also.

- Shared library text and data segments

  Starting at segment number 0x90000000, shared library text and data segments are allocated. These segment numbers are allocated globally (at the same address in all 64-bit address spaces) to allow sharing of the segments (in the case of text segments) and vm_map()ing of the segments (in the case of data segments). Global shared library text and data segments are maintained by the loader using the current method. Shared library text and data segments are limited to segment numbers 0x90000000 to 0x9FFFFFFF.

- User stack

  The initial user stack is allocated starting at the top of segment number 0xFFFFFFFF and will consume additional segment numbers below this as the stack grows. New segment numbers will be allocated at stack growth time. Only the segment number below the segment number containing the current top of stack is allocated. References to more than one segment number away from the current top of the stack are treated as wild references.

  Note: This restricts local variable allocation to less than 256 megabytes of the total allocation in each function. The stack growth is limited to segment numbers ranging from 0xF0000000 to 0xFFFFFFFF as segment numbers in the range 0xA0000000 to 0xEFFFFFFF are reserved for future system use.

- Big data programs

  The maxdata field is used for 32-bit programs to indicate that the heap should be placed in a different place where a larger heap can be accommodated. In 64-bit processes, the address space will always be able to accommodate a large heap, so no indication is necessary. The maxdata and maxstack fields, if set, are used to set the soft rlimit value for 32-bit and 64-bit applications. If the limit specified is greater than the hard rlimit values, the exec() will fail.

3.2.2 System Calls

Because the AIX kernel remains 32-bit, the interfaces to the various system calls must be through the types and structures of 32-bit mode C. 64-bit mode applications, however, are compiled with 64-bit mode types and structures. This section explains how the different types are communicated to the kernel.

On a 64-bit PowerPC, AIX will run both 32-bit-mode processes and 64-bit-mode processes. The problem is that 64-bit applications compiled with the same AIX header files that 32-bit processes use (but compiled for 64-bit execution mode) build data structures, arrays, and scalars using the rules of 64-bit C, though the kernel expects data structures, arrays, and scalars that match the ones built by 32-bit applications using 32-bit C. If a 64-bit application builds, for example, an iove C structure containing an address and a length, this structure cannot be passed directly to the kernel's kreadv() system call routine because that routine cannot interpret the 64-bit address and 64-bit length.

Clearly, some code must be placed between the 64-bit application's system calls and the 32-bit kernel (see Figure 6). AIX takes advantage of the following two features in implementing this interface code:

- The 32-bit and 64-bit name spaces are completely separate, and calls from 64-bit applications are never resolved to 32-bit entry points. Specifically, the calls from 64-bit applications to traditional system call entry points should not be resolved to the 32-bit entry points of the same names exported by the kernel.
- Since UNIX no longer makes a distinction between system calls and subroutines, it is no longer necessary to strictly follow the old UNIX semantics. If a caller passes a bad address to any system-supplied subroutine (whether system call or not), it is permissible to end the calling process, as will happen if a library routine dereferences an incorrect pointer. This means that routines, traditionally considered to be system calls, can reside in subroutine libraries, and almost all of the system call interface code needed for 64-bit processes can be placed in a user-mode library such as libc.a.

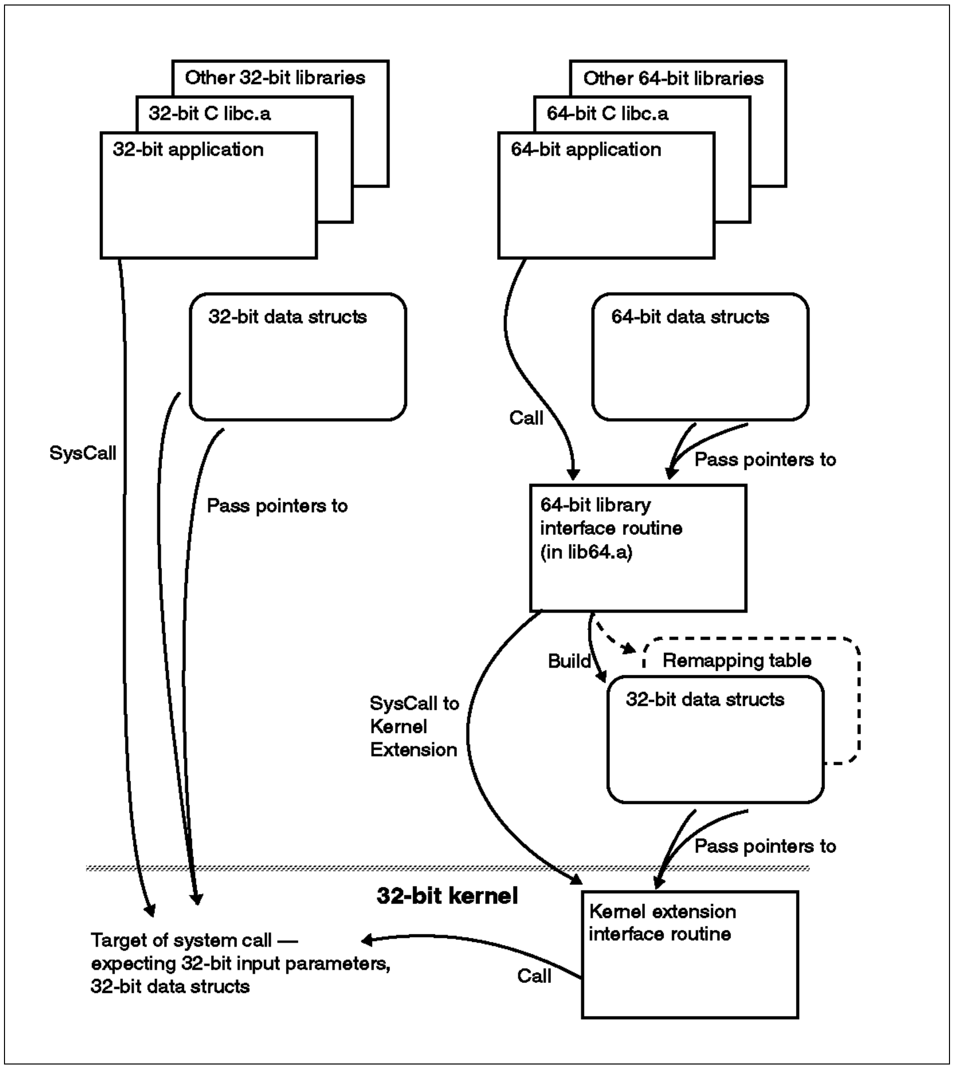


Figure 6. Interfacing 64-Bit Processes to a 32-Bit Kernel

Each system call that is exported by the 32-bit kernel is represented in 64-bit mode by a 64-bit-mode library routine in libc64.a that is the call target of what would have been (in 32-bit mode) a system call. These library routines handle any necessary reformatting of data to communicate with the 32-bit system call routines in the kernel. In many cases, they build a remapping table that tells how the required portions of the 64-bit address space should be reflected in a 32-bit map for the kernel.

On the kernel side, in 32-bit mode, a kernel extension routine is added for each of the system calls supported in 64-bit mode. These routines are invoked from the 64-bit library routines through AIX's syscall interface. They accept the reformatted data from the library routines and perform any necessary remapping of addresses using data supplied by the library routines. The kernel can then properly *see* data structures, buffers, and so on in the user spaces that are referred to in the call.

A typical system call involves several pieces of code located in various places in the system. The names of these various pieces are all derived from the original name of the 32-bit system call. If a particular 32-bit system call is named sample(), then:

- The name of the C language routine in libc64.a that intercepts 64-bit calls to sample() is precisely that: sample. Note that the kernel does not export a symbol named sample to 64-bit-mode processes, so all 64-bit calls to that label reach the library routine.
- The C language routine library, at the point where it needs to invoke the kernel extension routine for this system call, calls __sample() (two leading underscores). This is an assembler language *glue* routine reached through the branch-and-link instruction. The name of its actual entry point will be .__sample (leading period).
- The glue routine, still in the 64-bit library, loads an entry from the TOC and issues a system call instruction. The TOC entry is assembled as a pointer to sample64 (no prefix, suffix 64).
- The kernel extension routine is named sample64(), and this is the name that the kernel extension exports as a syscall. The kernel extension routine will itself call sample() the existing 32-bit kernel service.

The reason for the two underscores is for conformance with ANSI C, which reserves names not starting with two underscores (or one underscore and a capital letter) for the user, except for existing non-conforming UNIX names that are *grandfathered*. All external symbols in libc64.a must begin with two underscores.

3.2.2.1 64-Bit to 32-Bit Data Reformatting

The goal of the 64-bit interface is to make it appear to the kernel that the system service request came from a 32-bit program. To this end, the width of any data passed across the interface in either direction must be adjusted to match the expected size.

Data reformatting is done by the 64-bit library routine that is the call target of 64-bit system calls. It receives 64-bit data from the caller, does any necessary reformatting to 32-bit data, and calls its corresponding kernel extension routine that passes the (now 32-bit) data on to the kernel's system call service routine. On completion of the system call, a return code is (generally) passed back. In addition, the kernel may have passed data back in user space. When the library routine regains control, it expands the return code from 32 bits to 64 and expands any returned data in user space.

For scalar parameters, the library does the following before calling the kernel extension routine:

- char, unsigned char - Passed without change.
- short, unsigned short - Passed without change.
- int, unsigned int, long, unsigned long - Tested to make sure that the value being passed will fit in the 32-bit version of int or unsigned int. A value that is too large generally results in setting errno to EINVAL and returning a return code of -1. Values that are not too large are passed as 32-bit integers.

  Note: Integers that are larger than 2^{32} are valid in some cases. For example, the lseek() routine takes an off_t (that is a long) as the seek position, a value that can be larger than 2^{32} for large files. In this case, the system call is directed to the 32-bit llseek() interface that is prepared to handle long integers.
- float, double - Passed without change.
- pointer - Converted from 64-bit effective address to 32-bit effective address as described in Section 3.2.2.2, “64-Bit to 32-Bit Address Remapping” on page 85.

Many system calls involve passing the address of one or more structures in storage. If the structures involve any data types whose sizes differ between 32-bit and 64-bit mode (int, long, pointer), the library routine must pass a (32-bit) pointer to a local 32-bit copy of the data constructed on its own stack.

Some system calls are 64-bit-enabled, meaning they understand 64-bit pointers or longs. In such cases, the library code typically passes these by value in adjacent registers. The kernel code understands to parse the input as such. The shmat() call is an example of a service that understands a 64-bit address in adjacent registers.

If the kernel is going to look at the data being passed, the library routine allocates 32-bit versions of the same structure(s) and copies the data, field by field, through assignment statements. This results in automatic truncation of the int and long fields, some of which may need testing for magnitude before the conversion is done. Pointers are converted as described in Section 3.2.2.2, “64-Bit to 32-Bit Address Remapping” on page 85.

If the kernel fills in data as the result of a system call, the data must be widened by the library routine on return from the kernel extension. Space for the 32-bit version of the structures to be filled-in must be allocated by the library routine before calling the kernel extension. Assignment, field by field, will do the proper widening (zero extension or sign extension, as appropriate).

Returned pointers, such as from sbrk() or shmat(), require special handling between the library routine and kernel extension routine to ensure that the proper 64-bit values are returned to the library routine.

Note that some system calls involve data passing in both directions, into and out of the kernel, and thus require action on the part of the library routine before, and after, the system call.

3.2.2.2 64-Bit to 32-Bit Address Remapping

As shown in Figure 12, the PowerPC architecture divides the Effective Address (EA) into three fields:

- The Effective Segment ID (ESID)
- A 16-bit page number within the segment
- A 12-bit byte offset within the page

The width of the ESID varies with execution mode:

- In 32-bit mode, the ESID is 4 bits and is often referred to as the Segment Register number.
- In 64-bit mode, the ESID is 36 bits.

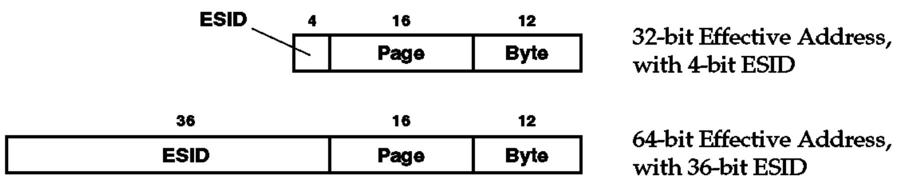

ESID
4
16
12
Page
Byte
32-bit Effective Address, with 4-bit ESID
36
16
12
ESID
Page
Byte
64-bit Effective Address, with 36-bit ESID

Table 12. Effective Segment IDs in 32-Bit and 64-Bit Mode

For each process, the kernel maintains an array of sixteen entries, each holding a segment register value that defines the mapping of user effective addresses to system virtual addresses at the point of a system call. When a 32-bit application issues a system call and passes pointers to data in user space, the entries in this array are loaded into segment registers by the copyin() and copyout() routines to access user-space data. The kernel never accesses user-space data directly, but only through copyin() and copyout().

The 32-bit AIX kernel does not generally understand 64-bit address spaces. To perform kernel services for 64-bit address spaces, the 16-element mapping array previously mentioned must be set up so that the relevant user-space data for a system call can all be accessed through 32-bit addresses.

The VMM provides services to remap 64-bit pointers passed through system calls into a 32-bit address space. The 64-bit library code fills out a data structure describing the 64-bit pointers to be remapped using one of the __remap*() services. This structure is passed to one of the remap*_64() kernel services by the 64-bit kernel extension to remap the specified segments in the user address space into a 32-bit address space for use by the normal kernel system call routines.

The general form of the remapping structure is an array of effective segment identifiers and an integer indicating the number of such ESIDs to be remapped. A pointer to this remapping structure is passed to the remap*_64() routine. An ESID is 36 bits long and is represented as two integers; the high-order 32-bits of the ESID in the first word, and the remaining four bits of the ESID in the high-order four bits of the second word. To optimize the remapping of a single ESID, the two words describing the ESID are passed in registers rather than passing a pointer to a remapping structure. Additional bits are defined in the unused bits of the second word to indicate which form is being used – pointer or inline registers.

Remapped addresses are only valid in the kernel for the duration of the system call. Subsequent system calls may not rely on 32-bit remapped addresses from previous system calls. Extensions with such requirements should save the segment register value and use the long-form virtual address instead or call as_unremap64() to obtain the 64-bit unremapped address to store.

3.2.2.3 Remap Library Data Structures

All the remap data structures are defined in the header file <sys/remap.h>, except the kernel MST-extension remap region that was discussed earlier.

This header file is shipped with AIX because all of these structures may be required by the user.

The addr struct is a library-side structure that holds the information relevant to the remapping of the address of a single 64-bit area. It also holds the size of the structure pointed to. Segment crossings are allowed for items in the addr struct, with the limitation that a single item to remap may not be contained in more than fifteen segments. This limits the maximum size of an object to be remapped to 3.75 GB, assuming that it is aligned to a segment boundary.

The uremap struct is a library-side structure that consists of an array of addr structs and an integer giving the size of the array. In the general case, the number of entries can be quite large, such as the number of addresses of environment variables passed on execve(). The size is set at 17 in the structure definition to cover cases that need to pass this many addresses (writev() is one such case). Cases with more than 17 addresses require individually-constructed structs similar to uremap or just the C-allowed use of an index greater than an array-size, with appropriate storage malloc()'d by the caller. The naddr field of the uremap struct can accommodate large numbers of addr structs input. The point is that __remap() is not limited to 17 addresses to remap on one call. It can accept a large quantity. As of this writing, the upper bound has not been set. The number 17 was specifically arrived at from the iovec struct. 17 is iovcnt+1 to allow for the vectors plus the address of the structure itself.

The remap struct is the basic data structure used to communicate remapping information from the library side to the kernel extension. A single remap struct contains an unremapped 64-bit address (actually ESID), and additionally, the low bits of a remap struct (address) also have meaning. This will be discussed later.

The kremap struct is an array of remap structs plus an integer giving the number of array elements actually used. In the general case, this structure is passed by the library side to the kernel extension, which accesses it through copyin64(). The elements of the r[.] array represent up to fifteen unremapped 64-bit addresses.

The remapping code gets the right answer implicitly by making sure that the library remap code and the kernel remap code allocate srnos in the same order while processing the kremap struct. It is particularly important, for segment crossings, that adjacent segments become allocated in the kernel as needed. The r_xingcnt field in the remap struct is used to indicate the number of segments that a particular mapping crosses, so the segments may be created adjacently.

It is important not to confuse the uremap and kremap structures. The uremap structures are where the full 64-bit to 32-bit address translations are kept for each 64-bit address remapped. The kremap structs are passed to the kernel and indicate on a segment-basis which 64-bit ESIDs are remapped to which 32-bit sreg numbers. The mapping of entries in a uremap struct to entries in a kremap struct is typically many-to-one. For example, two pointers in the same ESID will create two uremap entries and one kremap entry.

3.2.2.4 Remap Library Programming Interfaces

The remap library services set up the remap data structures to be passed to the kernel remap services. There are three library remapping interfaces:

```
union __remap_handle __remap(struct uremap *up, struct kremap *kp)
struct remap __remap1(struct addr *ua)
void __remap2(struct addr *ua, struct kremap *kp)
```

Three interfaces exist, rather than just one, to improve performance on remapping. For the case of syscalls that require only one or two parameters to be remapped, the parameters may be passed in registers. This avoids the copyin() to the kernel by remap_64() of a kremap struct and is a significant performance saving. So, __remap1() or __remap2() should be used when only one or two parameters require remapping, respectively. The __remap() call should be used for syscalls requiring three or more remappings.

Like other things in 64-bit libc.a, __remap() is only available to applications running in 64-bit mode and linked with the correct library. The input to __remap() are the 64-bit addresses to remap in the addr structs (in uremap). The __remap() code fills in the addr structs with remapped 32-bit addresses. It also fills in the kremap struct with the 64-bit ESIDs in the slot corresponding to the 32-bit sreg number that __remap() picked for the remapping. Slot 0 of the kremap struct corresponds to 32-bit sreg 1. This is because sreg 0 is never given out for reasons previously stated having to do with NULL pointers. As such, only fifteen different segments can be remapped with one kremap struct.

There is no formal provision at the moment for remapping a given user-mode address to a specific remapped address specified by the library. It is expected that applications can avoid dependencies such as this. In the event that this is not possible, two alternatives exist; add a new force-remap service, or create a kremap structure with the specified remapping as though it had already been processed by __remap().

3.2.2.5 Remap Optimization for Multiple Addresses

If more than one item must be remapped, there is a folding optimization done by __remap(). For the second, and subsequent addresses in the uremap struct, __remap() determines whether the ESID of the address being operated on has already been remapped. If so, there is no need to make another kremap entry for it.

The preceding works fine for circumstances where segment crossings are not involved. It does, however, become complicated when you put segment-crossings into the equation. Assume that __remap() is to process the uremap struct in array-order. The addresses will not be sorted. Also assume that the first parameter, which does not cross a segment boundary and resides in 64-bit ESID 3, becomes remapped to 32-bit srno 1. If the second parameter does cross a segment boundary and starts at 64-bit ESID 2, in the absence of any special optimizations, this second parameter would have to be remapped to 32-bit srnos 2 and 3 (0 is unavailable for use). Note that the second parameter must occupy adjacent 32-bit segments.

There are several ways around this problem. One way is to have __remap() presort the input addr structs by ascending address, but this could get expensive, since it could have a virtually unbounded number of input address structures to process. The actual __remap() implementation is a two-pass method. The first pass (for a majority of the cases) takes a simple approach. In the preceding example, it would simply associate the 32-bit srnos 1, 2, and 3 as explained. As long as the rest of the remappings fit in the maximum fifteen slots that are available, this is not a problem, and there is no expensive sorting or post-processing required. The simple first pass will do folding only as far as checking to see if the 64-bit ESID has been remapped.

It is also possible to have a situation similar to the preceding example, except that the second remapping could touch fifteen segments, starting with ESID 2. This case would fail the first pass, and a second pass becomes necessary to sort them into ascending order. Although this second pass, when required, does add some overhead, it guarantees that the ranges to be mapped will fit into fifteen segments.

3.2.2.6 Remap Kernel Programming Interfaces

There are remapping programming interfaces that are exported to kernel extensions from the kernel. These are primarily for use by the 64-bit kernel extension; however, applications that add their own system calls will need these as well. For this reason, these services take up regular namespace and are not prefixed with an underscore. See the man pages for formal documentation of these routines as_unremap64() is intended for kernel extensions that really need to have the unremapped address.

The library-side maps 64-bit addresses to 32-bit in that it selects the sreg numbers to correspond to the ESIDs of the 64-bit addresses. However, the kernel-extension still has to update the address space map for the 64-bit thread to reflect these values, so that when the kernel uses these 32-bit addresses to access memory, the proper SID will be inferred. These services update the MST-extension remap struct described previously:

```
int      remap_64(struct remap r)
int      remap1_64(struct remap r)
int      remap2_64(struct remap r1, struct remap r2)
unsigned long long as_unremap64(uint addr32)
```

The reason for three different remap routines is to optimize for cases where only one or two ESIDs are used. This is a very common case. The remap_64() call handles the case of greater than two remap structs input as well as all other cases. Parameters to remap1_64() and remap2_64() are passed entirely in registers, so these routines do not have to copyin() a remap struct from user-mode.

There is also an internal kernel routine, remap(), that is used by copyin64() and others. The purpose of remap() is to support 64-bit-enabled code in the kernel that handles non-remapped addresses. Code calling remap() passes the address of a remap-like struct on the stack (typically), and this is used as a save area for the regular MST remap fields that are overwritten by this remap() call. At the end of the copyin64(), or any other call, restoremap() is called with the address of the stack-resident remap-like struct. The original contents of the MST remap fields that were temporarily overwritten are written back to the MST. An additional internal routine, as_remapaddr(), is used to return the original 64-bit unremapped address (modulo SEGSIZE) for a given 32-bit remapped address.

```
static void remap(ptr64 addr, uint nbytes, struct remaps * ptr)
static void restoremap(struct remaps * ptr)
ptr64       as_remapaddr(uint addr32)
```

These services take only one address and length to remap, and there is no place, currently, where calls to remap() are nested without restoremap(). Only one data structure at a time can be remapped in this fashion. The on-stack remaps structure is able to store the entire mstext->remaps array when there are fifteen segments total in the range to be remapped through savemap().

3.2.2.7 Optimizations for One or Two Parameters

The library-side and the kernel-side remap routines have optimizations built into them to pass one or two parameters by value if possible. This provides a significant improvement over requiring a full copyin64() of the kremap structure every time. There are three cases that the remap() code has to handle:

- The output library remapping resulted in only one remapping or remap struct (R_IN_LINE - see the following).
- The output remapping resulted in two remappings, and __remap2() was called (with two addr structs only).
- The output remapping resulted in more than one remapping if __remap() was called.

The low order bit of the cookie passed to the kernel remap services, all the way from the library, identifies what the cookie actually is. For example, although remap_64() is documented to have a struct remap() passed to it, if the R_IN_LINE bit is not set in this, then this struct remap is a pointer to a struct kremap, on which a copyin64() must be performed. If, however, the R_IN_LINE bit is set on input, then that indicates that all of the parameters collapsed into a single remapping in the library (case 1), and remap_64() can call remap1_64() to do the work. This avoids the copyin() when many parameters reside in the same segment (ESID). This should be a common case.

The R_IN_LINE flag is interpreted differently by the remap2_64() kernel service. The __remap2() subroutine sets the R_IN_LINE flag in the first remap struct if the output remappings collapsed into one. This indicates that the second parameter to remap2_64() should be ignored. See the next section for coding and calling conventions for __remap2().

3.2.2.8 Using the Remapping Services

There are very rigid rules on how to invoke the remapping services and how to handshake with the kernel-side remapping wrappers. Typically, whatever library remapping service was invoked for a particular wrapper, it should have the corresponding kernel remapping service called as well. For example, if __remap2() is called in the library wrapper, the kernel-extension should call remap2_64().

The following is very important:

- For a given system call, it is only permissible to make a single call to the remapping kernel services. In other words, it is not legal to call remap_64() twice on the same system call. Similarly, it is not legal to call remap1_64() and then remap2_64() on the same system call.

To detect potential misuse of the remapping services, the remap data structures are coded with a unique code for whichever library remap service created them. If a kernel service other than the correct one for a given library service is called, an error will result. For example, callers of __remap1() must call remap1_64().

Typically, a library wrapper will call a remap service and check if the return code is -1. If so, it will fail. After that, the returned value remap structure from the particular __remap* service is passed as the first parameter. If this is a __remap2() call, the first parameter is kremap.r[0], and the second parameter is kremap.r[1]. Of course, each of these parameters is split into two 32-bit registers. A single remap struct fits into a single 64-bit register in user-mode but requires two registers each to pass to the kernel.

3.2.3 64-Bit XCOFF Format

The extended common object file format (XCOFF) combines the standard common object file format (COFF) with the TOC module format concept. This allows dynamic linking and replacement of units within an object file.

Until AIX 4.2, the XCOFF format assumed that addresses were 32 bits, memory could be no larger than 2^{32} bytes, and therefore, no object (csect, text section, and so on) could be larger than 2^{32} bytes. The XCOFF header files contained 32-bit fields for *value* in the symbol table, 32-bit length-fields for sections, and so on.

For PowerPC’s 64-bit execution mode, these sizes are too restrictive. The reason for moving an application from 32-bit mode to 64-bit mode is to use addresses larger than 32 bits. In general, this means that all fields in XCOFF structures that can hold an address or a size should be increased to 64 bits. At the very least, it should be possible to describe a *bss* (common, or uninitialized data) object with a size greater than 2^{32} bytes.

3.2.3.1 XCOFF Design

There are two XCOFF formats; one for 32-bit applications, and one for 64-bit applications. The 64-bit XCOFF differs from the 32-bit XCOFF format in several ways, as listed in the following:

- The file size can be up to 2^{63} bytes (rather than 2^{31} bytes).
- Each XCOFF section can be up to 2^{63} bytes (rather than 2^{31} bytes).

- The virtual addresses in the process can be up to 2^{64} bytes (rather than 2^{32}).
- The offsets of objects within the XCOFF file can be up to 2^{63} bytes (rather than 2^{31}).
- Line numbers can be up to 2^{31} (rather than 2^{15}).

The 64-bit XCOFF does not differ from the 32-bit XCOFF format in the following fields because they are considered to be big enough:

- Symbol table indexes will still be limited to 2^{31}, allowing about half this many symbols in an executable (each symbol uses an average of about two symbol table entries).
- The string table will be limited to 2^{31} bytes in size, limiting the sum of the length of all symbol names to less than this value.

The following design issues have been implemented:

- All header file declarations for both 32-bit XCOFF and 64-bit XCOFF are contained in the same files used for 32-bit XCOFF declarations.
- Field names within structures are the same for structures that have different versions for 32-bit XCOFF and 64-bit XCOFF.
- Where possible, the structure sizes and layouts from the 32-bit XCOFF definition were not changed. Some fields have been rearranged to avoid alignment padding and to increase the number of fields that are at the same offset in both XCOFF versions.
- Fixed width types are used in all header files. The following are the *fixed width* types: char, short, int, and long long. (Of course, these types are only fixed with respect to AIX.)

  Note: Pointers exist in some of the existing header files. Since pointers are not fixed-width types, source code using these pointers will not compile in 64-bit mode.

- Source code compatibility is maintained for 32-bit programs written to process 32-bit XCOFF files.
- Minimal changes are required to port a 32-bit program that manipulates 32-bit XCOFF files to a 32-bit program that manipulates 64-bit XCOFF files.
- Minimal changes are required to port a 32-bit program that manipulates 32-bit XCOFF files to a 64-bit program that manipulates 32-bit XCOFF files.

3.2.3.2 Using the XCOFF Formats

There are different options for an application to use the XCOFF formats. The following strategies are possible:

- Using 32-bit XCOFF declarations.

  To only use the 32-bit XCOFF definitions, an application must include the appropriate header files. This will define only the structures for 32-bit XCOFF files. The 64-bit XCOFF structures and field names will not be defined. Structure names and field names will match those in previous versions of AIX, providing source compatibility.

  Note: Existing uses of shorthand type notation (for example, uint, ulong) have been removed.

- Using 64-bit XCOFF declarations.

  To only use the 64-bit XCOFF definitions, an application must define the preprocessor macro __XCOFF64__ . This will define only the structures for 64-bit XCOFF files. The 32-bit XCOFF structures and field names will not be defined. Structure names and field names will match the 32-bit XCOFF versions.

- Using both XCOFF declarations.

  To use separate 32-bit XCOFF and 64-bit XCOFF definitions, an application must define both the preprocessor macros __XCOFF32__ and __XCOFF64__. This will define structures for both kinds of XCOFF files. Structure and typedef names for 64-bit XCOFF will have the suffix _64 added to them, even if a common structure could be used.

- Using a hybrid of both XCOFF declarations.

  To use a hybrid of both the 32-bit XCOFF and 64-bit XCOFF definitions, an application must define the preprocessor macro __XCOFF_HYBRID__. This will define single structures that can be used with both 32-bit XCOFF and 64-bit XCOFF, where possible. Where fields in structures are a different size or at a different offset, suffixes 32 and 64 are used to differentiate between the fields. For example, the symbol table definition (in /usr/include/syms.h) will have the names n_offset32 and n_offset64, which should be used for 32-bit XCOFF and 64-bit XCOFF files respectively.

Depending on the execution environment of the executable and the targeted XCOFF format, the following combinations exist:

- 32-bit program manipulating 32-bit XCOFF files.

  A 32-bit program that manipulates 32-bit XCOFF files will require no change to continue to do so with the new header files.

Note: Since the types of some fields are being changed from long to int, code that takes the address of such a field will result in a compiler warning when compiled in ANSI mode.

- 32-bit program manipulating 64-bit XCOFF files.

An existing 32-bit program that manipulates 32-bit XCOFF files can be recompiled to manipulate 64-bit XCOFF files by defining the symbol __XCOFF64__. Some code changes will be necessary, but the changes with respect to the file format will be limited to cases where 32-bit XCOFF and 64-bit XCOFF use different constructs. In particular, n_name will not be defined in struct syment, and use of struct auxent will require changes since auxiliary symbols are redefined.

- 64-bit program manipulating 32-bit XCOFF files.

  An existing 32-bit program that processes 32-bit XCOFF files can be recompiled to a 64-bit program without change (with respect to the XCOFF definition) with two exceptions:

  - Pointers in the existing XCOFF files will be defined as ints in 64-bit mode.
  - Existing header files use preprocessor macro definitions in some cases. These same macros may no longer exist when compiling in 64-bit mode, so incidental use of the macros may require a code change.

3.2.3.3 Incomplete aouthdr Structure

Non-executable XCOFF files do not require a full-size auxiliary header. Current practice defines a short 32-bit auxiliary header that is generated by the compiler or the linker when the output file is not an executable. A short 64-bit auxiliary header will not be required by this definition. Applications examining non-executables must examine f_opthdr in the XCOFF header to determine how much of the auxiliary header is in the file.

There will be no auxiliary header used for non-executable 64-bit XCOFF files. Applications needing the fields from the auxiliary header for non-executable 64-bit XCOFF files should use the information in the section headers to generate these values. The fields where this may be necessary are text, data, and BSS sizes.

3.2.3.4 XCOFF Magic Number

The calling conventions for 32-bit mode and 64-bit mode are different in detail because one saves 32-bit General Purpose Registers (GPRs) onto the stack frame, and the other saves 64-bit GPRs. Calling from a program of one mode to a subroutine of the other mode is not supported. The linkage editor ld refuses a request to link programs of differing execution mode.

Because of this, a new magic number has been introduced for 64-bit execution mode. The primary purpose of the XCOFF magic number is to identify the associated Application Binary Interface (ABI), which implies a hardware system and an execution environment. The 64-bit XCOFF magic number implies a 64-bit PowerPC processor and 64-bit execution mode on that processor.

The magic number keeps the linkage editor from binding 64-bit programs with 32-bit programs and keeps the loader from trying to execute 64-bit programs on 32-bit hardware.

The magic number is defined in the header file /usr/include/filehdr.h and has the name U803XTOCMAGIC with the value 0757.

3.2.4 Device Drivers

AIX 4.3 supports 64-bit applications on 64-bit PowerPC hosts in addition to maintaining support for 32-bit applications on all other supported hosts. Thus, on 64-bit hosts, both 32-bit and 64-bit applications can run simultaneously. To minimize the impact of adding 64-bit support, the kernel continues to run in 32-bit mode but provides interfaces to 64-bit applications by remapping the 64-bit application space address into a 32-bit address for the kernel. Thus, the following is true for device drivers in general and I/O drivers specifically:

- 32-bit versions of device drivers will operate correctly without change on AIX Version 4.3 in support of 32-bit applications.
- 64-bit applications require modification of only the entry points (such as ioctl()s) for proper operation.

The 4/8/8 model requires two primary changes for an I/O device driver:

- Providing ioctl support for 64-bit applications.
- Ensuring that structures describing fixed sized entities are size-invariant between both 32-bit and 64-bit applications.

3.2.4.1 Changes to ioctl()

The third argument of an ioctl call is referred to as the *arg* parameter. For some ioctls, the arg parameter can be a pointer. For AIX 4.3, the kernel guarantees that the arg parameter received by a device driver is always a 32-bit value. For 64-bit applications, the kernel will remap the address to a 32-bit address. Often, the arg parameter is a pointer to a data structure that may contain additional pointers. The kernel has no knowledge of this and, as a result, it is the device driver's responsibility to interpret these correctly. Device drivers that support 64-bit embedded pointers need to notify the kernel of this by setting the DEV_64BIT define for the d_opts flag passed to the devswadd() call from the config entry point of the device driver. For example a 64-bit-enabled SMP driver would use the following code segment:

```
     devsw_struct.d_opts = DEV_MPSAFE | DEV_64BIT;
     devswadd(devno,&devsw_struct);
```

For device drivers that do not set the DEV_64BIT flag, all ioctls from 64-bit applications will fail with an errno of EINVAL.

Since data structures with embedded pointers cannot remain size-invariant between 32-bit and 64-bit applications, a 64-bit-enabled device driver will need to maintain an internal-use-only 64-bit equivalent (recall the device driver will be compiled for 32-bit mode) of all such structures that can be passed as arg parameters. This can be accomplished by cloning the structure definition and replacing all pointers with type ptr64 (defined in types.h as unsigned long long).

3.2.4.2 Parameter Passing Examples

For example, assume a device driver's shipped header file has struct A, and a device driver supports an ioctl call whose arg parameter is the address of struct A:

```
struct A {
     int        aaa;
     char       *bbb;
     char       c;
     int        *ddd;
}
```

For a 32-bit application using struct A as the arg parameter of an ioctl, the device driver can recast the arg parameter as struct A. However, if the device driver determines the caller is a 64-bit application (through a call to the IS64U kernel macro), then the device driver will have to recast struct A to a new struct A64 defined as:

```
struct A64 {
     int         aaa;
     ptr64       bbb;
     char        c;
     ptr64       ddd;
}
```

The code segment of the device driver for this ioctl is similar to this:

```
          .
          .
          .
     if (IS64U) {
          /* The caller is a 64-bit application */
          x = (struct A64 *) arg;
          .
          .
          .
     } else {
          /* The caller is a 32-bit application */
          x = (struct A *) arg;
          .
          .
          .
     }
```

The following naming conventions are used to create the device driver's 64-bit equivalent structure:

- The 64-bit equivalent structure used by the device driver is included in the same shipped header file.
- It has a comment indicating that this is used for device drivers only and not applications.
- The name of the 64-bit equivalent structure is that of the original structure but with a 64 appended (for example, for sc_iocmd, it will be sc_iocmd64).

For the device driver to manipulate the 64-bit addresses, new 64-bit kernel services are provided. These kernel services support 32-bit unremapped addresses as well as 32-bit remapped addresses and 64-bit addresses. Thus all 64-bit-enabled drivers make the global kernel service replacements provided in Table 13.

Table 13. Old and New Kernel Services Used by Device Drivers

Old Kernel Service	**New Kernel Service**
copyin()	copyin64()
copyout()	copyout64()
xmattach()	xmattach64()
pinu()	xmempin()
unpinu()	xmemunpin()

The xmempin()/xmemunpin() calls use the same arguments as pinu()/unpinu(), with the exception that the third argument for xmempin()/xmemunpin() is the address of the cross memory descriptor from xmattach()/xmattach64() instead of a segflag. The xmattach64() call uses the same arguments as xmattach(), with the exception that the first argument for xmattach() is of type unsigned long long (ptr64) instead of type char *.

Note: xmattach64() enables xmempin(), so that the 64-bit address can be recast as a 32-bit address when passed to xmempin(). The xmdetach() call is used to undo xmattach64().

3.2.5 Loader

For AIX 4.3, the same basic loader system calls are provided to 64-bit programs. That is, there are 64-bit versions of execve(), fork(), exit(), load(), and unload() that are aware of the 64-bit user address space. There are also 64-bit versions of knlist() and sysconfig(), although these just interface to the existing 32-bit services. There is no 64-bit version of ptrace(), but 64-bit processes can be debugged by 32-bit debuggers. Finally, the loadquery(), loadbind(), and __loadx() functions are no longer system calls for 64-bit programs but are implemented in libc.a.

Four new external functions are added to the loader in AIX 4.3 to support 64-bit processing:

- ldr_init64()

  This function is called during kernel initialization when a 64-bit system boots.

- ldr_config64()

  This function is called by sysconfig() when a 64-bit machine is configured to run 64-bit processes.

- ldr_gettextusage64()

  This function computes the number of real memory pages used by the main executable of a process. It only needs to be called if the main executable is loaded in multiple working segments.

- ldr64()

  This is a new system call exported to the special 64-bit process that relocates shared objects. No other process has access to this system call.

On AIX 4.3, the same algorithm is used for loading 64-bit modules, but the work is split between kernel code and user-space code. The kernel part of the 64-bit loader is responsible for mapping the modules of a process into the

64-bit user address space. The user-space part processes the symbol tables and performs the relocation of the data sections.

Kernel extensions are still in XCOFF32 modules and they are entirely loaded and resolved by the kernel. The user-space processing of the shared library segments is handled by a privileged process running in 64-bit mode called the shared library loader assistant process or SHLAP.

The user-space processing of privately-loaded modules is handled by code that is loaded into a system-wide segment that is shared by all 64-bit processes. This code is called user-space loader assistant (USLA) and runs in the process of loading the module. The USLA is implemented as a loadable module that is prelinked at a fixed address, so that it will not have to be relocated. When an execve() system call leaves the kernel, it transfers control to the USLA that performs symbol resolution and relocation for privately-loaded modules. After load() calls, library code will be responsible for calling the USLA to complete the relocation of any newly-loaded modules.

Because the kernel is not performing symbol resolution and relocation for 64-bit processes, only a small portion of a 64-bit module needs to be addressable in the kernel. The kernel only needs to examine the XCOFF header, the section headers, the loader section headers, and the loader section import IDs. Even for extremely large programs, the size of these areas will be small. Only the import ID section is variable length, and its length depends on the number of dependents a module has, not on the size of the module itself. These portions of a module can be read into allocated memory, avoiding addressability problems inherent in the existing 32-bit loader.

3.2.6 Virtual Memory Manager

The AIX 4.3 design point is a 60-bit user address space. The areas impacted in the VMM are the address space code, the shared memory code, teaching VMM code to understand remapped addresses, and the remapping services themselves.

3.2.6.1 Executing a 64-Bit Program

The size of a user-address space only changes as a result of exec(). A 32-bit program may exec a 32 or 64-bit program, and conversely, all combinations are possible.

The VMM provides support routines for exec() processing to switch between a 32-bit and 64-bit executable. The routine vm_makeme64() is called when the exec()'d program is a 64-bit program. This routine pins and initializes the

64-bit u-block extension, initializes the 64-bit address space structures, initializes the Address Space Register (ASR) in the Machine State (MST) extension with the real address of the segment table, sets the sbreak and stack sizes, and marks the process as a 64-bit executable.

The routine vm_makeme32() is called whenever a 64-bit program execs(). It is called even if the program to be executed is 64-bits. This routine initializes a 32-bit user address space from the 64-bit one, clears the 64-bit MST extension address and marks the process as a 32-bit executable. The segstate structure is handled later by shm_asinit(), and the 64-bit u-block extension is freed in the subsequent call to vm_cleardata(). The vm_cleardata() call also initializes the sbreak value for the private segment and adjusts the storage protect key accordingly in the external page tables covering the user region. There is no service to re-initialize a 64-bit adspace to a newly-created 64-bit adspace, so it is necessary to call vm_makeme32, followed by vm_cleardata() and vm_makeme64(), when a 64-bit program exec()'s another 64-bit program.

3.2.6.2 Address Space Management

The address space management code is significantly impacted for 64-bits. The code was updated to understand segment numbers, or effective segment IDs, above the first sixteen IDs.

32-Bit Address Space Programming Interfaces

The following 32-bit services that operate on the process address space are exported to kernel extensions, so the AIX 4.3 versions of these services are binary-compatible with prior versions. For internal base kernel use, these address space services are extended to handle 64-bit address spaces but only by code that has been modified to be 64-bit aware. This means only by code that knows how to compute an appropriate adspace_t for a 64-bit address space:

```
caddr_t    as_att(adspace_t * adsp, vmhandle_t srval, caddr_t addr)
int        as_det(adspace_t * adsp, caddr_t addr)
vmhandle_t as_geth(adspace_t * adsp, caddr_t addr)
vmhandle_t as_getsrval(adspace_t * adsp, caddr_t addr)
void       as_puth(adspace_t *adsp, vmhandle_t srval)
void       as_seth(adspace_t * adsp, vmhandle_t srval, caddr_t addr)
adspace_t  *getadsp()
```

To provide compatibility for 32-bit kernel extensions, the 32-bit getadsp() kernel service is modified to determine if it is running under a 64-bit user address space, and if so, it will return the first adspace_t. This represents ESIDS 0-15. This could enable some extensions to run under the 4 GB boundary for 64-bit.

All of the 32-bit services listed may be used by a kernel extension or device driver, but they will only operate on addresses below 4 GB, even when under a 64-bit process. The service to compute an adspace_t for 64-bit, getadsp64(), is not exported from the kernel. Thus, these routines are not enabled outside the kernel to operate above the 4 GB line.

Kernel services and drivers should use the new 64-bit address space services described in the following.

For 64-bit address spaces (internal to the kernel, where there is getadsp64()), the address arguments specified preceding as caddr_ts are actually 32-bit quantities that are treated as offsets into the appropriate adspace_t. The only reason for keeping enablement of these services for 64-bit inside the kernel is that, on some system calls, there should be some performance improvement by only computing an adspace_t once.

64-Bit Address Space Programming Interfaces

The following additional address space services are provided for use by the 64-bit kernel extension and by other base kernel code that has been modified to be 64-bit aware.

All of the following services are exported:

```
unsigned long long as_att64(vmhandle_t srval, int offset)
int                as_det64(unsigned long long addr64)
vmhandle_t         as_geth64(unsigned long long addr64)
vmhandle_t         as_getsrval64(unsigned long long addr64)
int                as_puth64(unsigned long long addr64, vmhandle_t srval)
int                as_seth64(unsigned long long addr64, vmhandle_t srval)
int                IS64U
```

The address space programming model for 64-bit introduces a copy of all the 32-bit interfaces appropriately scaled for 64-bit addresses. All of the 64-bit services work properly under a 32-bit user address space or under a kproc as well as a 64-bit user address space:

One additional non-exported service is provided:

```
adspace_t   *getadsp64(unsigned long addr64)
```

The getadsp64() service exists to provide a bridge between the 64-bit and 32-bit services. The adspace_t returned by getadsp64() may be passed to any of the 32-bit services, and it will work properly. The one exception to the example is that 32-bit as_att() will not support attaching anywhere other than the first adspace_t. The advantage of using getadsp64() is the performance improvement of only computing an adspace_t once per system call.

The concept of an adspace_t really does not exist with the new 64-bit interfaces; getadsp64() is the only exception, and it is only for use with the 32-bit interfaces internal to the kernel. The advantage of not having an adspace_t externalized is that the width of an adspace_t is no longer surfaced to extensions. This saves the extensions the overhead of having to compute another adspace_t every 4 GB.

Kernel extensions writing new code to enable 64-bit support that need the address space services, should use the new *64 services instead of the current 32-bit services. The *64 services handle all the general cases for 64-bit and 32-bit address spaces. The 32-bit services will only work for addresses less than 4 GB outside the kernel.

How to Determine if this is a 64-Bit Address Space
The IS64U macro will return true if the user address space for the current process is 64-bits. This macro is valid only in kernel mode. If used inside the kernel, this will return the value of U.U_64bit directly. If used outside the kernel in an extension, this will generate a subroutine call to the new kernel service: _as_is64(). _as_is64() will simply return the value of the variable. U.U_64bit is managed by exec() in vm_makeme64/32. IS64U is defined in user.h.

3.2.6.3 Shared Memory Management
The shared memory code is impacted for 64-bit support since it must attach shared memory segments at large addresses. The functions shmat() and mmap() will behave as follows regarding segment number (ESID) allocation:

- If no fixed address is specified, then allocation takes place from the shmat()/mmap() pool at ESIDs: 0x70000000 - 0x7FFFFFFF.
- If a fixed address is specified by the user, then the allocation will be allowed as long as the ESID is less than 0x80000000 and is not ESID 0-2,13, or 15.
- The shared memory allocator internally allocates anywhere in the address space. The reason for this is that other areas of the kernel, for example the loader, need to insert segments at ESIDs greater than 0x80000000. Therefore, shm_insert() is allowed to insert anywhere, but the higher-level user-interfaces perform the validation for the user level.

Shared Memory User and Exported Kernel Programming Interfaces
The shared memory component has numerous system calls surfaced to the user. All of these system calls are registered in the 64-bit libc.a and the 64-bit kernel extension. The following is the list of the 32-bit shared memory system calls:

```
void* shmat(int shmid, const void *address, int shmflag)
int   shmctl(int shmid, int command, struct shmid_ds *buffer)
int   shmdt(const void *address)
int   shmget(key_t key, size_t size, int shmflag)
int   disclaim(char *address, uint len, uint flag)
```

The following are the new, 64-bit-ready interfaces for the shared memory services. These interfaces are called directly from the 64-bit kernel extensions only:

```
ptr64   _shmat64(int shmid, ptr64 address, int shmflag)
int     _shmdt64(ptr64 address)
int     _disclaim64(ptr64 addr,unsigned len,unsigned flag)
```

The _shmat64(), _shmdt64(), and _disclaim64() calls do not require parameter remapping, since they are 64-bit-enabled. They do, however, require 64-bit libc.a to split their address parameters into two adjacent general purpose registers for processing in 32-bit mode. the shmctl() call does require parameter remapping on the pointer to the shmid_ds. Additionally, shmget() takes a size_t as input. This typedef is an unsigned long, which has different widths in 32-bit and 64-bit programs.

The prototype to shmget() will not change for 64 bits. The low 32 bits of the 64-bit size will be passed to the kernel, with the size being range-checked for 32 bits in the library-side. There will be no increase in size of the supported memory region. The key_t parameter to shmget() is currently a long. It will be changed to always be an int. This will be true for 32-bit and 64-bit code to make it invariant. This allows predictable message-passing between 32-bit and 64-bit processes.

3.2.6.4 User Data and Stack Management

The 32-bit programming interfaces for adjusting a program's data size are brk() and sbrk():

```
int     brk(void *enddatasegment)
void *  sbrk(int increment)
```

There is a change to the sbrk() interface required by 64-bit mode and UNIX98 standards. For UNIX98, sbrk() needs to take a long on input.

This poses a breakage for those who want to have the UNIX95-behavior of sbrk() that obeys the preceding prototype, taking an int in 64-bit mode. The problem is that the 64-bit library wrapper for sbrk() has no way of determining whether it was passed a 32-bit value or a 64-bit value. The compiler will not ensure that the high 32-bits of a register are 0 for int's. Since most programmers going to 64-bit with their applications will require some porting effort to do so, changing sbrk() to the UNIX98 interface will add very little extra work. This does not mean that they have to change everything to UNIX98 conformance, just sbrk() in this case.

The sbrk() function prototype in <sys/unistd.h> was changed to pass a long for all 64-bit compilations; that is, if __64BIT__ is set. For 32-bit compilations, the sbrk() prototype will be conditionally compiled to generate the appropriate UNIX95 or UNIX98 prototype since the data-width between int and long does not change for 32-bit. Code that wants to run 64-bit must either make sure it passes a long, or if it obeys UNIX98, it is required to include <sys/unistd.h>. The standards require header file inclusion. The prototype for sbrk(), which defines it as taking a long, is as follows:

```
void *   sbrk(intptr_t increment)
```

`intptr_t` is a new type. Defined in <sys/inttypes.h>, it maps to a long.

3.3 Application Development

Section 3.2, “64-Bit Core Design” on page 75, explained the design issues of AIX 4.3 with respect to 64-bit application support. The changes in the core design of AIX have impacts on various components of the software development environment. This section describes what decisions have been made to provide a migration path from 32-bit to 64-bit applications. It shows the modifications that have been made to the most important tools in the software development area, such as the compiler, linker, and archiver.

3.3.1 C Compiler

This section discusses the implementation of 64-bit capabilities in the C for AIX compiler. The C compiler provides supporting functions that can enable the usability of 64-bit C syntax and semantics.

Each program compiled for execution on AIX is intended for execution in one particular *target execution mode*: 32-bit mode or 64-bit mode. The default compilation and assembly mode is 32-bit. This is the ILP32 model. The change from the default 32-bit to 64-bit mode is under user control. In the compiler, the option -q64 is used to change the compilation mode.

The default execution mode is not directly controllable by the user processes but can be examined indirectly (for code dynamically targeted to multiple environments) through the pointer or long type size. The compiler provides porting assistance options wherever there are statements that can be ambiguously interpreted for the LP64 environment.

When running 32-bit processes on 64-bit platforms, the execution is transparent and identical to executing on a 32-bit platform with no loss of performance. When trying to run 64-bit processes on 32-bit platforms, the execution will fail in an obvious manner.

The 64-bit implementation in the C front end does not change the default behavior of the compiler. The compiler only changes the behavior of code when compiled in 64-bit mode. Code that was compiled in 32-bit mode that has no requirements for large address spaces (pointers) or large object sizes (arrays and dynamic heaps) will not need to be recompiled to work in 32-bit mode on a 64-bit platform. You may recompile the code in 64-bit mode to check performance implications in 64-bit mode on a 64-bit platform.

While most code will recompile and execute properly in 64-bit mode, some code will behave differently, or may not function at all, due to nonportability deliberately or accidentally written into the code. Common causes of behavior changes are due to mixed use of long and int types in operators, especially comparison operators that will change the code execution path. Although the usual operand promotion rules do not change, the changed size of long types may yield surprising and unexpected results. Function arguments and return types from functions need to be checked for their actual value. Many library functions return long types and take long types that are implicit, such as size_t and ptrdiff_t. Structures, structure alignments, member alignments, bit fields, and enums are guaranteed to change when compiled in 64-bit mode (especially if they contain long and pointer types).

3.3.1.1 Compiler Mode

The generation of 64-bit instructions and 64-bit XCOFF is called the 64-bit compilation mode. The compiler invocation for setting the 64-bit versus 32-bit mode evaluates several sources. They are:

- Internal default
- Environment variable OBJECT_MODE
- Configuration file
- Command line
- Source file

The compiler evaluates the options in the given order of the items. The last one takes precedence. For example, if the environment variable OBJECT_MODE exists, it will replace the internal default of the compiler.

Table 14 provides a list of OBJECT_MODE settings and the compilation mode behavior.

Table 14. Settings for OBJECT_MODE and the Resulting Compiler Behavior

OBJECT_MODE Setting	**Compilation Mode Behavior**
not set	32-bit mode
32	32-bit mode
64	64-bit mode
32_64	fatal error and stop (unless there is explicit user setting in the config file or command line) with message: `1501-054 OBJECT_MODE=32_64 is for mixed-mode and is not a valid setting for the compiler.`
anything else	fatal error and stop (unless there is explicit user setting in the config file or command line) with message: `1501-055 OBJECT_MODE setting is not recognized and is not a valid setting for the compiler.`

This option allows the code to function in a 32- or 64-bit environment without excessive use of new option names. This will maintain compatibility with other tools that can exhibit 32/64-bit mode behavior since they will all use the OBJECT_MODE environment variable. It also maintains compatibility with machines without 64-bit capability that want to compile in 64-bit mode. In all cases, the user is free to override the environment variable with an explicit option in the config file or the command line.

32-bit mode is invoked by specifying -q32 on the compiler command line and is the default if OBJECT_MODE is not set. This option is equivalent to a direct expansion into the -qarch=com option. For the compilers that do not have 64-bit yet, use of the -q32 or the -q64 option will cause the following warning (this is the usual warning on unrecognized options):

```
1501-055 Option -q32, -q64 is not recognized and is ignored.
```

Problems with #pragma arch Suboptions in Source Files

The -q32/64 option has no pragma equivalence because the compilation mode must be determined before the compiler driver exits and invokes the compiler. Implicitly expanded options are parsed with the rest of the command line to produce a final compilation mode. From this compilation mode, the options are passed separately to the compiler, linker, and assembler. However, since the ARCH suboption has an equivalent #pragma arch suboption in the source file, the individual files may be compiled in a different mode than what was decided by the command line.

It was decided to disallow the setting of a #pragma arch suboption in a source file. This is a change in Version 4.0 of the C compiler that means a loss of backward compatibility with previous C compiler versions.

Mixed-Mode Compilation and Two-Step Compile and Linking

When you cause a mixed 32- and 64-bit compilation mode, your XCOFF objects will not bind. This will become obvious if the compile and link occurred in one step. However, you may not know this if the compile and link occurred in different steps. In other words, if you compiled and produced 64-bit objects, you need to remember to link using the 64-bit mode (when linking using xlc), otherwise the objects will not link. If the objects are in mixed XCOFF mode, then they will never link, and you must recompile completely, making sure that all objects will be in the same mode.

There is a set of new configuration file attributes that are used in place of the normal attributes whenever the compiler determines that the 64-bit mode is enabled. These new attributes are:

- crt_64
- gcrt_64
- mcrt_64

The new definitions for these attributes are:

crt_64 Path name of the object file passed as the first parameter to the linkage editor. If you do not specify either -p or the -pg option, the crt_64 value is used. The default depends on the compiler being used.

gcrt_64 Path name of the object file passed as the first parameter to the linkage editor. If you specify the -pg option, the gcrt value is used. The default depends on the compiler being used.

mcrt_64 Path name of the object file passed as the first parameter to the linkage editor if you have specified the -p option. The default depends on the compiler being used.

Note: The invocation of 64-bit mode using the -q64 option (either explicitly or using a stanza) automatically implies linkage in 64-bit mode. The compiler driver automatically and quietly generates the correct linker options (-b32 or -b64) to call the binder or the correct assembler option (-a32 or -a64) when calling the assembler. Therefore, these options do not need to be set by the user.

Predefined __64BIT__ Macro

When the compiler is invoked to compile for 64-bit mode, the preprocessor macro __64BIT__ is predefined. When it is invoked in 32-bit (default) mode, this macro is not defined. The variable can be tested through:

```
#if defined(__64BIT__)
```

or

```
#ifdef __64BIT__
```

to select lines of code (such as printf statements) that are appropriate for 64 or 32-bit mode. The ability to choose execution mode (of the final executable) at compile time and the existence of the __64BIT__ macro implies there is no need for an application to determine its execution mode at run time.

When the compiler is invoked to compile for 64-bit mode, this macro is set to a value of 1 internally, so that the C preprocessor and compiler will recognize it. It cannot be redefined or undefined. Any attempt at redefinition will fail.

3.3.1.2 Fixed-Width Types

There is a a set of types that maintain their width regardless of the compilation mode of the compiler. These types may be used if the program relies on an exact and unchanging size for the types.

Programs that exchange formatted messages are, for example:

- An X Windows server and client executing in different modes.
- Processes running in different modes that share data (using shmat() to jointly access and change common memory areas).
- Data files written by applications running in one mode and read by applications running in a different mode.

All of these demand the availability of fixed-width types.

ANSI introduced two sets of types. One is the signed fixed-size integral type:

- int8_t
- int16_t
- int32_t
- int64_t

The other is the unsigned fixed-size integral type:

- uint8_t

- uint16_t
- uint32_t
- uint64_t

These ANSI types are defined through the header <inttypes.h>. Note that the signed or unsigned are explicitly coded into the typedefs and not left to chance. Although it is unlikely that the defaults for short/int/long are unsigned, it is possible on some machines. Furthermore, by forcing the keyword, this would have the same error behavior in all cases if the user were to add a sign qualifier to the ANSI types, as in signed int8_t.

3.3.1.3 Structure Alignment and Bitfields

The LP64 specifications will change the size, member alignment, structure alignment, and bitfield sizes and alignment of most structures implicitly. Structures with only long and pointer types will at least double in size depending on the alignment mode.

Sharing data between 64-bit and 32-bit processes will not be possible unless the fixed-width types are used, or the structure is devoid of pointer and long types. Special attention needs to be paid to unions that attempt to overlay int types with long types or pointer types.

For the details of alignment in different modes and in combination with different compiler flags, consult the compiler reference manual.

Table 15 provides the different alignments found in 32- or 64-bit modes.

Table 15. Alignment of Basic Data Types in 32 and 64-Bit Mode

Type	**32-Bit**	**64-Bit**
char	1	1
short	2	2
int	4	4
long	4	8
long long	8	8
float	4	4
double	8	8
pointer	4	8

According to ANSI, a bit field will have a type that is a qualified or unqualified version of one of int, unsigned int, or signed int. Therefore, the ANSI mode cannot change the type.

Bitfields in ANSI mode can only be signed int or unsigned int. In extended mode, common mode, or k&r mode, non-integer bitfields are tolerated. When a non-integer bitfield is tolerated, it means that any type other than int will be converted to int.

The extended mode bitfields are updated to long types to admit 64-bit width in 64-bit mode. If a long type bitfield of length greater than 32-bits is used in 32-bit extended mode, the following message is given:

```
1506-003 (S) Width of a bit-field of type "long" cannot exceed 32.
```

If a long type bitfield of length greater than 64-bits is used in 64-bit extended mode, the following message is given:

```
1506-003 (S) Width of a bit-field of type "long" cannot exceed 64.
```

Bitfields are packed into the current word. Adjacent bitfields that cross a word boundary will start at a new storage unit. This storage unit is a word in power, full or natural alignment in 32-bit mode, but is a double word in 64-bit mode. In 64-bit mode, adjacent declarations of bitfields of type long can now be contained into one storage unit. Since long bitfields of greater than 32-bits were not permitted in 32-bit mode, this does not change and is not a portability problem.

Note that the packed alignment option just reduces the alignment ceiling to one, two, four, or eight bytes depending on the packed=1|2|4|8 setting and leaves the remaining alignment parameters unchanged.

3.3.1.4 Enum Support

Enum constants are always of type int, except when the range of these constants is beyond the range of int, in which case, they have type unsigned int. Enum variables may be smaller depending on the mode of -qenum=small|int|1|2|4|8 option.

small	Specifies that enumeration occupies a minimum amount of storage (either 1, 2, 4, or 8 bytes) depending on the range of enum constants.
int	Enumeration occupies 4 bytes and are represented by int.
1	Enumeration occupies 1 byte and are represented by char.
2	Enumeration occupies 2 bytes and are represented by short.

4	Enumeration occupies 4 bytes and are represented by int.
8	Enumeration occupies 8 bytes and are represented by long.

enum=int and enum=4 are not the same. The enum=4 allows signed and unsigned variant. The enum constants will usually be typed int, even in 64-bit mode, to enhance compatibility with 32-bit programs. Only when the range chooses an unsigned long or long, the constant will use unsigned long or long types respectively. In 32-bit mode, -qenum=8 will yield an warning message:

```
1506-749 (W) Enum=8 is not valid in 32-bit mode, setting enum=4 instead.
```

3.3.2 XL Fortran Version 5

XL Fortran Version 5.1 introduced a new compiler option, -q64. This allows the object code to run in 64-bit mode. The programming conventions are similar to C. For a better understanding of Fortran tuning on POWER3 processors, see *RS/6000 Scientific and Technical Computing: POWER3 Introduction and Tuning Guide*, SG24-5155.

3.3.3 System Libraries

AIX 4.3 provides a 32-bit Application Binary Interface (ABI) and a 64-bit ABI. The 32-bit ABI consists of the entire pre-AIX 4.3 ABI and provides binary compatibility at the same level as maintained by previous releases.

The dual ABI means two different version of all the Application Program Interfaces (APIs). The mechanism for this are two separate versions of all the objects in a given library. The objects are distinguished by distinct names. The linker is able to distinguish which object to use for a given symbol based on the differing object formats (32-bit and 64-bit).

The following libraries and APIs are not supported in the 64-bit environment:

- lib300.a - obsolete ASCII graphing library
- lib300s.a - obsolete ASCII graphing library
- lib4014.a - obsolete ASCII graphing library
- lib450.a - obsolete ASCII graphing library
- libIN.a - Interactive Systems library from RT days
- libPW.a - obsolete Programmer's Workbench library
- libcur.a - obsolete IBM-invented curses extensions
- libplot.a - obsolete ASCII graphing library
- libbsd.a - nonstandard BSD APIs (others are in libc.a)

The preceding APIs are also obsolete in 32-bit environments and will not be supported in the future. Also, the back level X11 compatibility and libcurses libraries do not have 64-bit versions.

The following functions will not be provided in the 64-bit version of libc.a:

- NC*
- NL*
- _NC*
- _NL*
- isj*
- jis*
- compile
- step
- advance

The asterisk represents a wild card.

3.3.4 Linker

Compilers, the assembler, and the binder create XCOFF64 object files when invoked in 64-bit mode. The AIX 4.3 linker links these object files in the same way that it links XCOFF32 object files.

The AIX linker supports the development of 64-bit applications, libraries, and kernel extensions. For 64-bit applications and libraries, the linker is able to read and write XCOFF64 files, performing internal processing appropriate for 64-bit mode. For 64-bit kernel extensions, the linker is able to mark exported symbols with storage-mapping class XMC_SV, XMC_SV64, or XMC_SV3264.

In this section, *mode* indicates the *linking mode*, which means whether an XCOFF32 or XCOFF64 file is generated as the output file. Mixed-mode linking is not allowed.

Archives may contain both XCOFF32 and XCOFF64 members. Depending on the mode, members in the appropriate format are processed, while other XCOFF members are silently ignored. Archives containing XCOFF64 members use a new archive file format that provides for separate global symbol tables for XCOFF32 and XCOFF64 members (see Section 3.3.5, “Archiver” on page 115). This new archive format is also used for all archives created on AIX 4.3, so the binder reads the new archive format even when running in 32-bit mode.

The following new command line flags and options in import files are introduced:

- **-b32** option

  Specifies 32-bit linking mode. In this mode, all input object files must be XCOFF32 files, or an error is reported. Only XCOFF32 archive members are processed. Other archive members are ignored. For import files specifying the mode of certain symbols, 64-bit imports are ignored.

- **-b64** option

  Specifies 64-bit linking mode. In this mode, all input object files must be XCOFF64 files, or an error is reported. Only XCOFF64 archive members are processed. Other archive members are ignored. For import files specifying the mode of certain symbols, 32-bit imports are ignored.

- **32** option in an import file

  This option can be used in an import file to specify that subsequent symbols should be processed when linking in 32-bit mode but ignored when linking in 64-bit mode. If no 32 or 64 option is specified, all symbols are processed in both 32 and 64-bit modes.

  Note: The syntax for import file options is a pound sign (#) followed by a blank followed by a list of options.

- **64** option in an import file

  This option can be used in an import file to specify that subsequent symbols should be processed when linking in 64-bit mode but ignored when linking in 32-bit mode. If no 32 or 64 option is specified, all symbols are processed in both 32 and 64-bit modes.

- **no32** or **no64** option in an import file

  This option overrides a previous 32 or 64 option. Subsequent symbols are processed in both 32 and 64-bit modes.

- **OBJECT_MODE** environment variable

  If the -b32 or -b64 options are not used, the OBJECT_MODE environment variable is examined to determine the linking mode. If the value of OBJECT_MODE is 64, 64-bit mode is used. If the value is 32_64, the linker prints an error message and exits with a non-zero return code. Otherwise, 32-bit mode is used.

If both -b32 and -b64 options are specified, the last specified option is used. If neither option is specified, the mode is determined from the value of the environment variable OBJECT_MODE.

New keywords are recognized in import and export files. The keywords are svc64 and svc3264, with synonyms syscall64 and syscall3264. For ease of use, svc32 and syscall32 are added as well. They are equivalent to svc and syscall. All these keywords may be in upper- or lower-case. The keywords are ignored in import files. In export files, a symbol exported with the svc64 keyword is given storage-mapping class XMC_SVC64 in the loader-section symbol table. Similarly, symbols exported with svc3264 are assigned a storage-mapping class, XMC_SVC3264. The existing flags and options -T, -D, -S, -bD, -bS, -bmaxdata, -bmaxstack, -bpD, and -bpT will accept 64-bit values as arguments. The 64-bit values are passed to the binder in the respective binder subcommands, regardless of the mode. The binder reports errors for used values that are too large for 32-bit mode. Depending on the options specified, some values are never used and do not result in an error.

3.3.5 Archiver

The AIX 4.3 the `ar` command handles the archiving of 64-bit XCOFF object modules in addition to the current 32-bit object modules. An archive file in AIX 4.2 supports only a single global symbol table to reference the symbols contained in all object-file modules within the archive. To support the two formats of object files, it is important that the symbols of 64-bit objects be distinguishable from those of 32-bit objects. This is not an issue for the old (pre-AIX 4.3) archive file format since 64-bit modules are not stored in these archives. For the AIX 4.3 archive format, however, there are two global symbol tables: one for 32-bit object symbols and one for 64-bit object symbols. The `ar` command is able to recognize each type of object file and store its symbols in the appropriate table.

The `ar` command maintains compatibility with the previous archive file format. If given an archive file of the old format, `ar` still adds, deletes, reorders, and lists members without altering the format of the archive file except in two cases: when the user explicitly requests conversion to the AIX 4.3 format by using the -o option, or when the user adds a 64-bit object to the archive. For the latter case, a 64-bit object cannot be handled by the old-format archive, so conversion is required. A mechanism is provided for `ar` to refuse the 64-bit object instead of converting the archive format.

When creating a new archive, the 4.3 `ar` command automatically uses the new format. For files that are not XCOFF objects of either type, `ar` processes them as usual. If such files are added to a nonexisting archive, the new format is used in creation. If ar is given an old-format archive, it is not reformatted (unless the user requests it). The new maximum size of an archive has increased from (10^{11} - 1) to (10^{19} - 1) bytes.

A new flag has been added, -X, which requires an argument of either 32, 64, or 32_64. This flag indicates to ar whether to accept only 32-bit objects or only 64-bit objects (in addition to any non-object files, which are always valid) or both. If both -X32 and -X64 are specified, ar treats it as though -X32_64 were specified and accept both object types. If only one of the options is specified, ar ignores all object files in the archive that are not of the specified type. If such objects are specified on the command line, an error message is issued, but other acceptable objects are still processed. If the -X option is given with an unrecognized argument, an error message is printed with the usage statement, and ar exits.

A new environment variable, OBJECT_MODE, is recognized by ar to determine the XCOFF file type(s) acceptable for processing. The values of OBJECT_MODE=32, OBEJECT_MODE=64, and OBJECT_MODE=32_64 all have the equivalent function of their -X flag counterparts. If both the environment variable and the -X flag are specified, the flag will take precedence over the environment variable. If no -X flag is given and OBJECT_MODE is unset or is set to an unrecognized value, 32-bit mode is used.

Displaying the symbol table with the -w option shows the symbol table depending on the chosen mode. In 32-bit mode, only the 32-bit symbol table is displayed; in 64-bit mode, only the 64-bit symbol table is displayed. In mixed mode, both are displayed. To distinguish the 32-bit table from the 64-bit table in mixed mode, each symbol table entry is followed by a field containing the characters 32 or 64, respectively. Each field is separated from the previous field by a single tab character. The 32-bit table is printed before the 64-bit table if both are present.

3.3.6 The dbx Debugger

The dbx command provides a symbolic debug program for C, C++, Pascal, and FORTRAN 32-bit and 64-bit programs. It is also able to process and examine core files generated from both 32-bit and 64-bit processes. dbx itself stays a 32-bit program, but its data is expanded to accommodate debugging of 64-bit programs. In case of debugging 32-bit code, there are *wrapper*, or interface routines, that translate from 32-bit formats to dbx's internal 64-bit data formats.

The dbx program:

- Automatically identifies the execution mode of the code
- Understands the new XCOFF64 format
- Understands the new archive format
- Understands the new coredump format
- Supports M:N threads debugging
- Accepts 64-bit addresses input
- Does arithmetical calculations in 64-bit precision
- Displays 64-bit values

The dbx parser has been changed to accept 64-bit addresses.

Note: To save typing of 16-digit long addresses, the user can set a dbx variable and use it as a base.

Commands will allow 64-bit values for addresses, subscripts, ranges, offsets, and so on. The 32-bit dbx required a suffix ll or ull on 64-bit number input. For example, 0x123456789ull. Any number that was too big for its type was set to the maximum value without any warning message. For ease of use, dbx now assumes long long for any input string greater than 8 digits for hex, 11 digits for octal, and 10 digits for decimal including leading zeroes if any. For convenience, underscores are allowed in any numeric input. For example, 0x1234_4567_890A_BCDE. They are ignored and not counted in the preceding sizes like in the assembler.

3.3.7 Commands and Utilities

All commands that needed to be modified for 64-bit support were modified to work with 32-bit and 64-bit objects (object files or processes). No commands and utilities were converted to 64-bit executables; they remain 32-bit executables.

There are two major reasons for changing commands:

- The new XCOFF64 format
- Data values that might exceed 2^{31}

In general, no command user interfaces or new flags were added to these commands, with the exception of the XCOFF-specific commands and the `lint` command. For most of the commands that deal with XCOFF files, a new flag was added, -X that requires an argument of either 32, 64, or 32_64. This flag indicates whether to recognize only 32-bit objects, 64-bit objects, or both. If both -X32 and -X64 are specified, the command treats it as -X32_64 and recognizes both object types. If only one of the options is specified, the command ignores all object files that are not of the specified type. If such objects are specified on the command line, an error message is issued, but other acceptable objects are still processed. If the -X option is given with an unrecognized argument, an error message is printed with the, usage statement and the command exits.

For the same XCOFF-specific commands, the environment variable OBJECT_MODE determines the XCOFF file type(s) to be recognized. The defined values of OBJECT_MODE=32, OBJECT_MODE=64, and OBJECT_MODE=32_64 all have the equivalent function of their -X flag counterparts. If both the environment variable and the -X flag are specified, the flag takes precedence over the environment variable. If no -X flag is given and OBJECT_MODE is unset, 32-bit mode is used. If OBJECT_MODE is set to an undefined value, an error message is printed, and the command fails unless the value is overridden on the command line.

Chapter 4. Application Development and Pthreads

This chapter details the changes in AIX 4.3 that may have an impact on the work of application developers. Applications may exhibit better performance by using new features and functions that are available in AIX 4.3.

4.1 C Language Standards

AIX Version 4 Release 3 made several changes and additions to conform to the ISO C Language Standard Normative Addendum One. The changes concern the handling of multibyte and wide character text formats.

The changes include new and altered programming interface specifications, many of which are contained in new #include files. For more information, see the National Language Support chapter of *AIX Version 4.3 General Programming Concepts: Writing and Debugging Programs*. The document is part of the online documentation, which is supplied with AIX. If you have not installed the documentation on a local machine, it can also be viewed on the internet using the URL:
`http://www.rs6000.ibm.com/doc_link/en_US/a_doc_lib/aixgen/`

4.2 IEEE POSIX and UNIX98 Conformance

AIX 4.3 is now aligned with the following standards:

- ISO/IEC 9945-1:1996 that incorporates ANSI/IEEE Std POSIX 1003.1-1990, 1003.1b-1993, 1003.1c-1995, and 1003.1i-1995 (1003.1b-1993 and 1003.1i-1995 are real-time extensions; 1003.1c-1995 is a threads extension).
- ISO C Amendment 1: 1995 (multibyte support).
- The Open Group UNIX98 specification that adds:
  - Extended threads functions over POSIX threads, based on industry input from Sun, Digital, HP and DCE.
  - Dynamic linking extensions to permit applications to share common code across many applications and ease maintenance of bug fixes and performance enhancements for applications.
  - N-bit cleanup (64-bit and beyond) to remove any architectural dependencies in the UNIX specification. This is of particular relevance with the IBM move to 64-bit UNIX.
  - Year 2000 alignment to minimize the impact of the millennium rollover.

4.2.1 Realtime Options

AIX 4.3 does not support the realtime optional parts of the IEEE POSIX and UNIX98 specifications. In particular, the routines provided in Table 16 are not supported in AIX Version 4.3.

Table 16. Unsupported Real-Time Routines

clock_getres()	clock_gettime()
clock_settime()	fdatasync()
lio_listio()	mlock()
mlockall()	mq_close()
mq_getattr()	mq_notify()
mq_open()	mq_receive()
mq_send()	mq_settattr()
mq_unlink()	munlock()
munlockall()	nanosleep()
sched_get_priority_max()	sched_get_priority_min()
sched_getparam()	sched_getsceduler()
sched_rr_get_interval()	sched_setparam()
sched_setscheduler()	sched_yield()
sem_close()	sem_destroy()
sem_getvalue()	sem_init()
sem_open()	sem_post()
sem_trywait()	sem_unlink()
sem_wait()	shm_open() function
shm_unlink()	sigqueue()
sigtimedwait()	sigwaitinfo()
timer_create()	timer_delete()
timer_getoverrun()	timer_gettime()
timer_settime()	

4.2.2 Unsupported Threads Options

AIX 4.3.2 does not support the optional pthread interfaces provided in Table 17.

Table 17. Unsupported Optional Threads Interfaces

pthread_attr_getinheritsched()	pthread_attr_setinheritsched()
pthread_mutex_getprioceiling()	pthread_mutex_setprioceiling()

4.2.3 Dynamic Linking Extension

The dynamic linking extension that came out of the Aspen group comprises a set of four routines and a header file to provide a portable API for manipulation of an implementation-defined class of files, such as shared libraries. These routines are based on those introduced in UNIX System V Release 4.

Use of dynamic linking allows several benefits for application developers:

- The ability to share commonly-used code across many applications, leading to disk and memory savings.
- It allows the implementation of services to be hidden from applications.
- It allows the re-implementation of services. For example, to permit bug and performance fixes or to allow multiple implementations selectable at runtime.

4.2.3.1 dlopen()

The dlopen() function is used to dynamically load a module into a process' address space. The value returned by dlopen() is a handle that can be passed to dlsym() to look up symbols in the loaded module. The handle can also be passed to dlclose() to allow the module to be removed from the address space.

Synopsis: `#include <dlfcn.h> void *dlopen(const char *pathname, int flags);`

If the `<pathname>` is /unix, dlopen() returns a handle that can be used to look up symbols in the current kernel image, including all kernel extensions. If `<pathname>` is NULL, a handle for the main executable is returned. Otherwise, `<pathname>` names a module that will be loaded.

If `<pathname>` contains a slash character(/), the pathname is used directly, whether it is an absolute or a relative path. Otherwise, a search for the named module is made. Directories to be searched are listed in:

1. Value of LIBPATH when the process was first loaded

2. Current value of LIBPATH that can be modified during execution with the `setenv` command

The new module and its dependents are actually loaded with the load() system call. If the main program was built with the -brtl option, the run-time linker processes the loaded modules. Next, initialization routines are called for modules loaded for the first time.

If dlopen() succeeds, it returns a handle that can be used for calls to dlsym() and dlclose(). Otherwise, dlopen() returns NULL and sets errno. If errno is set to ENOEXEC, additional information can be obtained by calling dlerror().

4.2.3.2 dlsym()

This function returns the address of a symbol in a module opened by dlopen().

Synopsis: `#include <dlfcn.h> void *dlsym(void *handle, const char *name);`

The argument `<handle>` must be a value returned by dlopen() that has not been passed to dlclose(). The argument `<name>` is the name of a symbol or the special value RTLD_EP. For functions, the symbol name should not begin with a period.

If the `<name>` is RTLD_EP, the address of the entry point of the module is returned. If there is no entry point, the address of the data section of the module is returned. The returned value may be passed to loadbind().

In general, the module denoted by `<handle>` and its original dependents are searched in breadth-first search order, based on the import file IDs listed in each module. If a module is linked with the -brtl option or the -G flag, the dependency list will contain all modules listed on the command line in the same order. Otherwise, all dependent modules will be listed in an unspecified order.

If dlsym() succeeds, it returns the address of the desired symbol. Otherwise, NULL is returned.

4.2.3.3 dlclose()

This function is used to unload a module loaded by dlopen(). The function is implemented by calling unload(). If this is the last use of the module, it is removed from the address space. Termination functions are called by unload() before the modules are actually unloaded.

The following is a synopsis of getdate():

```
#include <dlfcn.h> int dlclose(void *handle);
```

If dlclose() succeeds, 0 is returned. Otherwise, errno will be set to EINVAL, and EINVAL will be returned as well.

4.2.3.4 dlerror()

This function is used to return error information about the most recent call to the dlopen(), dlsym(), or dlclose() call. If dlopen() fails and sets errno to ENOEXEC, dlerror() will return a pointer to a buffer describing reasons for the failure. In all other failing cases, errno will have been set, and dlerror() will return the formatted string corresponding to errno.

Synopsis: `#include <dlfcn.h> char *dlerror(void);`

Error information is reset after a call to dlerror(). Therefore, if two consecutive calls are made to dlerror(), the second call will return a pointer to a null string.

Note: The dlerror() function is not thread-safe since the string may reside in a static area that is overwritten whenever an error occurs.

4.2.4 Year 2000

The following APIs and commands were changed in accordance with the UNIX98 specification:

4.2.4.1 getdate()

The following is a synopsis of getdate():

```
struct tm *getdate(const char *string);
```

The entry for getdate() states the following with respect to the format code %y:

```
    "%y     year within century (00-99)"
```

%y is now defined such that, when a century is not otherwise specified, values in the range 69-99 refer to the twentieth century, and values in the range 00-68 refer to the twenty-first century. The %C specifier has been added to the interface to denote the century and interprets the %y specifier in the absence of a century as noted in the section above.

4.2.4.2 strptime()

The following is a synopsis of strptime():

```
char *strptime(const char *buf, const char *format, struct tm *tm);
```

The entry for strptime() states the following with respect to the format code %y:

"%y is the year within century [00,99]; leading zeros are permitted but not required

%y is now defined such that, when a century is not otherwise specified, values in the range 69-99 refer to the twentieth century, and values in the range 00-68 refer to the twenty-first century.

4.2.4.3 date Command

Century handling has been added as follows:

```
# date mmddhhmm[[cc]yy]
```

cc is the century specifier.

4.2.4.4 prs Command

The `prs` command is part of SCCS and has been changed such that the -c option

```
-c cutoff
```

indicates the cut off date-time, in the form:

YY[MM[DD[HH[MM[SS]]]]]

The YY specifier is a two digit specifier to the year and, therefore, does not denote the century. YY is now defined such that values in the range 69-99 refer to the twentieth century, and values in the range 00-68 refer to the twenty-first century.

4.3 M:N Pthreads (4.3.1)

AIX 4.3.1 replaced the previous 1:1 threads implementation model with an M:N version. The M:N model complies with the UNIX98 pthreads standard, which includes the POSIX pthreads standard. Previous releases of AIX Version 4 complied with Draft 7 of the POSIX pthreads standard. AIX 4.3.1 is binary compatible with previous releases. The UNIX98 implementation is the default for application development, but you can use the cc_r7 or xlc_r7 compiler interfaces to develop new applications using Draft 7 pthreads. Users may need to alter existing source code to obtain the required function on AIX 4.3.1 using the default UNIX98 pthreads library.

4.3.1 Porting Application from Draft 7 Pthreads

There are very few differences between Draft 7 and the final standard.

- There are some minor errno differences. The most prevalent is the use of ESRCH to denote the specified pthread could not be found. Draft 7 frequently returns EINVAL for this failure.
- Pthreads are joinable by default. This is a significant change since it can result in a memory leak if ignored.
- Pthreads have process scheduling scope by default.
- The subroutine pthread_yield has been replaced by sched_yield.
- The various scheduling policies associated with the mutex locks are slightly different.

4.3.2 The M:N Model

In the M:N model, there are two underlying types of pthreads. Those with PTHREAD_SCOPE_SYSTEM, or system scope contention, otherwise known as global threads, and those with PTHREAD_SCOPE_PROCESS, or process scope contention. These threads are known as local threads. There are also two types of thread schedulers in the system. The AIX kernel scheduler schedules all kernel threads. There is also a user thread scheduler, which schedules the local pthreads in a process.

Global threads are mapped 1:1 to kernel threads, and hence, are scheduled exclusively by the AIX kernel scheduler. The 1:1 threads model used by prior releases of AIX Version 4 only uses global threads. Figure 7 shows the two threading models.

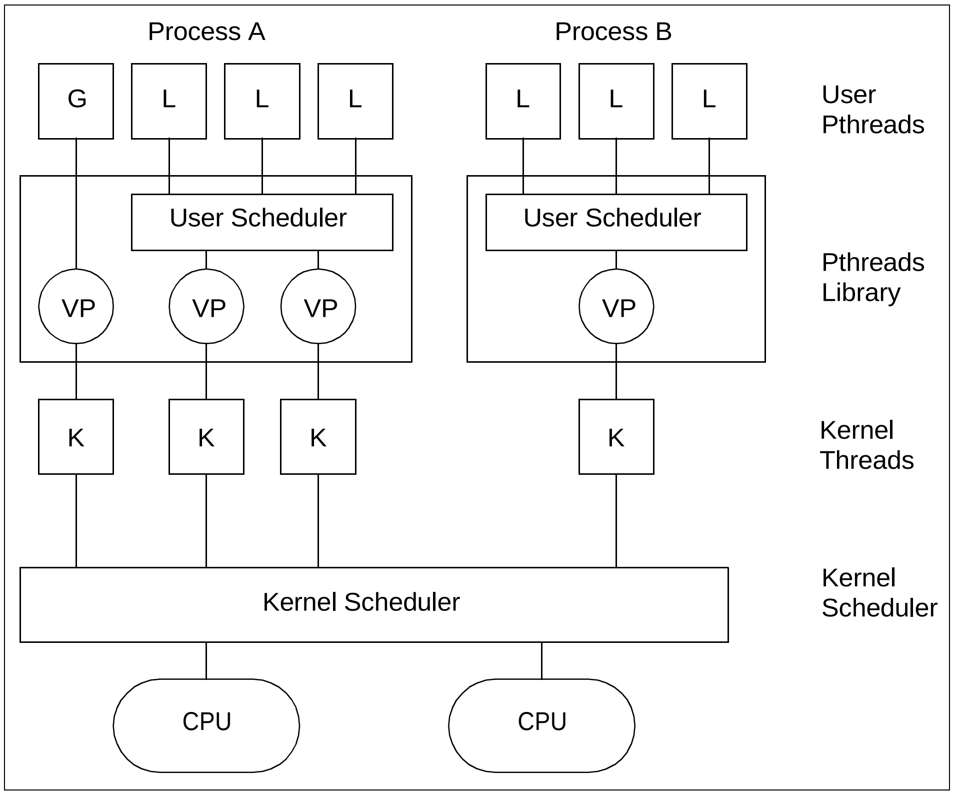


Figure 7. M:N Threads Model

The local pthreads are multiplexed over a set of kernel threads by the user scheduler, which is part of the pthreads library.

4.3.3 User Scheduler

The user scheduler is run on a dedicated *hidden* pthread, which is created when the pthreads library is initialized in M:N mode at process startup. There is one user scheduler for each process using the M:N model. The hidden pthread is created with system scope contention and therefore is scheduled directly by the kernel scheduler.

The user scheduler maintains a runqueue of runnable local pthreads and dispatches them on available kernel threads. Each kernel thread is represented in the pthreads library by a *virtual processor* structure (VP). There is a 1:1 mapping between VPs and kernel threads.

The user scheduler catches a SIGWAITING signal. Applications should not catch this signal, it is only for system use.

Each time a local pthread is created, terminates, or goes to sleep in the library, the user scheduler examines the ratio of kernel threads to active and sleeping user pthreads. If they are not consistent with the required values, then VPs, and hence, kernel threads, are created or destroyed as required. A VP that is to be deleted first places the user pthread it was running on to the queue of runnable pthreads maintained by the library. It then adds itself to the list of zombie VPs, and marks the underlying kernel thread for termination. The user scheduler traverses the list of zombie VPs on a regular basis and deletes the redundant VP structures.

Time slicing of local threads is initiated by the AIX scheduler, which sets a flag in the currently running kernel thread once it has obtained a full timeslice. On return from the clock tick interrupt handler, if the timeslice flag is set, the thread will call the user scheduler. The user scheduler places the current local thread on the local pthreads library runqueue and then selects the highest priority thread to run. If there are no threads on the run queue, then the current thread continues to run.

The user scheduler controls which pthreads are woken when a pthread event occurs. For example, when a mutex lock is released. The sleeping pthreads may have system-wide (global) or process-wide (local) contention scope. The user scheduler favors pthreads with system-wide scope over those with process-wide scope, regardless of their priorities. Priority is only used to decide between pthreads with the same contention scope. If they have the same priority, then the pthread that has been waiting the longest will be woken first.

When a local pthread makes a system call, it may block in the kernel waiting for a response from the system call. In this instance, the kernel thread and VP are not available to run another local pthread.

Consider a process with N+1 local threads, and N VPs, where one thread writes data to a pipe, and N threads read data from the pipe. The process would encounter a deadlock situation when the N threads reading from the pipe were blocked in the kernel. There would be no VP available for the N+1 thread to run on to write data to the pipe. This situation is avoided by a special check in the routine that a thread calls when about to block in the kernel. If the thread about to block is on the only VP of the process that is not already blocked, then the user scheduler is activated and instructed to create a new VP and kernel thread to run another local thread.

4.3.4 Mutex Locks

In previous versions of AIX, when a mutex lock is blocked, a pthread attempting to get the lock sleeps in the kernel. The internal structure of the mutex lock has been changed so that the list of threads waiting for the mutex is maintained in the user address space by the pthreads library. This is to allow the user scheduler to achieve relatively uniform levels of multiplexing over the remaining pthreads sharing the VPs. When the mutex lock is freed, the user scheduler examines the list of threads waiting for the mutex and activates one of them.

4.3.5 Tuning

The M:N pthreads implementation provides several environment variables that can be used to affect application performance. If possible, the application developer should provide a front-end shell script to invoke the binary executables in which the user may specify new values to override the system defaults. The following environment variables can be set by end users and are examined at process initialization time.

AIXTHREAD_SCOPE

This variable can be used to set the contention scope of pthreads created using the default pthread attribute object. It is represented by the following syntax:

```
AIXTHREAD_SCOPE=[P|S]
```

The value P indicates process scope, while a value of S indicates system scope. If no value is specified, then the default pthread attribute object will use process scope contention.

AIXTHREAD_MNRATIO

This variable allows the user to specify the ratio of pthreads to kernel threads. It is examined when creating a pthread to determine if a kernel thread should also be created to maintain the correct ratio. It is represented with the following syntax:

```
AIXTHREAD_MNRATIO=p:k
```

where `k` is the number of kernel threads to use to handle `p` pthreads. Any positive integer value may be specified for `p` and `k`. These values are used in a formula that employs integer arithmetic, which can result in the loss of some precision when big numbers are specified. If k is greater than p, then the ratio is treated as 1:1. If no value is specified, the default ratio depends on the default contention scope. If system scope contention is the default, the ratio is 1:1. If process scope contention is set as the default, the ratio is 8:1.

AIXTHREAD_SLPRATIO

This variable is used to determine the number of kernel threads used to support local pthreads sleeping in the library code on a pthread event. For example, attempting to obtain a mutex. It is represented by the following syntax:

`AIXTHREAD_SLPRATIO=k:p`

where `k` is the number of kernel threads to reserve for every `p` sleeping pthreads. Notice that the relative positions of the numbers indicating kernel threads and user pthreads are reversed when compared with AIXTHREAD_MNRATIO. Any positive integer value may be specified for `p` and `k`. These values are used in a formula that employs integer arithmetic, which can result in the loss of some precision when large numbers are specified. If k is greater than p, then the ratio is treated as 1:1. If the variable is not set, then a ratio of 1:12 is used.

The reason for maintaining kernel threads for sleeping pthreads is that, when the pthread event occurs, the pthread will immediately require a kernel thread to run on. It is more efficient to use a kernel thread that is already available than it is to create a new kernel thread once the event has taken place.

AIXTHREAD_MINKTHREADS

This variable is a manual override to the AIXTHREAD_MNRATIO. It allows you to stipulate the minimum number of active kernel threads. The library scheduler will not reclaim kernel threads below this number.

SPINLOOPTIME

This variable controls the number of times the system will try to get a busy lock without taking a secondary action, such as calling the kernel to yield the processor. This control is really intended for MP systems where it is hoped that the lock is held by another actively running pthread and will soon be released. On uniprocessor systems, this value is ignored.

YIELDLOOPTIME

This variable controls the number of times that the system yields the processor when trying to acquire a busy mutex or spin lock before actually going to sleep on the lock. This variable has been shown to be effective in complex applications where multiple locks are in use.

4.3.6 Combined Thread-Safe Libraries

Non-thread-safe and thread-safe libraries have been combined into one set of libraries, thereby turning thread-safety on by default.

AIX Version 4.2	libc.a (non-thread-safe) and libc_r.a (thread-safe).
AIX Version 4.3.1	libc.a which is thread-safe.

Libraries, such as X11R6, which link with libc.a are not thread-safe by default; they are thread-aware.

New libraries that are thread-safe include:

- libbsd.a
- libm.a
- libmsaa.a
- librts.a
- libodm.a
- libs.a
- libdes.a
- libXm.a
- libXt.a
- libX11.a

These thread-safe libraries enable a convenient programming model for exploiting SMPs and simplify exploitation of threads by applications, middleware, and other API providers.

4.4 Pthreads Suspend and Resume (4.3.2)

The pthreads library has been enhanced to provide the ability to suspend and resume individual threads. This function is added to AIX 4.3.2 to assist in the porting of applications from other platforms.

The pthreads implementation on AIX 4.3.2 complies with the UNIX98 standard. The four new API functions are not part of this standard, and this is indicated by appending _np to their names to indicate that they are NON-POSIX compliant.

The four new user functions are:

- `int pthread_suspend_np(pthread_t thread);`
- `int pthread_continue_np(pthread_t thread);`
- `int pthread_attr_setsuspendstate_np(pthread_attr_t *attr, int suspendstate);`

- `int pthread_attr_getsuspendstate_np(pthread_attr_t *attr, int *suspendstate);`

The pthread_suspend_np and pthread_continue_np functions are used to immediately suspend and resume execution of the thread indicated by the function argument.

The pthread_attr_getsuspendstate_np and pthread_attr_setsuspendstate_np functions are used to get and set the value of the new suspendstate member of the pthread_attr_t structure. The suspendstate argument can be set to either PTHREAD_CREATE_SUSPENDED_NP or PTHREAD_CREATE_UNSUSPENDED_NP. The default value of the suspendstate attribute of a pthread_attr_t structure is PTHREAD_CREATE_UNSUSPENDED_NP.

The new functions work in both the 1:1 and M:N threading environments.

4.5 Pthread Debug Library (4.3.3)

Pthread debug library (libpthdebug.a) provides a set of APIs that allows debuggers to provide pthread specific debugging. Currently only the dbx debugger gains access to all the private thread information through the use of internal data structures.

The pthread library (libpthreads.a) creates and maintains pthread, mutex, attribute, condition variable and read/write lock objects information in debugee's address space. The pthread debug library provides information about these objects. It also provides function which allows the debuggers to hold and unhold pthreads, and access and set the context of a pthread.

The debuggers that uses pthread debug library should be compiled with 32-bit mode, since neither ptrace() nor ptracex() functions are supported in 64-bit mode.

The APIs provided with pthread debug library fall into following categories:

- Session Functions

  The pthread debug library assigns a unique session handle to the process the debugger is debugging. This user handle is passed back to the debugger whenever the pthread debug library invokes a call back function.

- Execution Control Functions

  These functions are provided to control the execution of the debuggee's threads. They can be used continue one or more pthreads by informing the pthread library which threads to hold and which threads to unhold.

- Context Functions

  These functions are provided to set or get the context information of a pthread from either the kernel or the pthread data structure om the debuggee's address space.

- Object Functions

The pthread debug library maintains lists for pthreads, attributes, mutexs, mutex attributes, condition variables, condition variable attributes, read/write locks, read/write lock attributes, pthread specific keys, and active keys. Each of them is represented by a type specific handle. The pthread debug library provides functions to get the handle of each object in the list and to get detailed information about the object.

4.6 Preserve Modified Ptrace Data (4.3.2)

AIX 4.3.2 has improved the performance of the `ptrace()` subroutine, which is used by debuggers to control the execution of applications under their control. Debuggers use a private copy of the text pages for the application being traced and any shared libraries it uses. This allows the debugger to modify the text pages to insert breakpoints without affecting any other processes on the system that may be running the same executable or shared library text.

Prior to AIX 4.3.2, when the application being debugged calls the load(), or loadbind() routines to load a private module into its address space, the system loader reloads fresh copies of all the text pages for the application and any required shared libraries. In so doing, any modifications made to the text pages are lost, so the debugger has to reinsert breakpoints after the application calls load() or loadbind().

The function of the ptrace routine has been modified along with the system loader to maintain ptrace altered copies of text pages across calls to load or loadbind. This will improve the performance of the debugger when controlling large applications that call load or loadbind many times since breakpoints and other changes will not have to be reinserted.

4.7 Direct I/O

Direct I/O is a way of opening JFS files that allows for disk reads and writes using less CPU cycles than normal I/O. The main CPU savings come from avoiding the memory-to-memory copy done by the JFS. As the difference between memory and CPU cycle times increases, the savings achieved by avoiding this copy becomes greater.

Direct I/O offers a big performance gain for applications that do large block I/O (32 KB or greater) to JFS files and a smaller increase in performance for small block I/O. It does not improve raw I/O performance.

With normal I/O, the I/O request is first sent to the device driver. To service the request the device driver uses Direct Memory Access (DMA) to copy the data to or from pages in a file persistent segment. The data is then copied between the persistent segment and userspace through calls to vmcopyin() or vmcopyout(). Thus, a file's persistent segment acts as a file cache.

With direct I/O, the data is not cached, but rather, I/O is done directly to a user supplied buffer through cross-memory technology. In other words, DMA is done directly from the disk to user space and conversely through the device strategy routine of the JFS file system.

Optimization was also made to the DMA setup routines. This improves large block I/O to JFS files and to raw logical volumes. However, the benefits of the DMA changes are much less than the benefits of direct I/O.

Take Note

It is important to note that since direct I/O reads are done synchronously and there is no read-ahead benefit, if they are not used correctly, they can also take much longer. The only read-ahead-like semantics that direct I/O can benefit from will be read-ahead performed by the disk device driver (normally 32 KB). For this reason, it is very important for a direct I/O reader to specify large read requests to match the performance of normal cached I/O readers. To match the performance of normal cached I/O readers, a direct I/O reader should issue read requests of at least 128 KB.

Direct I/O is considered *advisory*. This means that if a file is opened for direct I/O, and another application decides to open that same file for normal I/O, the file will be opened using normal cached I/O until direct I/O can be resumed (the normal I/O application closes the file).

Files can also be opened for a deferred update using the O_DEFER flag. If a file is opened with O_DEFER, and a subsequent open for direct I/O is issued, all writes will use normal cached I/O. Similarly, if another application opens the file with O_DEFER while it is already opened for direct I/O, all I/O to the file will be cached.

4.7.1 Opening Files for Direct I/O

A new flag, O_DIRECT, has been defined in fcntl.h. When an application opens a file specifying or calling this flag through the fcntl() system call, I/O to this file will be done using direct I/O.

4.7.2 Inode Flags

When a file is using direct I/O, the i_flag field in the inode is set with the IDIRECT flag, defined in inode.h. Even so, it is not enough to simply have a flag in the inode to determine if the file is using direct I/O or not. If a normal I/O application opens the file while a direct I/O application currently has it open, then all I/O will be done using normal I/O until the normal I/O reader or writer closes the file. A count of direct I/O readers is maintained to determine if the direct I/O can be resumed. A new field in the inode, i_diocnt, has been added for this purpose. This field indicates if any application has the file opened for direct I/O.

4.7.3 JFS Function Calls for Direct I/O

There are only a few functions that were affected in the JFS for direct I/O to be implemented. These functions are serialized by the inode lock and are described below.

jfs_map()	If a file opened for direct I/O is mapped, the IDIRECT flag is reset, and all subsequent I/O will be done using normal I/O. If the mapped file is then closed, direct I/O will be resumed.
jfs_close()	Close semantics are closed with direct I/O. When the final close occurs, (checked by the counts on gnode) the IDIRECT flag in the inode is turned off. If a close is initiated by a normal I/O reader or writer and another application opens the file for direct I/O, all cached pages are flushed to disk, and direct I/O is resumed on the file.
jfs_dio()	jfs_dio() is called from jfs_rdwr(). If the FDIRECT flag is set, jfs_dio() evaluates if direct I/O can be done for a particular file. This function performs all alignment and file state consistency checking.

If a read or write request cannot be done using direct I/O, jfs_dio() returns a non-zero return code, and the request is done through normal cached I/O.

4.7.4 System Archive Utilities

Archive commands are typical applications that can benefit from the use of direct I/O. Therefore, the standard system commands `tar`, `backup`, `restore`, and `cpio` have been enabled to use direct I/O. Since these commands are *read-once* commands, that is, they do not reference the data again after it has been read and written to media, the copyin() and copyout() characteristics of normal cached I/O consume a lot of unnecessary CPU when these commands are executing. The enabling of these commands has been accomplished by changing all calls to open() and setting the O_DIRECT flag.

4.8 Shared Memory Enhancements

The following enhancements reflect changes to the shared memory function in AIX.

4.8.1 Larger Shared Memory Regions (4.3.1)

The maximum size of a shared memory region created by the shmget() routine and attached to a process' address space by the shmat() routine has been increased from 256 MB to 2 GB. Prior to AIX 4.3.1, it was possible to mmap() a memory mapped file of up to 2 GB, but an anonymous memory region was limited to 256 MB. This meant that a large memory region had to be created in several 256 MB portions and each portion attached individually. AIX 4.3.1 has removed this restriction, so it is now possible to attach a 2 GB memory region with one call to the shmat() routine.

If EXTSHM=ON is set and an application performs a shmget() with a size greater than SEGSIZE-PAGESIZE, the system will use the traditional shmat() and not mmap() as would be the case when EXTSHM=ON.

4.8.2 128 KB Shared Memory IDs (4.3.2)

AIX 4.3.2 now supports 128 KB mem, sem, and shm IDs, up from 4 KB in the previous releases.

4.8.3 Shared Memory Debugging Enhancements (4.3.2)

AIX 4.3.2 has added the facility for additional information to be included in the core file that may be produced by an application program when it encounters certain types of error.

There are two methods of enabling the extra information to be included in the core dump.

- The application can use the sigaction routine to set the SA_FULLDUMP flag for the signal that will cause the core file to be generated.
- Enable full core information as the default, either from the SMIT Change / Show Characteristics of Operating System panel, the Web-Based System Manager Devices panel by selecting sys0, or by using the command:

  ```
  chdev -l sys0 -a fullcore='true'
  ```

When an application faults, and a full core is generated, the core will include all the shared memory regions of the faulting process that are currently attached.

The dbx debugger has been changed to understand the extra information in the core file and allow the developer to interrogate the user defined variables contained in the shared memory regions of the process at the time of termination.

4.9 DMA Pre-Translation (4.3.2)

DMA pre-translation of memory buffers reduces the cost of setting up DMA operations. Its objective is to reduce DMA setup path length for selected and predetermined I/O operations to improve performance. The enhances the performance of network memory buffers (mbufs), file system I/O, raw I/O, and page I/O.

In previous AIX versions, during a DMA operation, the majority of path length was spent in page translation/lookup paths to get the virtual to physical address translations for DMA.

The term *pre-translation* refers to the concept of performing the virtual to physical address translations for a data buffer to be involved in a DMA operation once, for the life of the data buffer, instead of for each individual I/O setup for the buffer. In general, a subsystem desiring to benefit from the performance gain of pre-translation calls a new kernel service passing in a buffer address, length, and cross-memory descriptor. The kernel service will attach to the cross memory descriptor pre-translation information for the buffer. Then, whenever the buffer is used for I/O, the DMA services recognize the presence of pre-translation info in the cross-memory descriptor and avoid page table lookups.

There is no change required by device drivers to take advantage of this enhancement, as long as a network driver is using mbufs from the global net_malloc pool and performing dynamic on-the-fly DMA mappings (compared with copying data to pre-mapped buffers).

4.10 Fast fork() Function (4.3.1)

AIX Version 4.3.1 introduces a fast-fork function called f_fork() that is based on IEEE POSIX specifications. The f_fork() call is precisely like fork() except:

- It is required that the child process calls one of the exec functions immediately after it is created. Since fork handlers are never called, the application data, mutexes, and the locks are all undefined in the child process.

The use of f_fork() will significantly enhances the performance of Internet and Web server applications that need to create many short lived child processes.

4.11 New Sockets System Call (4.3.2)

AIX 4.3.2 has added the new send_file() system call. Its use is aimed at applications that transmit file data across the network using a socket connection. It offers a speed improvement over the traditional method of sending a file across the network by avoiding unnecessary data copying where possible.

```
#include <sys/socket.h>

ssize_t send_file(Socket_p, sf_iobuf, flags)

int *Socket_p;
struct sf_parms *sf_iobuf;
uint_t flags;
```

Using send_file eliminates the need to read a file just to send it across the network. Applications can remove the read() call, and therefore avoid redundant transfer of data between kernel space and user space. The send_file call reads file data into the new Network Buffer Cache (NBC). The NBC is allocated from an area of the mbuf pool and uses mbufs for file data storage. The networking subsystem then transmits the data directly from the mbufs in the NBC across the specified socket. The system call dynamically caches the data in the NBC, thus improving performance for files that are sent frequently across the network, and which do not change often. This feature can be disabled on a file by file basis.

The application sending the data will need to be altered to use the send_file call. The greatest improvement in performance can be gained by using direct I/O to read the file that is to be transmitted. This can be achieved simply by opening the file using the O_DIRECT flag. This flag enables send_file to bypass the JFS cache when reading the file, thus further reducing the number of data transfers required.

The size of the NBC, and various cache tuning parameters can be altered using the `no` command. The options that can be changed are:

nbc_limit	Maximum size of the NBC. Specifies in KB the maximum amount of memory that can be used for the NBC. The default value is derived from the size of the mbuf pool (thewall), which in turn, is determined from the amount of physical memory. If a system has less than 512 MB of memory, the default value of nbc_limit is 0.
nbc_max_cache	Maximum size of a cache object in the NBC. Specified in bytes, default value is 131072 (128 KB) bytes.
nbc_min_cache	Minimum size of a cache object in the NBC. Specified in bytes, default value is 1.
send_file_duration	Specifies the cache validation duration for all the file objects that system call send_file accessed in the Network Buffer Cache. This attribute is in number of seconds, the default is 300 for 5 minutes. 0 means that the cache will be validated for every access.

Cache statistics can be viewed using the command `netstat -c`, that provides an output similar to the following:

```
# netstat -c
Network Buffer Cache Statistics:
-------------------------------
Current total cache buffer size: 325312
Maximum total cache buffer size: 325312
Current total cache data size: 207450
Maximum total cache data size: 207450
Current number of cache: 25
Maximum number of cache: 25
Number of cache with data: 25
Number of searches in cache: 2063
Number of cache hit: 1044
Number of cache miss: 81
Number of cache newly added: 25
Number of cache updated: 0
Number of cache removed: 0
Number of successful cache accesses: 1069
Number of unsuccessful cache accesses: 56
Number of cache validation: 0
```

The send_file call only supports the TCP/IP protocol. In other words, the sockets of type SOCK_STREAM in address family AF_INET. Both blocking and non-blocking connections are supported. In blocking mode, send_file blocks until all of the file data has been transmitted. In non-blocking mode, or in the event send_file is interrupted, the system call updates parameters in the sf_parms structure to indicate the amount of data that has actually been transmitted.

4.12 Network Buffer Cache Improvements (4.3.3)

The Network Buffer Cache (NBC) was introduced in AIX 4.3.2 to improve the performance of the network file servers, such as the Web, FTP, and SMB servers. In AIX 4.3.3 new features have been added allowing NBC to have a bigger capacity and multiple keys for the cache access mechanism.

A new, secondary, key access to data in the NBC has been added in order to make the FRCA HTTP GET engine (see “HTTP GET Kernel Extension (4.3.3)” on page 421) search among cache data objects using the URL of the object. The primary key access is the vnode address of the file cached and it is used by the send_file() system call.

The base NBC configuration uses network buffers with an upper size limit that is defined by thewall variable and cannot grow over 1 GB. Taking into account that a normal network I/O activity may take about 250 MB, the effective maximum NBC cache is about 750 MB. This size is too small for most servers that access large file sets.

The new AIX 4.3.3 design of NBC allows the use of additional 256 MB memory segments for caching additional data. Since over one million memory segments are available, NBC is able to handle a huge cache on top of the wall. The additional segments are private and are mapped and pinned in memory only when referenced and unmapped when network I/O is done.

In order to achieve optimal performance with private segments, the network device drivers must be able to support the private segments. Otherwise, the data from the private segments must be copied from the private segments to normal mbufs, before transmission. The network devices that currently support private segments are Gigabit Ethernet, 100 Mbps Ethernet, and 155 Mbps ATM PCI.

In order to control these segments, new `no` parameters have been added:

nbc_pseg	Maximum number of private segments that can be created by the NBC. The default value is 0. In each private segment only one file can be stored, so you should use an appropriate value to store many files.
nbc_pseg_limit	Maximum size of pinned memory that can be created for private segments. The value is in kilobytes and the default is half of the physical memory size.

When one of the two nbc_pseg or nbc_pseg_limit value is reached, cache data may be flushed in order to let new data in. If one of the two value is set to 0, all existing NBC private segments will be flushed and released and no more segments are created.

Previous `no` options (nbc_limit, nbc_max_cache, nbc_min_cache) are not changed. The maximum memory used in global segment is given by nbc_limit. Objects with size smaller than nbc_min_cache are not cached. Objects bigger than nbc_max_cache were previously left out of the NBC, now they are put in private segments if they are physically smaller than 256 MB.

New statistics are added to keep track of private segment usage in NBC. They are displayed by the `netstat -c` command:

- Current total cache data size in private segments: byte count of the total object size currently cached in private segments.
- Maximum total cache data size in private segments: highest total object size in bytes that has ever been cached in private segments.
- Current total number of private segments: number of private segments currently used in the cache.
- Maximum total number of private segments: highest number of private segments that has ever been used in cache.
- Current number of free private segments: current number of private segments.
- Current total NBC_NAMED_FILE entries: number of cache entries that are indexed by multiple keys.

- Maximum total NBC_NAMED_FILE entries: the highest number of secondary cache entries that has ever been created in cache.

4.13 Binder Library Enhancements (4.3.2)

The binder library, libld.a, that provides functions to allow an application to create and manipulate XCOFF object files, has been enhanced in AIX Version 4.3.2 to support a cross mode environment.

This allows 32-bit applications to create and manipulate both 32-bit and 64-bit objects using a consistent interface. The changes also allow 64-bit objects to create and manipulate both 32-bit and 64-bit objects. The functions in the library transparently open both 32-bit and 64-bit object files, as well as both small format and large format archive files.

An application need not know the format of an object file before opening it. It can call the ldopen function and then check the magic number of the file or archive member.

4.14 Fast Single Instruction Breakpoint (4.3.3)

In some cases, it may be useful for certain trap instructions to be handled by the process being debugged, instead of causing the process to be stopped and the debugger to be notified. This capability can be used to patch running programs or programs for which source code is not available. In order for a process to use this capability, fast traps must be enabled, which requires a ptrace() call from a debugger on behalf of the process.

The fast trap instruction is an unconditional trap immediate instruction of the form:

```
twi 14,r13,0xNXXX
```

To allow a process to handle fast traps, a debugger uses the subroutine call:

```
ptrace(PT_SET,pid,0,PTFLAG_FAST_TRAP,0)
```

This capability can be canceled with another call:

```
ptrace(PT_CLEAR,pid,0,PTFLAG_FAST_TRAP,0)
```

If a process is enabled to handle fast traps when the debugger detaches, the capability is automatically cleared.

A typical usage of the fast trap instruction could be:

1. A debugger or a special purpose tool replaces some specific bl instructions with the fast trap instruction.
2. The debugger or the tool enables the fast trap capability using ptrace().
3. The debugger installs the SIGTRAP signal handler so that the signal handler can modify the behavior of the debugged program without source code.
4. The replaced routine runs without being interfered by the debugger.

4.15 Java Developer’s Kit (4.3.3)

Java Developer’s Kit (JDK) 1.1.8 is shipped with AIX 4.3.3. JDK 1.1.8 is the latest release of Java environment in the Java 1 platform. New features included in JDK 1.1.8 are:

- Java Remote Method Invocation - Internet Inter-ORB Protocol

  A new version of RMI can run over IIOP and Java programs that use the RMI can interoperate with Common Object Request Broker (CORBA) objects which are programmed in other languages.

- Java Security Migration Aid

  This feature provides the more robust policy-based security model of the Java 2 platform in the Java 1 platform environment. The security migration aid is intended to help users migrate from the relatively simple Java 1 platform security mode to the finer grained Java 2 platform model. The Security Migration Aid supports the use of security policies, permissions, tools, and Java run time security managers as defined in the Java 2 platform security model.

- Java Database Connectivity - Object Database Concavities

  This feature enhances the capability of enterprise customers to communicate with databases using Java. The bridge provides JDBC access to databases with ODBC drivers.

- Swing

  Part of the Java Foundation Classes (JFC) that implements a new set of GUI components with a pluggable look and feel. Swing is implemented in 100% Pure Java and is based on the JDK 1.1 light-weight UI framework. The pluggable look and feel lets you design a single set of GUI components that can automatically have the look and feel of any OS platform (Windows, Solaris, Macintosh). Swing components include both 100% Pure Java versions of the existing AWT component set (Button, Scrollbar, Label, to name a few), plus a rich set of higher-level components (such as tree view and list box).

- Big Decimal

  IBM has enhanced Java's big decimal math class by adding support for floating point arithmetic. Computer systems must provide an arithmetic that gives results that people expect. This is not available in Java today; a decimal floating point arithmetic is needed -- one that gives the same results as the arithmetic that people learn at school. IBM's dig decimal class implements the decimal arithmetic defined in the ANSI standard X3.274-1996.

JDK 1.1.8 is included in the AIX BOS operating system CD-ROM in AIX 4.3 and a bonus pack CD-ROM in AIX 4.2.

4.16 Ship Perl on AIX (4.3.3)

The Perl script language is shipped with AIX 4.3.3. The version shipped is 5.5.3 which is most stable version at this time of writing. Perl is packaged in the perl.rte fileset and included in an AIX BOS CD-ROM.

The following simple Perl script send an HTTP GET request to localhost and prints the HTML document returned:

```
#!/usr/bin/perl
use LWP::Simple;
$doc=get 'http://localhost';
print $doc
```

Note that to use the WWW features of Perl, you need to download libwww-perl module from http://www.linpro.no/lwp/.

4.17 KDB Kernel Debugger and the kdb Command (4.3.3)

The KDB Kernel Debugger provides a symbolic debugger for the AIX kernel, kernel extensions, and device drivers. `kdb` is also a command to allow examination of system crash dumps. `kdb` is an alternative to the current kernel debug `crash` command, and the KDB Kernel Debugger is an alternative to the LLDB kernel debugger.

Note that the KDB Kernel Debugger and the `kdb` command are two separate entities. The KDB Kernel Debugger is a debugger you can use to debug the kernel, device drivers, and other kernel extensions. You can use the `kdb` command to view data contained in system image dumps. However, the `kdb` command may be run on an active system to view system data. Also note that although they are separate and have different roles, they do share many commands, since both work with the AIX kernel.

Usage Notes

To use the KDB Kernel Debugger, you need to use the `bosboot -k` command, as described in “AIX Documentation Updates for the KDB Kernel Debugger” on page 145, so that you can use a KDB kernel. A *KDB kernel* is a version of the AIX kernel that has been designed to work with the new `kdb` command and KDB Kernel Debugger tools.

An example of how to enable the KDB kernel and then use the KDB debugger is in “Enabling the KDB Kernel Debugger” on page 147. You do not need to do this if you only want to use the `kdb` command, either on an active system or to view a system dump.

To determine what kernel you are currently using, use the method described in “Enabling the KDB Kernel Debugger” on page 147.

4.17.1 Fileset Changes Associated with the Introduction of KDB

AIX 4.3.3 now includes the `kdb` command and KDB kernels. The filesets that have been changed include:

- bos.up ships unix_kdb which is the UP version of the KDB kernel.
- bos.mp ships unix_mp_kdb which is the MP version of the KDB kernel.
- bos.sysmgt.serv_aid ships `kdb`, `kdb_up`, and `kdb_mp` which are the kdb dump readers.
- bos.msg.en_US.rte includes the kdb catalog for the kdb dump readers.

You can look at the /unix files installed on your system by executing:

```
# ls -l /unix
lrwxrwxrwx   1 root     system        21 Aug 20 14:17 /unix ->
/usr/lib/boot/unix_mp
# ls -l /usr/lib/boot/un*
lrwxrwxrwx   1 root     system        21 Aug 20 14:17 /usr/lib/boot/unix ->
/usr/lib/boot/unix_mp
-r-xr-xr-x   1 root     system   3673916 Aug 19 07:05 /usr/lib/boot/unix_mp
-r-xr-xr-x   1 root     system   5523759 Aug 19 08:36
/usr/lib/boot/unix_mp_kdb
```

Note that it is the `bosboot` command, together with the setup of these /unix files, that builds the boot logical volume that contains the kernel that you use.

4.17.2 AIX Documentation for KDB

The `kdb` command and KDB Kernel Debugger are documented in *AIX Version 4.3 Kernel Extensions and Device Support Programming Concepts,* SC23-4125, in section *KDB Kernel Debugger and Command,* as well as in the `kdb` man page. The following are links to articles for the following three topics:

- KDB Kernel Debugger and `kdb` Command

  This describes the KDB Kernel Debugger and `kdb` command. It describes what these tools are and how you can use them.

- Subcommands for the KDB Kernel Debugger and `kdb` Command

  This includes a description of each of the subcommands, with each subcommand identified as being available through the KDB Kernel Debugger, the `kdb` command, or both. Along with a description of each subcommand there is at least one example for at least one of the subcommand's options.

- Using the KDB Kernel Debug Program

  This provides some examples of how to use KDB to perform common debugging tasks. This section includes a simple kernel extension and program to load it; this is used in examples of using the KDB Kernel Debugger that you can step through to learn about this tool.

4.17.3 AIX Documentation Updates for the KDB Kernel Debugger

In the *Kernel Extensions and Device Support Programming Concepts* manual, SC23-4125, the section *Loading and Starting the KDB Kernel Debugger* should be replaced with the following section.

4.17.3.1 Loading and Starting the KDB Kernel Debugger

The KDB Kernel Debugger must be loaded at boot time. This requires that a boot image is created with the debugger enabled. To enable the KDB Kernel Debugger, the `bosboot` command must be invoked with a KDB kernel specified and options set to enable the KDB Kernel Debugger. KDB kernels are shipped as /usr/lib/boot/unix_kdb for UP systems and /usr/lib/boot/unix_mp_kdb for MP systems; as opposed to the normal AIX kernels of /usr/lib/boot/unix_up and /usr/lib/boot/unix_mp. The specific kernel to be used in creation of the boot image may be specified using the -k option of `bosboot`. The KDB Kernel Debugger must also be enabled using either the -I or -D options of `bosboot`.

Examples of `bosboot` commands that build boot images using the KDB kernel for a MP system are:

- `bosboot -a -d /dev/ipldevice -k /usr/lib/boot/unix_mp_kdb`

  The KDB Kernel debugger is disabled.

- `bosboot -a -d /dev/ipldevice -D -k /usr/lib/boot/unix_mp_kdb`

  The KDB Kernel Debugger is enabled but is not invoked during system initialization.

- `bosboot -a -d /dev/ipldevice -I -k /usr/lib/boot/unix_mp_kdb`

  The KDB Kernel Debugger is enabled and is invoked during system initialization.

The file /usr/lib/boot/unix_kdb would be used, instead of /usr/lib/boot/unix_mp_kdb, for a UP system.

KDB Environment Notes

The execution of the `bosboot` command only builds the boot image; this boot image is not used until the machine is restarted.

External interrupts are disabled while the KDB Kernel Debugger is active.

If the KDB debugger is invoked during system initialization the `g` subcommand must be issued to continue the initialization process.

The KDB Kernel Debugger requires exclusive access of a machine.

The links /usr/lib/boot/unix and /unix are not changed by `bosboot`. However, these links are used by user commands such as `sar, crash`, and others to read symbol information for the kernel. Therefore, if these commands are to be used with a KDB boot image these links should point to the kernel specified for the `bosboot` command that created the KDB boot image. This may be done by removing and recreating the links. This must be done as root. For the previous `bosboot` examples the following command sequence would set up the links correctly:

1. `rm /unix`
2. `ln -s /usr/lib/boot/unix_mp_kdb /unix`
3. `rm /usr/lib/boot/unix`

```
4. ln -s /usr/lib/boot/unix_mp_kdb /usr/lib/boot/unix
```

Similarly, if you chose to quit using a KDB kernel then the links for /usr/lib/boot/unix and /usr/lib/boot/unix_mp_kdb should be modified to point to the kernel specified to `bosboot`.

Default Kernel for bosboot

/unix is the default kernel used by `bosboot`. Therefore, if this link is changed to point to a KDB kernel, subsequent `bosboot` commands which do not have a kernel specified will use the KDB kernel unless this link is changed. The link can be changed with a method similar to the previous command sequence that used the `rm` and `ln` commands.

4.17.4 Enabling the KDB Kernel Debugger

An AIX documentation topic (See “KDB Kernel Debugger and kdb Command” on page 145.) describes how to enter the KDB environment. It also shows you that you check if the KDB environment is currently active on your system by executing:

```
# kdb
             (0)> dw kdb_avail
```

If the above `dw` subcommand returns a 0, the KDB Kernel Debugger is not available. This section guides you through the use of the KDB kernel and the associated KDB Kernel Debugger. Read the entire section before you try the examples.

Documentation Update

To check for the availability of the KDB Kernel Debugger, use only the `kdb` subcommand `dw kdb_avail`, not the subcommand `dw kdb_wanted`.

4.17.4.1 Checking the Kernel

You can combine the `kdb` and `crash` commands to do:

```
# echo 'dw kdb_avail'|kdb|egrep "expected|kdb"
(0)> dw kdb_avail
(0)> expected symbol or address
```

The error message `expected symbol or address` means that you are not using the KDB kernel, so you cannot use the KDB Kernel Debugger. To use this tool (the tool is not invoked during the boot process in this case), enter:

```
# bosboot -a -d /dev/ipldevice -D -k /usr/lib/boot/unix_kdb

bosboot: Boot image is 8491 512 byte blocks.
# shutdown -Fr
```

When your system has completed its boot process, you can again use the `kdb` and `echo` commands to verify that you should be able to use the KDB Kernel Debugger:

```
# echo 'dw kdb_avail'|kdb|egrep "expected|kdb"
(0)> dw kdb_avail
kdb_avail+000000: 00000001 00000001 00000001 02000000  ................
```

This command has returned a value of `000000: 00000001` so you can now use the KDB Kernel Debugger.

Debug Terminal Choice

Even though you can now use the KDB Kernel Debugger, you will not usually enter the debugger environment with the `kdb` command. (That is, a breakpoint subcommand would need to be set.) To use the KDB kernel Debugger you need to use an ASCII terminal connected to a native serial port. Press the Ctrl-\ keys simultaneously on a tty keyboard to enter the KDB Kernel Debugger.

Graphical console devices are not supported, but native keyboards are.

4.17.4.2 Using the KDB Kernel Debugger

To use an ASCII terminal for the debugger on host A, you can:

1. Connect the native serial port to another AIX system's serial port using a null modem cable. Assume that this second system is host B.
2. Connect to host B from whatever client you are using. Some clients are:
   - A telnet session.
   - An aixterm, dtterm or xterm on a network X11 server.
3. Install the uucp fileset (bos.net.uucp) on host B:
   1. Use `script` to log your work.
   2. From your client session on host B, use the `cu` command to establish a tty session on host A.

   These steps are similar to:

   ```
   root@itsosmp:/tmp > script kdb-dbg.out
   ```

```
Script started, file is kdb-dbg.out
root@itsosmp:/tmp > cu -ml tty0
```

4. Login, then execute the required key sequence:
   - If you are using a tty session from any client, press the Ctrl-\ (control and back-slash) keys simultaneously to obtain output similar to the following:

     ```
     #
     # Debugger entered via keyboard.
     05001E28     beq-    cr0.eq,<050021C0>
     KDB(0)>
     KDB(0)> stat
     POWER_RS1 machine with 1 cpu(s)
     .......... SYSTEM STATUS
     sysname... AIX        nodename.. itsosrv2
     release... 3          version... 4
     machine... 000000131C nid....... 0000131C
     Debugger entered via keyboard.
     age of system: 9 hr., 23 min., 7 sec.
     .......... SYSTEM MESSAGES

     AIX Version 4.3
     sysconfig SYS_SINGLELOAD failed
     <- end_of_buffer
     KDB(0)> q
     ```

     Note that the debugger advises that it was entered from the keyboard.
   - If the previous attempt fails, or if host A is physically located in a convenient location, you can also use the native keyboard sequence. This still requires a client session through the tty connection, since the graphics adapters are not supported. Press the Ctrl-Alt-Numpad4 keys simultaneously to obtain:

     ```
     # Debugger entered via keyboard with key in SERVICE position using numpad 4
     .waitproc+000038     bne-    cr0.eq,<.waitproc+000038>
     KDB(0)> q
     ```

   Note that the debugger message is different for this sequence, but in both examples, the q subcommand quits the KDB Kernel Debugger and returns the system to the normal processing environment.

Complete your debugger work as normal.

Limit Use of KDB

You should disable the debugger when you are finished using it. Otherwise, if you were to accidentally press the debugger key sequence you would interrupt all system users.

For example, all communications stops, as shown by this increase in ping time from another system while the debugger is active:

```
64 bytes from 9.3.187.212: icmp_seq=11 ttl=255 time=1 ms

64 bytes from 9.3.187.212: icmp_seq=12 ttl=255 time=6785 ms
```

4.17.4.3 Disabling the KDB Kernel Debugger and Kernel

You can disable the KDB Kernel Debugger, but not the kernel, by entering:

```
# bosboot -a -d /dev/ipldevice -k /usr/lib/boot/unix_kdb

bosboot: Boot image is 8483 512 byte blocks.
# shutdown -Fr
```

When the system completes the boot process, verify that the KDB Kernel Debugger is disabled by entering:

```
# echo 'dw kdb_avail'|kdb|egrep "expected|kdb"
(0)> dw kdb_avail
kdb_avail+000000: 00000000 00000002 00000000 02000000  ................
```

Since the link commands were not done along with the bosboot command that enabled the KDB Kernel Debugger, some commands may be affected. For example, to use crash on the live system in this state, you need to explicitly specify the /unix file by executing:

```
# crash /dev/mem /usr/lib/boot/unix_kdb
> stat
        sysname: AIX
        nodename: itsosrv2
        release: 3
        version: 4
        machine: 000000131C00
        time of crash: Fri Aug 27 13:34:46 CDT 1999
        age of system: 10 min.
        xmalloc debug: disabled
```

Likewise, the kdb command that you can use to confirm the KDB kernel is in use needs to be changed. The incorrect version is:

```
# echo 'dw kdb_avail'|kdb|egrep "expected|kdb"
[kdb_read_mem] no real storage @ 000DFA8C
```

If you specify the KDB kernel, then it works as expected.

```
# echo 'dw kdb_avail'|kdb /dev/mem /usr/lib/boot/unix_kdb|egrep
"expected|kdb"
(0)> dw kdb_avail
kdb_avail+000000: 00000000 00000001 00000002 02000000  ................
```

To return to the original, default kernel environment, execute:

```
# bosboot -a -d /dev/ipldevice -k /usr/lib/boot/unix_up

bosboot: Boot image is 6339 512 byte blocks.
# shutdown -Fr
```

No other steps are required for this example since the /unix and /usr/lib/boot/unix links were not changed.

4.17.5 A Comparison of the New and Existing Kernel Tools

This section enables you to become familiar with the new kernel tools by comparing the subcommands of the new tools with the subcommands from the existing tools that do a similar function.

4.17.5.1 crash and kdb Subcommands Cross Reference

Table 18 on page 151 cross references the crash and kdb commands. In some cases it is possible that several kdb subcommands may be required to perform the same function as a single command in crash. In such cases, all of the appropriate kdb subcommands are listed.

Also note that even though the subcommands perform approximately the same function, there are often differences between the syntax for the crash and kdb subcommands. Refer to the AIX documentation on the crash and kdb subcommands for more information. Table 18 on page 151 is ordered alphabetically by crash subcommand.

Table 18. A Comparison of crash and kdb Subcommands

Dump Tool Subcommands		**Function**
crash	kdb	
buf	buf	Display system buffer headers

Dump Tool Subcommands		Function
crash	kdb	
buffer	Use both buf and one of d, dw, dd, dp, dpw, dpd commands to display memory	Display data in a system buffer
calc	hcal, dcal	Calculator. Note, KDB does not support operator precedence. This will be changed in the future, but currently input to hcal and dcal are processed from left to right and parentheses are not supported.
callout	trb	Display entries on the active trb list
cm	sw	Change segment registers used in address resolution.
conv	dcal, hcal	hex/decimal conversion
cpu	cpu	Switch CPUs
dblock	None	Display a streams data block header
decode	dc, dcp	Decode an instruction word. Note, dc and dcp actually disassemble the code at a specified address; they do no disassemble a user specified instruction word.
devsw	dev	Show device switch table entries
dlock	dla	Search for possible deadlocks
dmodsw	Dmodsw,dmodsw	Display the streams driver switch table
ds	ts	Find symbol closes to a given address
du	Use both ttid and one of d, dw, dd, dp, dpw, dpd commands to display memory	Hex and ASCII dump of a thread's uthread structure and of the user structure of the process for the thread
dump	None	Display component dump table and allow selection for formatting usint the dump formatting routines
errpt	None	Display error log entries
file	file	Display file table entries

Dump Tool Subcommands		**Function**
crash	kdb	
find	find	Search for a pattern
fmodsw	Fmodsw	Display streams module switch table
fs	f	Trace a kernel stack
help	help	Display help
hide	set no_symbol	Hide symbols for symbolic name translation. Note, KDB does not currently have a function to turn off symbolic name translation for specific symbols; symbolic name translation is either on or off. The set no_symbol subcommand toggles the setting.
id	dc, dcp	Instruction decode (disassembly)
inode	ino	Display i-node table
kfp	None	Set frame pointer for use by trace subcommand
knlist	None	Display addresses for specified symbols
le	lke	Display load list entries
link	None	Traverse a linked list
linkblk	None	Display streams linkblk table
lock	slk,clk	Print lock information
mblock	msg	Display streams message block headers
mbuf	mbuf	Display mbuf structures
mst	mst	Display mstsave portion of the uthread structure
ndb	tcb, udb, sock, mbuf, ifnet	Display network kernel data structures
netm	None	Display net_malloc_police records
netstat	None	Display network statistics
nm	nm	Display symbol value and type
od	d, dw, dd, dp, dpw, dpd	Display memory
ppd	ppda	Display per-processor data area

Dump Tool Subcommands		**Function**
crash	kdb	
prall	None	Print an assortment of structures to stdout without further user interaction.
print	None	dbx style formatted display of data for a structure
proc	proc	Display process table entries
qrun	sqh,sqe	Display list of scheduled streams queues
queue	streams, stream, queues, queue	Display the streams queue
quit	q	Exit
search	None	Display the symbol at an indicated address
segst64	u -64	Display segstate information for a 64-bit process
set	set	Set user options
socket	sock	Display socket structures
stack	Use both stack and one of d, dw, dd, dp, dpw, dpd commands to display memory	Display the memory for a kernel stack
stat	stat	Display statistics found in a dump
status	sw, thread, proc	Display a description of the kernel thread scheduled on a processor
stream	streams, stream	Display the stream head table
symptom	None	Display a symptom string for a dump
tcb	mst	Display the mstsave portion of the user structure for a thread
thread	th	Display thread table entries
trace	strack	Display a kernel stack
ts	ts	Find the text symbol closes to an address
tty	None	Display tty structures

Dump Tool Subcommands		Function
crash	kdb	
unhide	set no_symbol	Restore symbols for symbolic name translation. Note, KDB does not currently have a function to turn off symbolic name translation for specific symbols; symbolic name translation is either on or off. The set_nosymbol subcommand toggles the setting.
user	user	Display uthread structure and associated user structure
var	var	Display tunable system parameters
vfs	vfs	Display entries in the VFS table
vnode	vnode	Display v-node data
which	None	Display the name of the kernel source file containing a symbol
xmalloc	xmalloc	Display information concerning the allocation and usage of kernel memory
!	!	Run shell commands
?	?	Display command summary

4.17.5.2 LLDB and KDB Subcommands Cross Reference

Table 19 on page 155 cross references the LLDB and KDB subcommands. In some cases it is possible that several KDB subcommands may be required to perform the same function as a single command in LLDB. In such cases, all of the appropriate KDB subcommands are listed. Also note that even though the subcommands perform approximately the same function, there are often differences between the syntax for the LLDB and KDB subcommands. Refer to the AIX documentation on the LLDB and KDB subcommands for more information. Table 19 on page 155 is ordered alphabetically by LLDB subcommand:

Table 19. A Comparison of lldb and kdbSubcommands

System Debugger Subcommands		Function
lldb	kdb	
alter	m, mw, md, mp, mpq, mpd	Alter memory
back	mr iar	Decrement the Instruction Address Register (IAR).

System Debugger Subcommands		Function
lldb	kdb	
break	b, lb	Set a breakpoint.
breaks	b, lb, wr, ww, wrw, lwr, lww, lwrw	Lists currently set breakpoints.
buckets	unknown	Displays statistics for net_malloc memory pool
clear	c, lc, ca, cw, lcw	Clears (removes) breakpoints
cpu	cpu	Sets the current processor or shows processor states
display	d, dw, dd, dp, dpw, dpd	Displays a specified amount of memory.
dmodsw	dmodsw	Displays the STREAMS driver switch table.
drivers	dev	Displays the contents of the device driver table.
find	find, findp	Finds a pattern in memory.
float	dr fp	Displays the floating point registers.
fmodsw	Fmodsw, fmodsw	Displays the STREAMS module switch table.
fs	ino, vno, vfs	Displays the internal file system tables.
go	g	Starts the program running.
? or help	h	Displays the list of valid commands.
loop	bt and [	Run until control returns to this point.
mst64	mst	Displays mstsave64 of a 64-bit process.
map	exp	Displays the system loadlist.
mblk	kmstats	Displays the contents of message block structures.
netdata	mbuf, udb, soc, tcpcb	Displays the mbuf, ndd, socket, inpcb, and tcpcb data structures.
next	mr iar	Increment the Instruction Address
origin	Not Applicable	Sets the origin.

System Debugger Subcommands		Function
lldb	kdb	
ppd	ppda	Displays per-processor data.
proc	proc	Displays the formatted process table.
queue	queue, queues	Displays contents of STREAMS queue at specified address.
quit	ca, g, q	Ends a debugging session.
reason	stat	Displays the reason for entering the debugger.
reboot	eboot	Reboots the machine.
reset	NONE	Releases a user-defined variable.
screen	NONE	Displays a screen containing registers and memory.
segst64	unknown	Displays the states of all memory segments of a 64-bit process.
set	mr	Defines or initialize a variable.
sregs	dr	Displays segment registers.
sr64	NONE	Displays segment registers only in 64-bit context.
st	mw	Stores a full word in memory.
stack	stack	Displays a formatted kernel stack trace.
stc	m	Stores one byte in memory.
step	n,s,S,B	Performs an instruction single-step.
sth	sth	Stores a halfword in memory.
stream	stream, streams	Displays stream head table.
swap	NONE	Switches from the current display and keyboard to another RS232 port.
sysinfo	stat	Displays the system configuration information.
thread	thread, ttid, tpid	Displays thread table entries.

System Debugger Subcommands		Function
lldb	kdb	
trace	trace	Displays formatted trace information.
trb	trb	Displays the timer request blocks.
tty	NONE	Displays the tty structure.
un	dc	Displays the assembly instruction(s).
user	user	Displays a formatted user area.
user64	user	Displays the user structure of a 64-bit process.
uthread	mst,user	Displays the uthread structure.
vars	NONE	Displays a listing of the user-defined variables.
vmm	vmker, pfhdata, vmstat, pdt, scb, pft, pte, apt, vmwait, vmadder, ames, rmap, vmlog, zproc	Displays the virtual memory data structure.
watch	wr, ww, wrw, lwr, lww, lwrw	Watches for load or store at an address.
xlate	tr, tv	Translates a virtual address to a real address.

4.18 Malloc Enhancements (4.3.3)

The AIX memory subsystem has been significantly enhanced in this release.

4.18.1 Replaceable Malloc Subsystem

You can now use alternative memory allocation routines through the use of environment variables. This allows you to fully replace the system malloc routine, including the one used within libc itself without any relinking. Previously, a malloc replacement would only be used for application calls directly linked with a new malloc but would not be utilized by shared libc routines. This may have resulted in undetectable memory references. This enhancement helps make memory references more visible, thus enabling third party memory access products performing functions including:

- Garbage collection

- Memory usage analysis
- High speed memory allocation

4.18.1.1 Implementation

The memory subsystem includes the following interfaces:

- malloc()
- free()
- realloc()
- calloc()
- mallopt()
- mallinfo()

The existing memory subsystem works for both threaded and non-threaded applications. Your user defined memory subsystem should be thread-safe since there are no checks to verify that it is. If a non-thread safe memory module is loaded in a threaded application, memory and data may be corrupted.

Your user defined memory subsystem 32- and 64-bit objects must be placed in an archive with the 32-bit shared object named `mem32.o` and the 64-bit shared object named mem64.o.

You can enable your memory subsystem either by using the MALLOCTYPE environment variable or using a global variable _malloc_user_defined_name bound statically in the user application. Note that the check for a user supplied memory subsystem will only be done once per process and once set cannot be changed.

Replacement memory subsystems written in C++ are not supported due to the use of the libc.a memory subsystem by the C++ library libC.a.

The complete implementation details are in the AIX documentation section “User Defined Malloc Replacement” in *General Programming Concepts* SC23-4128.

4.18.2 Malloc Multiheap

This enhancement is very important if you are concerned about the performance of threaded applications running on multiprocessor systems, such as Netscape’s Web Server. A single free memory pool, or heap, is provided by default by malloc. With AIX version 4.3.3, the capability to enable the use of multiple heaps of free memory is provided, which reduces thread contention for access to memory. You can use this feature by using the MALLOCMULTIHEAP environment variable, as described in the readme file /usr/lpp/bos/README. For example, set

```
MALLOCMULTIHEAP=true
```

Setting MALLOCMULTIHEAP in this manner will enable malloc multiheap in its default configuration, with all 32 heaps and the fast heap selection algorithm.

4.18.3 Debug Malloc

Debug Malloc is a powerful, effective tool for developing more reliable software, since software will be much less likely to have memory management related problems. However, this facility may initially appear to cause problems since it will result in the manifestation of software defects that have previously been dormant or difficult to discover. Now a core file will be generated so that software problems can be analyzed and resolved early in your software’s life cycle.

This debug facility provides memory overlay detection capabilities for the user level malloc() and free() routines which are similar to those provided by the xmalloc debug facility, which already is a standard kernel debug tool shipped with AIX. The kernel tool is referred to as the Memory Overlay Detection System (MODS). Debug Malloc can be turned on without modifying executables, simply by exporting an environment variable. Activation and configuration of the Debug Malloc capability is available at runtime using the MALLOCTYPE and MALLOCDEBUG environment variables.

Debug Malloc detects a variety of incorrect scenarios, including:

- Writing to memory that is owned by another program or routine
- Writing past the end (or before the beginning) of declared variables or arrays
- Writing or reading past the end (or before the beginning) of dynamically allocated memory
- Writing to or reading from freed memory
- Trying to free the same memory twice
- Exiting without freeing allocated memory

It works by allocating at least a full page (4096 bytes) for every memory allocation, and at free time, hiding the page so that any references will cause a page fault. It causes larger quantities of memory to be used, but it is very effective in finding incorrect memory uses.

A thorough discussion of this facility is in the /usr/lpp/bos/README file.

Debug Malloc Limitations

Debug Malloc is not appropriate for full-time, constant, or system-wide use. Although it is designed for minimal performance impact upon the application being debugged, it may have significant negative impact upon overall system throughput if it is used widely throughout a system.

The README file describes this and other limitations.

4.18.3.1 Using Debug Malloc

This section includes four sample programs and their output that illustrate the use of some different Debug Malloc options. The comments included at the start of each program's text describes what the program shows and how you can use it with Debug Malloc enabled.

- The doublefree.c program tries to free the same memory twice

```
/*
 * doublefree.c: This program attempts to free the same memory twice.
 *
 * To build and run this program with Debug Malloc enabled, perform the
 * following steps:
 *
 *   cc -o doublefree doublefree.c
 *   export MALLOCTYPE=debug
 *   export MALLOCDEBUG=validate_ptrs
 *   doublefree
 *
 * Upon completion of this program, turn off Debug Malloc and examine the
 * core file as follows:
 *
 *   export MALLOCTYPE=
 *   export MALLOCDEBUG=
 *   dbx doublefree
*/

#include <stdio.h>
#include <stdlib.h>

main()
{
char *ptr = NULL;

   ptr = (char *) malloc(sizeof(char) * 4096);
   free(ptr);
   free(ptr);
   exit(0);
}
```

You can use this program in the following sequence:

1. Compile the program, and then execute it to verify that it does not create a core file. That is, it appears to work correctly.

```
# ls -l
total 8
-rw-r--r--   1 root     sys          624 Aug 23 16:08 doublefree.c
# make doublefree
        cc -O  doublefree.c -o doublefree
# ./doublefree
# ls -l
total 24
-rwxr-xr-x   1 root     sys         4278 Aug 23 16:09 doublefree
-rw-r--r--   1 root     sys          624 Aug 23 16:08 doublefree.c
```

2. Set up the Debug Malloc environment, and then execute the program again.

```
# export MALLOCTYPE=debug
# export MALLOCDEBUG=validate_ptrs
# ./doublefree
Debug Malloc: Buffer (0x20002000) has already been free’d.
IOT/Abort trap(coredump)
# ls -l
total 40
-rw-r--r--   1 root     sys         8115 Aug 23 16:12 core
-rwxr-xr-x   1 root     sys         4278 Aug 23 16:09 doublefree
-rw-r--r--   1 root     sys          624 Aug 23 16:08 doublefree.c
```

3. Turn off the Debug Malloc environment with the commands:

```
# export MALLOCTYPE=
# export MALLOCDEBUG=
```

4. You now have an error message and a core file to analyze. You can use the `dbx` command to analyze the core file.

```
# dbx doublefree
Type ’help’ for help.
reading symbolic information ...warning: no source compiled with -g

[using memory image in core]

IOT/Abort trap in raise at 0xd016fd28
0xd016fd28 (raise+0x4c) 80410014        lwz   r2,0x14(r1)
(dbx) where
raise(??) at 0xd016fd28
abort() at 0xd0169450
do_validate_part2(??, ??, ??) at 0xd0163424
do_debug_free(??) at 0xd01630c4
```

```
main() at 0x10000340
(dbx)
```

The where dbx subcommand has provided you with some more information so that you can return to your source code to look for where it is incorrectly freeing memory. In this case, there should only be one, not two, calls to the free function.

- The memleak.c program exits without freeing allocated memory.

```
/*
 * memleak.c: This program exits without freeing allocated memory.
 *
 * To build and run this program with Debug Malloc enabled, perform the
 * following steps:
 *
 * cc -o memleak memleak.c
 * export MALLOCTYPE=debug
 * export MALLOCDEBUG=report_allocations
 * memleak
 *
 * Upon completion of this program, turn off Debug Malloc as follows:
 *
 * export MALLOCTYPE=
 * export MALLOCDEBUG=
*/

#include <stdio.h>
#include <stdlib.h>

main()
{
char *ptr = NULL;

   ptr = (char *) malloc(sizeof(char) * 4096);
   ptr = (char *) malloc(sizeof(char) * 4096);
   ptr = (char *) malloc(sizeof(char) * 4096);
   exit(0);
}
```

You can use this program in the following sequence:

1. Compile the program, and then execute it to verify that it does not display any error message. That is, it appears to work correctly.

```
# ls -l
total 8
-rw-r--r--   1 root     sys          622 Aug 23 17:19 memleak.c
# make memleak
        cc -O  memleak.c -o memleak
# ./memleak
# ls -l
total 16
-rwxr-xr-x   1 root     sys         4016 Aug 23 17:19 memleak
-rw-r--r--   1 root     sys          622 Aug 23 17:19 memleak.c
```

2. Set up the Debug Malloc environment, and then execute the program again.

```
# export MALLOCTYPE=debug
# export MALLOCDEBUG=report_allocations
# ./memleak
Current allocation report:
    Allocation #1: 0x2000A000
        Allocation traceback:
        0x2000B024  __start
        0x2000B028  main
        0x2000B02C  malloc

    Allocation #2: 0x20007000
        Allocation traceback:
        0x20008024  __start
        0x20008028  main
        0x2000802C  malloc

    Allocation #3: 0x20004000
        Allocation traceback:
        0x20005024  __start
        0x20005028  main
        0x2000502C  malloc

    Allocation #4: 0x20001FF8
        Allocation traceback:
        0x2000201C  __start
        0x20002020  main
        0x20002024  malloc
        0x20002028  atexit
        0x2000202C  malloc

Total allocations: 4.

# ls -l
total 16
-rwxr-xr-x   1 root     sys         4016 Aug 23 17:19 memleak
-rw-r--r--   1 root     sys          622 Aug 23 17:19 memleak.c
```

You do not get a core file created, but memleak's output should help you debug the memleak.c source program. The program memleak.c has three calls to malloc, but none of these are freed before the program exits. The Debug Malloc output has four allocation records because, as explained in the *report_allocations* section of the /usr/bos/lpp/README file, one allocation record will always be listed for the atexit() handler that dumps the allocation records.

3. Turn off the Debug Malloc environment with the commands:

```
# export MALLOCTYPE=
# export MALLOCDEBUG=
```

Note for Long Output

If this debug method results in more than one screen of output, you may be presented with misleading, confusing malloc output. This may happen if you redirect memleak's output, for example, to the pg command. This can be avoided by saving your program's standard output in a file, and then turning off Debug Malloc before you check the output. In this example, you should execute:

```
# ./memleak > memleak.out2
# export MALLOCTYPE=
# export MALLOCDEBUG=
# pg memleak.out2
```

- The overwrite.c program attempts to write beyond allocated memory

```
/*
 * overwrite.c: This program attempts to write beyond allocated memory.
 *
 * To build and run this program with Debug Malloc enabled, perform the
 * following steps:
 *
 *   cc -o overwrite overwrite.c
 *   export MALLOCTYPE=debug
 *   export MALLOCDEBUG=align:0
 *   overwrite
 *
 * Upon completion of this program, turn off Debug Malloc and examine the
 * core file as follows:
 *
 *   export MALLOCTYPE=
 *   export MALLOCDEBUG=
 *   dbx overwrite
 *
 * Then, perform the following steps:
 *
 *   export MALLOCTYPE=debug
 *   export MALLOCDEBUG=align:2
 *   overwrite
 *
 * Turn off Debug Malloc and examine the new core file.
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void copy7();
void copy8();
void copy9();
```

```
main()
{
   copy7(); /* OK with both align:0 and align:2 */
   copy8(); /* seg faults with align:0, OK with align:2 */
   copy9(); /* seg faults with align:0 or align:2 */
   exit(0);
}

void copy7()
{
char *ptr = NULL;

   ptr = (char *) malloc(sizeof(char) * 7);
   strcpy(ptr, "123456");  /* six bytes + null = seven bytes */
   free(ptr);
   return;
}

void copy8()
{
char *ptr = NULL;

   ptr = (char *) malloc(sizeof(char) * 7);
   strcpy(ptr, "1234567");  /* seven bytes + null = eight bytes */
   free(ptr);
   return;
}

void copy9()
{
char *ptr = NULL;

   ptr = (char *) malloc(sizeof(char) * 7);
   strcpy(ptr, "12345678");  /* eight bytes + null = nine bytes */
   free(ptr);
   return;
}
```

You can use this program in the following sequence:

1. Compile the program, and then execute it to verify that it does not create a core file. That is, it appears to work correctly.

```
# ls -l
total 8
-rw-r-----   1 root     sys         1443 Aug 24 00:12 overwrite.c
# make overwrite
        cc -O  overwrite.c -o overwrite
# ./overwrite
# ls -l
total 24
-rwxr-xr-x   1 root     sys         5380 Aug 24 00:13 overwrite
-rw-r-----   1 root     sys         1443 Aug 24 00:12 overwrite.c
#
```

2. Set up the Debug Malloc environment, with the align option is set to 0, and then execute the program again.

```
# export MALLOCTYPE=debug
# export MALLOCDEBUG=align:0
# ./overwrite
Segmentation fault(coredump)
# ls -l
total 40
-rw-r--r--   1 root     sys         8115 Aug 24 00:16 core
-rwxr-xr-x   1 root     sys         5380 Aug 24 00:13 overwrite
-rw-r-----   1 root     sys         1443 Aug 24 00:12 overwrite.c
#
```

3. Turn off the Debug Malloc environment with the commands:

```
# export MALLOCTYPE=
# export MALLOCDEBUG=
```

4. You can use the dbx command to analyze the core file.

```
# dbx overwrite
Type 'help' for help.
reading symbolic information ...warning: no source compiled with -g

[using memory image in core]

Segmentation fault in strcpy.strcpy [overwrite] at 0x100005c4
0x100005c4 (strcpy+0xe4) 9ce50001       stbu   r7,0x1(r5)
(dbx) where
strcpy.strcpy() at 0x100005c4
copy8() at 0x100003e8
main() at 0x10000328
(dbx) q
#
```

5. In the previous step, overwrite failed in the copy8() function. Set up the Debug Malloc environment again, but this time with align set to 2, and then execute the program again.

```
# export MALLOCTYPE=debug
# export MALLOCDEBUG=align:2
# ./overwrite
Segmentation fault(coredump)
# ls -l
total 40
-rw-r--r--   1 root     sys         8115 Aug 24 00:36 core
-rwxr-xr-x   1 root     sys         5380 Aug 24 00:13 overwrite
-rw-r-----   1 root     sys         1443 Aug 24 00:12 overwrite.c
#
```

6. Turn off the Debug Malloc environment with the commands:

```
# export MALLOCTYPE=
# export MALLOCDEBUG=
```

7. You can use the dbx command to analyze the core file.

```
# dbx overwrite
Type ’help’ for help.
reading symbolic information ...warning: no source compiled with -g

[using memory image in core]

Segmentation fault in strcpy.strcpy [overwrite] at 0x100005c4
0x100005c4 (strcpy+0xe4) 9ce50001       stbu   r7,0x1(r5)
(dbx) where
strcpy.strcpy() at 0x100005c4
copy9() at 0x10000388
main() at 0x1000032c
(dbx) q
#
```

This second debug attempt using a different align option has not failed in neither the copy7() or copy8() functions, but only the copy9() function. To understand why, you need to review the bos README file that contains a detailed discussion about the use of Debug Malloc. Specifically, it says:

```
Additional Information about align:n Option
-------------------------------------------
The following formula can be used to calculate how many bytes of overreads
and/or overwrites Debug Malloc will allow for a given allocation request
when MALLOCDEBUG=align:n and size is the number of bytes to be allocated:

                 ((((size / n) + 1) * n) - size) % n
```

The README then explains what happens with the two values of align used in this example, that is the value two and the special case of zero. The overwrite program includes three functions (copy7(), copy8() and copy9()) that each allocate seven bytes of storage using malloc.

- The copy7() function copies exactly seven bytes into the memory provided by malloc, so there are no writes beyond allocated memory. This means that overwrite does not fail in this function.
- The copy8() function copies eight bytes into memory, so it overwrites the allocated area by one byte.

  For align:0, no bytes of overwrites are allowed so the core file dbx debug output shows that the overwrite program failed in copy8().

  For align:2, the formula works out to be one since the value of size is 8 and the value of n, the align option, is 2. As the bos README explains, the overwrite program allocated an odd number of bytes. Consequently, Debug Malloc allocates an extra byte so it will accept a one byte overhead. This is why `overwrite` does not fail in copy8(), but continues to copy9() where it does fail.

- The copy9() function copies nine bytes into memory, so it overwrites the allocated area by two bytes. This amount is not acceptable for either align scenario. The overwrite program will always fail here if it has not already failed or exited before it gets to copy9() in its program sequence. You can repeat this example without copy8() in the overwrite.c source program to verify the copy9() failure for align:0.

- The final example is the postfree.c program that attempts to write to freed memory.

```
/*
 * postfree.c: This program attempts to write to freed memory.
 *
 * To build and run this program with Debug Malloc enabled, perform the
 * following steps:
 *
 *   cc -o postfree postfree.c
 *   export MALLOCTYPE=debug
 *   export MALLOCDEBUG=postfree_checking
 *   postfree
 *
 * Upon completion of this program, turn off Debug Malloc and examine the
 * core file as follows:
 *
 *   export MALLOCTYPE=
 *   export MALLOCDEBUG=
 *   dbx postfree
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

main()
{
char *ptr = NULL;

   ptr = (char *) malloc(sizeof(char) * 4096);
   free(ptr);
   strcpy(ptr, "This is an error.");
   exit(0);
}
```

You can use this program in the following sequence:

1. Compile the program, and then execute it to verify that it does not create a core file. That is, it appears to work correctly.

```
# ls -l
total 8
-rw-r--r--   1 root     sys          656 Aug 24 01:00 postfree.c
# make postfree
        cc -O  postfree.c -o postfree
```

```
# ./postfree
# ls -l
total 24
-rwxr-xr-x   1 root     sys         4928 Aug 24 01:01 postfree
-rw-r--r--   1 root     sys          656 Aug 24 01:00 postfree.c
#
```

2. Set up the Debug Malloc environment, and then execute the program again.

```
# export MALLOCTYPE=debug
# export MALLOCDEBUG=postfree_checking
# ./postfree
Segmentation fault(coredump)
# ls -l
total 40
-rw-r--r--   1 root     sys         8115 Aug 24 01:05 core
-rwxr-xr-x   1 root     sys         4928 Aug 24 01:01 postfree
-rw-r--r--   1 root     sys          656 Aug 24 01:00 postfree.c
#
```

3. Turn off the Debug Malloc environment with the commands:

```
# export MALLOCTYPE=
# export MALLOCDEBUG=
```

4. You can use the `dbx` command to analyze the core file.

```
# dbx postfree
Type 'help' for help.
reading symbolic information ...warning: no source compiled with -g

[using memory image in core]

Segmentation fault in strcpy.strcpy [postfree] at 0x1000046c
0x1000046c (strcpy+0x8c) 94e50004       stwu   r7,0x4(r5)
(dbx) where
strcpy.strcpy() at 0x1000046c
main() at 0x10000344
(dbx)
```

The `where` dbx subcommand has provided you with some more information so that you can return to your source code to look for where it is incorrectly writing to freed memory. In this case, the free before the strcpy is incorrect and this is detected by using Debug Malloc.

Chapter 5. Logical Volume Manager Enhancements

AIX 4.3, AIX 4.3.1, AIX 4.3.2, and AIX 4.3.3 received enhancements to the logical volume group scalability, synchronization performance, online backup, and mirroring functions. These changes enhance AIX’s image as a robust and powerful operating system for increasingly demanding customer requirements.

In this chapter, the major new features of logical volume manager are described.

5.1 Logical Volume Synchronization

The following commands now support the -P flag to allow the user to specify the number of LPs to sync. in parallel.

- `/usr/sbin/syncvg`
- `/usr/sbin/lresynclv`

The -P flag is followed on the command line by the number of partitions to be synchronized as follows:

```
syncvg [-i] [-f] [-H] [-P NumParalleLPs] {-l|-p|-v} Name[-P
num_parallel_lps]
```

```
lresynclv [-H] [-P NumParalleLps] -l LVid
```

The valid range for NumParallelLps is 1 to 32. If the number entered is less than one then `num_parallel_lps` defaults to one. If the number entered is greater than 32 then `num_parallel_lps` will be set to 32.

The `mklv` and `chlv` commands were updated in AIX 4.3.0 to allow synchronized updates of volume groups in a concurrent environment. All nodes that share disks must be available at the time the updated command is issued in order for updates to take place. If a system already has an existing LV with the same name as a new one being added to another system, the command will fail. Other conflicts are also detected to provide stable LV updates in a shared environment. All systems must be running AIX 4.3.0 or higher in order to use this enhancement.

5.2 importvg Learning Mode (4.3.2)

A new option has been created for the LVM `importvg` command. This new option, -L for learning mode, is executed on a shared volume group in a cluster. It allows the LVM actions of creation, deletion, or extension performed on one cluster node to be propagated to other nodes connected to the same shared volume group.

```
importvg [-V MajorNumber] [-y  VolumeGroup] [-f] [-c] [-x] | [-L
VolumeGroup] [ -n] [-F] PhysicalVolume
```

The -L flag takes a volume group and learns about possible changes performed to that volume group. Any new logical volumes created as a result of this command inherit the ownership, group identification, and permissions of the /dev special file for the volume group listed in the -y flag.

To use this feature, note the following:

- The volume group must not be in an active state on the system executing the -L flag.
- The volume group's disks must be unlocked on all systems that have the volume group varied on and operational. Volume groups, and their disks, may be unlocked, remain active, and used through the `varyonvg -b -u` command.
- If an active node has both added and deleted logical volumes on the volume group, the -L flag may produce inconsistent results. The -L flag should be used after each addition or deletion rather than being deferred until after a sequence of changes.

Figure 8 shows an example of a multi-tailed system.

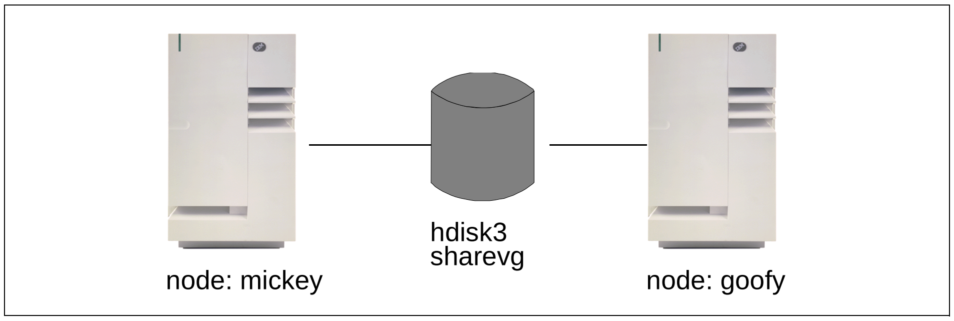


Figure 8. Importvg -L Example

There are two machines, goofy and mickey, that share one disk. The volume group sharevg is created on the shared disk. Both goofy and mickey are aware of the sharevg volume group. The sharevg in node mickey is varied on and in node goofy, it is varied off.

On node mickey, to make the sharevg unlocked, enter:

```
#varyonvg -b -u sharevg
```

On node goofy, to read the LVM information made by node mickey, enter:

```
# importvg -L sharvg datavg hdisk3
```

On node mickey, to return to the normal mode and release the lock, enter:

```
# varyonvg sharevg
```

It should be noted that the volume group sharevg remained on line during the entire operation, therefore, not affecting production work.

5.3 importvg Fast Mode (4.3.2)

A new option -F has been added to `importvg` command. The command syntax is shown in the following example.

```
importvg [-V MajorNumber] [-y  VolumeGroup] [-f] [-c] [-x] | [-L
VolumeGroup] [ -n] [-F] PhysicalVolume
```

It provides a fast version of importvg that checks the Volume Group Descriptor Areas (VGDA) of only the disks that are members of the designated volume group. As a result, if a user exercises this flag, they must ensure that all physical volumes in the volume group are in a good and known state. If this flag is used on a volume group where a disk may be in missing or removed state, the command may fail, or the results may be inconsistent. This flag has the advantage of avoiding a lengthy search for missing disks that happens during normal `importvg` processing. Administration of large SSA disk arrays will greatly benefit from the terse search the -F option provides.

5.4 Raw LV Online Mirror Backup Support (4.3.1)

The LVM in AIX 4.3.1 provides a snap shot capability for raw mirrored logical volumes. One mirror of a mirrored logical volume can be used to archive the data on the raw logical volume without splitting the mirror copies from each other (only the logical partitions that have changed during the system backup need to be resynchronized).

Table 20 lists the new options added to `chlvcopy` command.

Table 20. chlvcopy New Options in AIX 4.3.1

Flag	Description
-b	Mark a mirror copy as an online backup copy.
-c	Identify which mirror copy used as online backup copy. The allowed values of copy are 1, 2, or 3. If this option is not specified, the default for copy is the last mirror copy of the logical volume.
-B	Unmark a mirror as an online backup copy.
-f	Force LV copy to be marked as backup even if there are stale partitions.

5.4.1 Removal of 1016 PPs per Physical Volume Limit (4.3.1)

The support for greater than 1016 PPs per physical volume has been added in AIX Version 4.3.1. To support a VG exceeding the limit of 1016 PPs using the same VGDA and VGSA areas, the number of disks supported in the volume group has been reduced.

The -t flag was added to the `chvg` and `mkvg` commands to convert and create a volume group with multiples of 1016 partitions per disk. This reduces the total number of disks that can be added to the volume group by same fraction. Once a volume group is changed or created to hold more than 1016 physical partitions per disk, it cannot be imported into AIX versions earlier than 4.3.1.

The -t factor allows (factor * 1016) PPs per physical volume. For example, a partition size of at least 16 MB would be needed to create a volume group with a 10 GB disk. Or with at factor size of 2, a smaller partition size of 8 MB can be used. However, this limits the total number of disks that can be added to the volume group. If a factor value is used, a maximum of MAXPVS/factor disks can be included in the volume group.

The relationship of factor -t, PP numbers per physical disk, and the number of disks allowed in one VG is provided in Table 21.

Table 21. Factor -t

Factor t	PP numbers	Number of disks in VG
1	1016	32
2	2032	16

Factor t	PP numbers	Number of disks in VG
3	3048	10
4	4064	8
5	5080	6
6	6096	5
7	7112	4
8	8128	4
16	16256	2

Following is an example of a 4.3 GB disk. The PP size 8 MB is needed for creating the volume group vg1 without the factor -t:

```
# lsvg vg1
VOLUME GROUP:   vg1                VG IDENTIFIER:00091974e54218d7
VG STATE:       active             PP SIZE:      8 megabyte(s)
VG PERMISSION:  read/write              TOTAL PPs:      537 (4296
megabytes)
MAX LVs:       256                     FREE PPs:       537 (4296 megabytes)
LVs:            0                  USED PPs:       0 (0 megabytes)
OPEN LVs:       0                  QUORUM:         2
TOTAL PVs:      1                  VG DESCRIPTORS: 2
STALE PVs:      0                  STALE PPs:      0
ACTIVE PVs:     1                  AUTO ON:        yes
MAX PPs per PV: 1016               MAX PVs:        32
```

When using the factor -t to create vg1, the PP size of 4 MB can be used. Then, the maximum PPs per physical volume becomes 2032, and the maximum physical volumes allowed in the volume group is 16. Following is an example:

```
# mkvg -t 2 -y vg1 hdisk4
0516-1193 mkvg: WARNING, once this operation is completed, volume group vg1
        cannot be imported into AIX 430 or lower versions. Continue (y/n) ?
y
0516-631 mkvg: Warning, all data belonging to physical
        volume hdisk4 will be destroyed.
mkvg: Do you wish to continue? y(es) n(o)? y
vg1

# lsvg vg1
VOLUME GROUP:   vg1                      VG IDENTIFIER:  00091974e54743e9
VG STATE:       active                   PP SIZE:        4 megabyte(s)
VG PERMISSION:  read/write             TOTAL PPs:      1075 (4300
megabytes)
MAX LVs:       256                   FREE PPs:      1075 (4300 megabytes)
LVs:            0                      USED PPs:       0 (0 megabytes)
OPEN LVs:       0                      QUORUM:         2
TOTAL PVs:      1                      VG DESCRIPTORS: 2
STALE PVs:      0                      STALE PPs:      0
ACTIVE PVs:     1                      AUTO ON:        yes
MAX PPs per PV: 2032                   MAX PVs:        16
```

Note

When using the -t flag with `chvg` and `mkvg`, it must be entered from the command line.

5.5 Physical Partition Support (4.3.1)

The support for physical partition sizes of 512 MB and 1024 MB have been added to AIX 4.3.1. Volume groups created with this new size cannot be imported into previous versions of AIX.

In pre-AIX 4.3.1 versions, if you add a new volume group, the physical partition sizes in megabytes you are allowed to chose are: 1, 2, 4, 8, 16, 32, 64, 128, and 256.

For AIX 4.3.1, and later releases, if you add a new volume group, the physical partition sizes in megabytes you are allowed to chose are: 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, and 1024 MBs.

If you select a physical partition size of 512 or 1024 MBs on a AIX 4.3.1 or later system, the volume group created cannot be imported on older versions of AIX.

5.6 Big Volume Groups (4.3.2)

A new volume group (VG) format is added in AIX 4.3.2, which increases the maximum number of disks that can be included in a volume group from 32 to 128. The maximum number of logical volumes in this new volume group format is increased from 256 to 512. This means:

- The maximum physical volume (PV) number in one volume group is increased from 32 to 128 (1024 in the future)
- The maximum logical volume (LV) number in one volume group is increased from 256 to 512 (1024 in the future)

Table 22 provides information about LVM restrictions in several AIX versions.

Table 22. Limitations of LVM

Version	Volume group (per system)	Physical volume (per volume group)	Physical partition (per physical volume)	Logical volume (per volume group)	Logical partition (per logical partition)
AIX 4.3.2	255	128	1016*-t factor	512	32,512
AIX 4.3.1	255	32	1016*-t factor	256	32,512
AIX 4.3.0	255	32	1016*-t factor	256	32,512
AIX 4.2	255	32	1016	256	32,512

The LVM in AIX 4.3.2 supports both the small VG configurations of the previous versions of AIX and the new big VG configuration. A migration path is provided to convert old volume groups to the new volume group format, provided there are sufficient free partitions on each of the physical volumes in the volume group to be allocated.

The following sections explain the changes for the bigger VGDA/VGSA, which describe a volume group to a system completely. The changes needed for commands, library, and the LV device driver are also discussed.

5.6.1 Changes to LVCB

The original design of the VGDA and VGSA limited the number of disks that can be added to the volume group at 32 and the total number of logical volumes at 256 (including one reserved for LVM internal use). With the increasing use of disk arrays, the need for the increased capacity for a single volume group is greater.

Following are the basic concepts of VGDA, VGSA, and LVCB.

VGDA Stands for volume group descriptor area. The VGDA contains information that describes the mapping of physical partitions to logical partitions for each logical volume in the volume group, as well as other vital information, including a time stamp. The VGDA is stored on each physical volume of the volume group.

VGSA Stands for volume group status area. VGSA contains information, such as which physical partitions are stale and which physical volumes are missing (that is, not available or active), when a vary-on operation is attempted on a volume group. The VGSA is stored on each physical volume of the volume group.

LVCB Stands for logical volume control block. The LVCB is the first 512 bytes of a logical volume. This area holds important information, such as the creation date of the logical volume, information about mirrored copies, and possible mount points in the journaled file system (JFS). Certain LVM commands are required to update the LVCB as part of the algorithms in LVM.

The Logical Volume Control Block has been moved from the first block of the logical volume to inside the VGDA for better preservation. Though database programs that use logical volumes as raw devices skip this block, obliteration of the LVCB has caused confusion and loss of information such as intra-policy, inter-policy, upperbound, and so on. Since other subsystems, such as diagnostics, IPL, and ROS, do use the LVCB without using the LVM access routines, the LVCB will be maintained at both places.

5.6.2 General Enhancements for Big VG

The following sections describe the general limitations and updates required to implement big VG support on AIX.

5.6.2.1 Command Changes

To support the big VG format, some new options have been added to the commands `mkvg`, `chvg`, `importvg`, `mklv`, and `chlv` commands.

The mkvg Command

The following lists some of the major changes to the `mkvg` command:

- The new option -B creates a big VG format volume group. This can accommodate up to 128 physical volumes and 511 logical volumes (one reserved for LVM internal use).
- If you do not use the -B option, the `mkvg` command will create the a VG with 1016*factor(-t) physical partitions and 32/factor(-t) disks per volume group.
- The option -G creates the volume group with enough space reserved at the beginning of the disk to expand the VGDA to include 1024 disks in the future without having to migrate the physical partitions.

Following is an example of the `mkvg` command. To create a big VG, testvg, on hdisk1, enter:

```
# mkvg -B -y testvg hdisk1
```

To see the attributes of this VG, enter:

```
# lsvg testvg
VOLUME GROUP:   testvg             VGIDENTIFIER:061515169c44a3e
VG STATE:       active             PP SIZE:        4  megabyte(s)
VG PERMISSION:  read/write         TOTAL PPs:      80  (320megabytes)
MAX LVs:        512                FREE PPs:       80  (320 megabytes)
LVs:            0                  USED PPs:       0  (0 megabytes)
OPEN LVs:       0                  QUORUM:         2
TOTAL PVs:      1                  VG DESCRIPTORS: 2
STALE PVs:      0                  STALE PPs:      0
ACTIVE PVs:     1                  AUTO ON:        yes
MAX PPs per PV: 1016               MAX PVs:        128
```

This example shows the new limit values for big VG: MAX PVs is 128 and MAX LVs is 512.

The chvg Command

The option -B converts a small VG to a big VG. Once all the logical volumes are in closed/synced state (file systems unmounted), and if all the physical volumes are in the ACTIVE state in the volume group, the -B flag can be used to convert the small VG to a big VG format. This operation expands the VGDA/VGSA to change the total number of disks that can be added to the volume group from 32 to 128.

If you want to convert the rootvg, you will get the following error message:

```
# chvg -B rootvg
0516-1212 chvg: rootvg cannot be converted to the big volume group format.
0516-732 chvg: Unable to change volume group rootvg.
```

If both -t and -B flags are specified, factor will be updated first, and then the VG is converted to the big VG format (sequential operation).

First create a small VG, testvg, on hdisk1:

```
#mkvg -y testvg hdisk1
```

To see the small VG information, enter:

```
#lsvg hdisk1
VOLUME GROUP:   testvg            VG IDENTIFIER:  00615151692724b1
VG STATE:       active            PP SIZE:        4 megabyte(s)
VG PERMISSION:  read/write        TOTAL PPs:      81 (324 megabytes)
MAX LVs:        256               FREE PPs:       81 (324 megabytes)
LVs:            0                 USED PPs:       0 (0 megabytes)
OPEN LVs:       0                 QUORUM:         2
TOTAL PVs:      1                 VG DESCRIPTORS: 2
```

```
STALE PVs:      0               STALE PPs:      0
ACTIVE PVs:     1               AUTO ON:        yes
MAX PPs per PV: 1016            MAX PVs:        32
```

To convert a small VG into a big VG:

```
# chvg -B test
0516-1224 chvg: WARNING, once this operation is completed, volume group
test
cannot be imported into AIX 431 or lower versions. Continue (y/n) ?
y
0516-1164 chvg: Volume group testvg changed. With given characteristics
testvg can include up to 128 physical volumes with 1016 physical partitions
each physical volume.
```

To see the attributes of this VG, enter:

```
# lsvg test
VOLUME GROUP:   test             VG IDENTIFIER:  00615151692724b1
VG STATE:       active           PP SIZE:        4 megabyte(s)
VG PERMISSION:  read/write       TOTAL PPs:      81 (324 megabytes)
MAX LVs:        512              FREE PPs:       79 (316 megabytes)
LVs:            0                USED PPs:       2 (8 megabytes)
OPEN LVs:       0                QUORUM:         2
TOTAL PVs:      1                VG DESCRIPTORS: 2
STALE PVs:      0                STALE PPs:      0
ACTIVE PVs:     1                AUTO ON:        yes
MAX PPs per PV: 1016             MAX PVs:        128
```

As shown, the number of TOTAL PPs remain unchanged. The number of free PPs are reduced by two. These two PPs are reserved for the larger VGDA/VGSA.

If you do not have enough space on your disk, suppose disk1 on the small VG is full, such as:

```
# lspv hdisk1
PHYSICAL VOLUME:    hdisk1              VOLUME GROUP:     testvg
PV IDENTIFIER:      00615151648abe10    VG IDENTIFIER:006151516a0af1a9 PV
STATE:              active
STALE PARTITIONS:   0                    ALLOCATABLE:      yes
PP SIZE: megabyte(s)                     LOGICAL VOLUMES:  2
TOTAL PPs:          81 (324 megabytes)   VG DESCRIPTORS:   2
FREE PPs:           0 (0 megabytes)
USED PPs:           81 (324 megabytes)
FREE DISTRIBUTION:  00..00..00..00..00
USED DISTRIBUTION:  17..16..16..16..16
```

The following example shows what happens if you want to convert the small testvg into big testvg:

```
# chvg -B testvg
0516-1214 chvg: Not enough free physical partitions exist on hdisk1 for the
expansion of the volume group descriptor area. Migrate/reorganize to free
up 2 partitions and run chvg again.
0516-732 chvg: Unable to change volume group testvg.
```

You must migrate or reorganize the volume group using `migratepv` or `reorgvg` to free up enough physical partitions for the system to expand the VGDA/VGSA.

The importvg Command

The option -R restores the ownership, group ID, and permissions of the logical volume special device files. These values will be restored only if they were set using -U, -G, and -P flags of `mklv` or `chlv` commands. The -U, -G, and -P flags are for root to define the ownership, group, and permissions of the LV you are creating respectively. This flag is applicable only for big VG format volume groups.

The mklv Command

If you create a logical volume in a big VG, you can use the following three new options (using root privileges):

- Option -U specifies the user ID for logical volume special file.
- Option -G specifies the group ID for the logical volume special file.
- Option -P specifies the permissions (file modes) for the logical volume special file.

The chlv Command

The three new options are the same with `mklv` command (using root privileges):

- Option -U specifies the user ID for logical volume special file.
- Option -G specifies the group ID for the logical volume special file.
- Option -P specifies the permissions (file modes) for the logical volume special file.

Note

When using the above new options in `mkvg`, `chvg`, `importvg`, `mklv`, and `chlv` commands, the commands must be entered from the command line. There are no `smit` or `wsm` interfaces for them.

5.6.2.2 Header File Changes

The following header file was changed to support big VGs:

- lvmrec.h

5.6.2.3 Default Maximum PPs for Each Physical Volume - 1016

No matter if you create a big VG or small VG, the `mkvg` command will still use 1016 as the default value for the number of physical partitions per physical volume. If you use the -t (factor) option together with the big VG option, you can create the volume group with the desired partition size and number of partitions.

The -t volume group factor was first introduced in AIX 4.3.1. See 5.5, “Physical Partition Support (4.3.1)” on page 176 for reference. The number of physical partitions calculated is 1016 * t factor per physical volume. The size for each of the physical partition is up to 1024 MB.

The `mkvg` command, by default, creates a volume group that can accommodate 255 logical volumes and 32 physical volumes (disks). These limits can be extended to 511 logical volumes and 128 physical volumes by specifying the -B flag.

5.6.3 Small VG to Big VG Conversion

To convert the small VG to a big VG, a number of free physical partitions are needed to expand the VGDA/VGSA. Depending on the size of the physical partition and the current size of the VGDA, the number of partitions required are calculated. Since the first partition starts immediately after the end of the secondary VGDA, if it is occupied by a logical partition, it will be migrated to another free physical partition on the disk. This first physical partition will then be removed from the list of available partitions (not be moved or allocated for any reason), and the remaining partitions will be renumbered. After the conversion, the total number of physical partitions on the disk will not change, except that the extra partitions allocated for the VGDA are marked non allocatable.

You should be aware of the following items when you perform a conversion from a small VG to a big VG:

- All the disks in the volume group should have enough free PPs for the conversion to be possible.
- All the logical volumes in the volume group must be in closed/synced state.
- All physical volumes must be available and be in the ACTIVE state.

- The volume group is varied on in management mode to prevent opening of any logical volumes.
- The ownership/permissions of special device files will only be preserved if `mklv` or `chlv` are used with the -U, -G, or -P flags.

Currently, the `migratepv` command does not allow the migration of individual physical partitions. The conversion needs to free up just enough physical partitions from the beginning of the disk to elsewhere. The current implementation will try to migrate the partitions within the physical volume, and the user must move the partitions to other disks in the volume group.

5.6.4 Big VG Limitations

The following list are the limitations of a big VG:

- A big VG is not enabled for concurrent access. This is posed by the communication path used by the concurrent logical volumes. It will be prohibitively slower for the big VG to communicate across nodes due to an increase in the number of disks.
- The rootvg cannot be converted to the big VG format.
- A big VG cannot be imported or activated on pre-AIX 4.3.2 levels.

5.7 Concurrent Online Mirror Backup and Special File Support (4.3.2)

AIX 4.3.1 provided support for an on line backup mechanism for a mirrored raw logical volume. But it lacked support for file system access and restricts concurrent mode access for the volume group.

AIX 4.3.2 enhances the capabilities of the online backup in AIX 4.3 to support:

- Concurrent mode volume groups
- Filesystem and database access

Note that file system access does not mean JFS access. This enhancement to the LVM still requires additional steps (such as unmounting a file system) to prevent data loss.

This new feature is used for HACMP concurrent mode (mode 3). While in concurrent mode, you can designate a mirror copy in a VG as a backup copy to archive the data on the raw logical volume without affecting the other mirror copies. It improves the system availability to end users.

Use a second LV and special device to allow access to the backup mirror copy. All I/O and ioctls coming to this second LV would be routed to the actual logical volume to be serviced. A number of changes were made to VGDA to support this new type of LV.

If the LV contains a file system, there will be two serial writes to support the new mount point. In order to support this function, there are updates to the LVCB and the superblock of the new file system.

5.7.1 Limitations

Following are some limitations:

- The original logical volume cannot be removed while the on line backup copy exists. No changes are allowed to the original logical volume structure, or attributes, while the on line backup exists. But you still can make changes to the file system mounted over the logical volume.
- All partitions of a logical volume must be fresh before you mark a mirror copy as an on line backup. Only one copy may be designated as an on line backup copy.
- This function is currently documented only in the man pages. Use at your own risk.

5.7.2 Commands Changed

Using the `chlvcopy` command, you can mark, or unmark, a mirror copy as an on line backup copy and change the backup mirror copy characteristics for a logical volume.

The syntax is:

```
chlvcopy -B | { -b [ -c copy ] [ -f ] [ -P ] [ -l newlvname ] [ -w ] } LV
name
```

Note

Although the `chlvcopy` command can mark online backup copies on logical volumes that are open (including logical volumes containing mounted file systems), this is not recommended unless the application is at a known state at the time the copy is marked as a backup. The backup copy is internally consistent at the time the `chlvcopy` command is run, but consistency is lost between the logical volume and the backup copy if the logical volume is accessed by multiple processes simultaneously, and the application is not at a known state. When marking an open logical volume, data may be lost or corrupted. Logical volumes should be closed before marking on line backup copies in order to avoid a potential corruption window.

The -P, -l, and -w are new options for AIX 4.3.2.

If the persistence flag -P is not set to prevent the loss of backup data, the volume group should be set to not automatically varyon, and the -n flag should be used with varyonvg to prevent stale partitions from being resynced. If the persistence flag -P is set, the following applies: in the event of a crash while an on line backup copy exists (or multiples exist), the existence of copies is retained when the system is rebooted.

Use the -l or -P flag to prevent the volume group from being unstable on prior releases of AIX.

Table 23 lists the new options of `chlvcopy` command in AIX 4.3.2.

Table 23. New Options for chlvcopy Command in AIX 4.3

Flag	Description
-P	Maintains information about the existence of an online backup copy across a reboot and also allows other nodes (in a concurrent mode environment) to be aware of the existence of the online backup(s).
-l	New name of the backup logical volume. If one is not provided, one will be created by the system.
-w	Allow backup copy to be writable (default is to create the backup copy as READ ONLY)

To create an on line backup, perform the following steps:

1. Unmount the file system from the mirrored LV
2. Execute chlvcopy -b -c <#> -l <newlvname> <original_lvname>

3. Remount the original file system
4. Execute mount -o ro /dev/<newlvname> /<backupfs>
5. Backup the backupfs
6. Execute unmount /<backupfs>
7. Execute chlvcopy -B original_lvname

In general, use `chlvcopy` the same as you would `splitlvcopy`.

5.8 Online JFS Backup (4.3.3)

Making an online backup of a mounted JFS file system creates a snapshot of the logical volume that contains the file system while applications are still using the file system. Be aware, though, that since the file writes are asynchronous, the snapshot may not contain all data that was written immediately before the snapshot is taken. Modifications that start after the snapshot begins may not be present in the backup copy. Therefore, it is recommended that file system activity be minimal while the split is taking place.

5.8.1 Split Off a Mirrored Copy

In order to make an online backup of a mounted file system, the logical volume that the file system resides on must be mirrored. The JFS log logical volume for the file system must also be mirrored. The number of copies of the jfs log must be equal to the number of copies of the file systems's logical volume.

To split off the mirrored copy of the file system, use the `chfs` command and you can control which copy is used as the backup by using the copy attribute. The second copy is the default if a copy value is not specified.

The following example shows a copy of the file system /testfs split off. The example assumes that there are three copies of the file system and three copies of the JFS log, as you can verify using:

```
# lsvg -l testvg
testvg:
LV NAME             TYPE       LPs   PPs   PVs  LV STATE       MOUNT POINT
loglv00             jfslog     1     3     3    open/syncd     N/A
lv00                jfs        10    30    3    open/syncd     /testfs
```

Issuing the `chfs` command you can use the third copy of /testfs to produce a new /backup file system:

```
chfs -a splitcopy=/backup -a copy=3 /testfs
```

Once this command completes successfully, a copy of the file system is available read-only in */backup*. Remember that additional changes made to the original file system after the copy is split off are not reflected in the backup copy.

Note

Splitting a mirrored copy of a file systems means that one copy is temporary dedicated to backup activities and is not available to provide data availability. It is recommended that you have three mirrored copies of the file system so you can recover a disk problem before the backup copy has been reintegrated on the file system.

The /backup file system has been created pointing to the third copy of the original file system. If you issue again the `lsvg` command you can see what has changed:

```
# lsvg -l testvg
testvg:
LV NAME             TYPE       LPs   PPs   PVs  LV STATE      MOUNT POINT
loglv00             jfslog     1     3     3    open/syncd    N/A
lv00                jfs        10    30    3    open/stale    /testfs
lv00copy00          jfs        0     0     0    open???????   /backup
```

The /testfs file system is still made by a logical volume with three copies, but now it has *stale* partitions, since one copy is no longer kept in sync with the other and is made available for read-only access through the /backup mount point. Note the new lv00copy00 logical volume: it is in an *open* state, it cannot be in *sync* or in *stale* so that is why there are the question marks, and no logical partitions are assigned to it since it only points to a specific copy of lv00.

The /backup file system has been mounted and is available to use. If you issue the `mount` command you can see it mounted with read-only permissions and without any JFS log.

```
# mount
  node       mounted        mounted over    vfs       date        options
-------- ---------------  ---------------  ------ ------------ ---------------
         /dev/hd4         /                jfs    Jul 29 16:15 rw,log=/dev/hd8
         /dev/hd2         /usr             jfs    Jul 29 16:15 rw,log=/dev/hd8
         /dev/hd9var      /var             jfs    Jul 29 16:15 rw,log=/dev/hd8
         /dev/hd3         /tmp             jfs    Jul 29 16:15 rw,log=/dev/hd8
         /dev/hd1         /home            jfs    Jul 29 16:16 rw,log=/dev/hd8
         /dev/lv00        /testfs          jfs    Jul 29 16:35 rw,log=/dev/loglv00
         /dev/lv00copy00  /backup          jfs    Jul 29 16:40 ro
```

Now it is possible to read the data in /backup.

When a copy of a file system is split off, some activity is made on the jfslog in order to keep consistent the structure of the original file system and the read-only copy. During such operation no other splitting must be done on file systems that use the same jfslog. They must be delayed until the previous split is finished.

For example, consider the case of two file systems, /team1 and /team2, that use the same jfslog. If you want to make a online backup of both on the same tape drive, you have to split first /team1, wait for the read-only copy to be available and then you may split /team2. Finally you can do the backup. If, for some reason, you split both file systems in the same moment, one split will succeed and the other will fail.

5.8.2 Reintegrate a Mirrored Backup Copy

Once a backup has been made, the copy can be reintegrated as a mirrored copy unmounting the backup file system and using the `rmfs` command. All the mirrored copies are put in sync automatically.

To restore the copy used in the previous section, you can simply execute the following commands:

```
# umount /backup
# rmfs /backup
rmlv: Logical volume lv00copy00 is removed.
# lsvg -l testvg
testvg:
LV NAME             TYPE       LPs   PPs   PVs  LV STATE      MOUNT POINT
loglv00             jfslog     1     3     3    open/syncd    N/A
lv00                jfs        10    30    3    open/syncd    /testfs
```

5.9 Mirroring and Striping Support (4.3.3)

Starting from AIX 4.3.3 it is possible to create logical volumes that are both mirrored and striped, providing RAID 0+1 capability. Performance improvement of striping technique can be combined with the availability provided by mirroring. A typical example is given in Figure 9 on page 189, where a logical volume is striped on three disks and it also have another copy on other three disks.

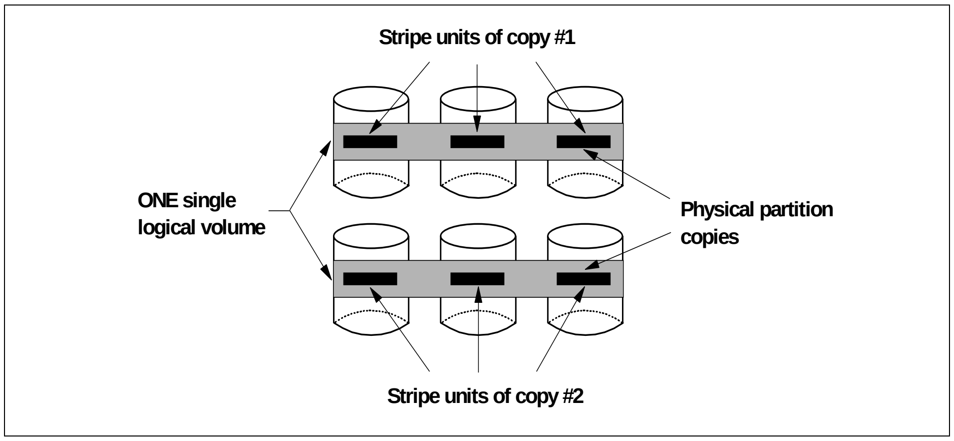


Figure 9. Mirrored and Striped Logical Volume

If no mirror is present, a failure on one disk causes all the striped logical volume content to be unavailable. Introducing the second (or third) copy of the logical volume, the failure of a disk does not affect access to data as long as there is one available copy of data on another disk.

Since this is a new feature of AIX 4.3.3, the volume group in which a mirrored and striped logical volume is created cannot be imported by any other machines running previous version of AIX. When you create such logical volume, a warning message is prompted and you are requested to confirm your choice.

From an administrator point of view, very few changes have been made to LVM command interface in order to make mirror and striping available on the same logical volume. Logical volumes are created like in previous releases but copies may be specified, added, and removed when the logical volume is striped.

A new concept of allocation policy is added to LVM. Since striped logical volumes are very sensitive to the loss of a disk, it is important to force them to have mirrored copies to different physical volumes and not to have partitions allocated for one copy on the same physical volume with the partitions from another copy.

This new allocation policy is called *super strict*. It is mandatory for striped and mirrored logical volumes, but it can be chosen also for other logical volume types. Every LVM command that have the -s flag to define the allocation policy now may have one of the following values:

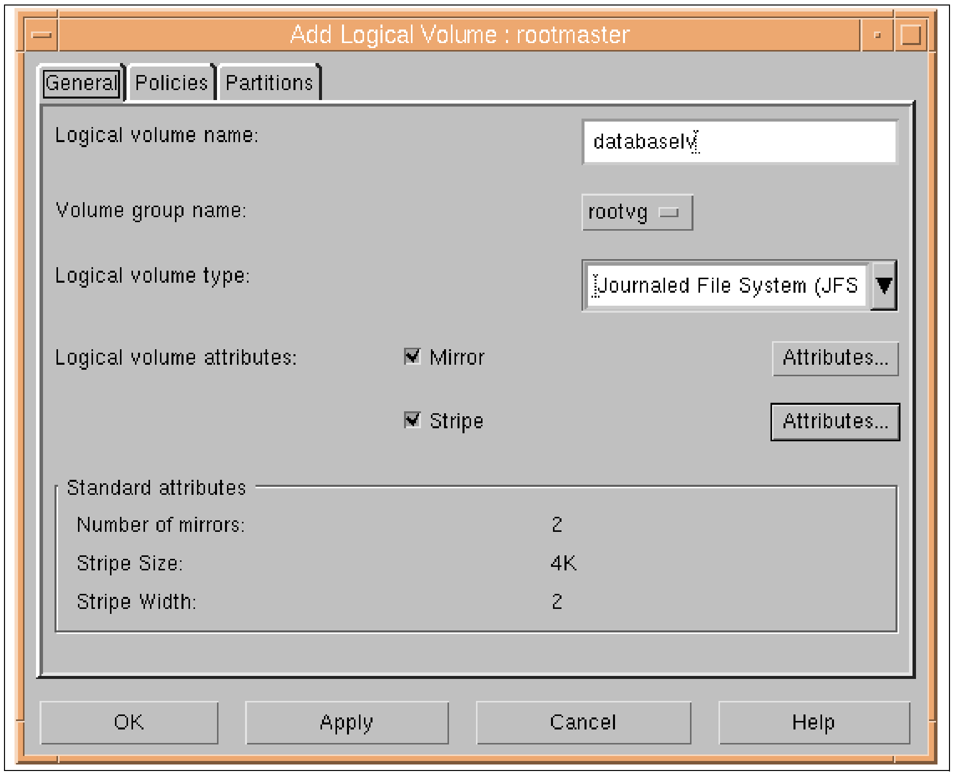
-s y	Strict allocation policy: copies for a logical partition cannot share the same physical volume.
-s n	No allocation policy: copies for a logical partition can share the same physical volume.
-s s	Super strict allocation policy: the partitions allocated for one mirror cannot share a physical volume with the partitions from another mirror.

You can either create a new striped and mirrored logical volume with a single command or create first a striped logical volume and then you create the copies.

Note

If you create a striped logical volume with SMIT or Web-Based System Manager, the administration tool ensures that the super strictness policy is applied. On the other hand, if you use `mklvcopy` to create the mirrored copies, you *must* remember to use the -s s option to have super strictness policy. If you do not specify the policy the copies will be created but without the correct policy and you may loose data in case of disk failure.

Web-Based System Manager has been also modified in order to make user create mirrored and striped logical volumes. An example of a Web-Based System Manager panel is in Figure 10 on page 191.



Logical volume name: databaselv

Volume group name: rootvg

Logical volume type: Journaled File System (JFS

Logical volume attributes: [✓ Mirror] [✓ Stripe]

Standard attributes

Number of mirrors:	2
Stripe Size:	4K
Stripe Width:	2

Figure 10. Logical volume creation with Web-Based System Manager

In order to replace a failed disk that contains a striped and mirrored logical volume, you should use the `replacepv` command. If your failed disk is, for example hdisk5 and you have a spare disk hdisk20, you must first be sure that hdisk20 does not belong to any volume group, then you can issue the following command:

```
# replacepv hdisk5 hdisk20
```

Chapter 6. System Management and Utilities

Web-Based System Manager is an AIX system administration tool for administering an AIX host locally or over the Internet. Web-Based System Manager is the next step in the evolution of AIX system administration tools. This evolution has included System Management Interface tool (SMIT), Motif SMIT, Distributed SMIT, and Visual System Manager (VSM). SMIT and VSM have been major contributors to customer satisfaction regarding AIX system management and usability. It is an objective for Web-Based System Manager to encompass the system administration capabilities and surpass the usability of these predecessors.

The objectives of Web-Based System Manager are:

- Simplification of AIX administration by a single new interface.
- Enable AIX systems to be administered from almost any client platform.
- Enable AIX systems to be administered remotely.
- Provide an administrative environment that exploits the similarities of the Windows 95/NT and AIX CDE desktop environments so that users of the system will find a large degree of look and feel consistency between Web-Based System Manager and these two primary desktop environments.

Web-Based System Manager provides a comprehensive system management environment and covers most of the tasks in the SMIT user interface.

Note: SMIT continues to fulfill the need for system administration from an ASCII terminal.

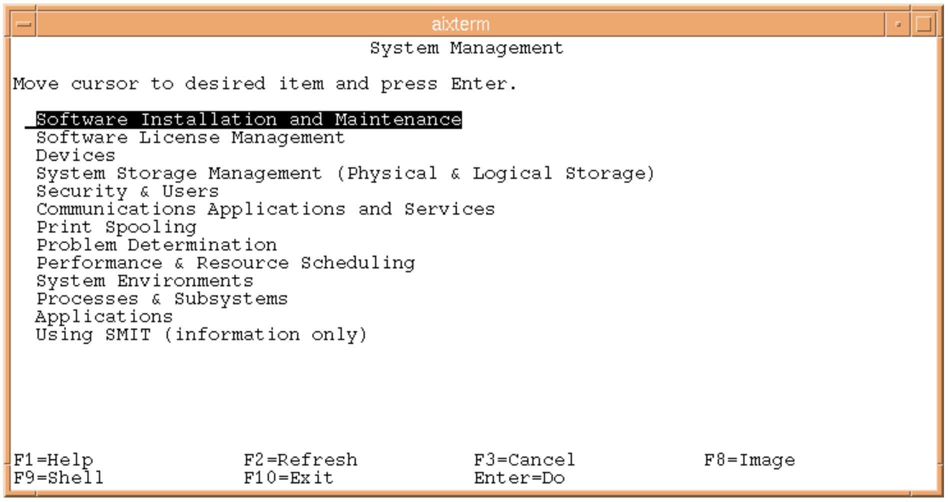
6.1 Overview of Existing AIX Systems Management

Since the introduction of AIX Version 3.1 there have been a number of system management tools available to help system administrators manage their installations. These include SMIT, DSMIT, and VSM.

6.1.1 SMIT Overview

SMIT was introduced in AIX Version 3.1. It provides a menu-driven interface for approximately 160 local system management tasks. In SMIT, the user is guided by a series of interactive menus and dialogs that automatically build, execute, and log commands required to accomplish selected operations. SMIT eliminates the need to know, or learn, command-level syntax for system management tasks. SMIT is easily extendable, and many LPPs and customers have added their own SMIT menus and dialogs.

Figure 11 shows the default SMIT menu, as seen on an X-based display.



```
                                   System Management

Move cursor to desired item and press Enter.

  Software Installation and Maintenance
  Software License Management
  Devices
  System Storage Management (Physical & Logical Storage)
  Security & Users
  Communications Applications and Services
  Print Spooling
  Problem Determination
  Performance & Resource Scheduling
  System Environments
  Processes & Subsystems
  Applications
  Using SMIT (information only)




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

Figure 11. Default SMIT Menu

SMIT provides a character-based interface and a graphical interface. The character-based interface, ASCII SMIT (as shown in Figure 11), can be run on a TTY terminal, while the graphical interface, Motif SMIT (as shown in Figure 12), requires an X Windows compatible graphical display. Motif SMIT provides a point and click interface to the SMIT menus.

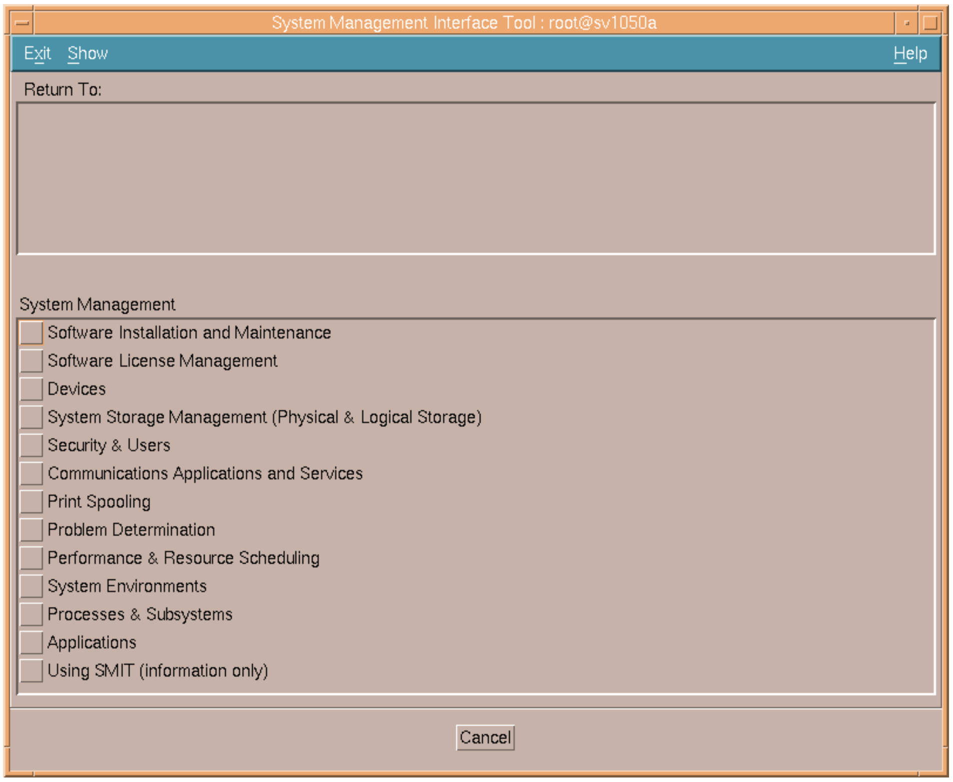

System Management Interface Tool : root@sv1050a
Exit Show
Help
Return To:
System Management
Software Installation and Maintenance
Software License Management
Devices
System Storage Management (Physical & Logical Storage)
Security & Users
Communications Applications and Services
Print Spooling
Problem Determination
Performance & Resource Scheduling
System Environments
Processes & Subsystems
Applications
Using SMIT (information only)
Cancel

Figure 12. Default Motif SMIT Menu

6.1.2 DSMIT Overview

DSMIT (Distributed SMIT) makes the function of SMIT available in a distributed systems environment. DSMIT allows the administrator to perform SMIT tasks across multiple systems simultaneously from a single point of control. Concurrent and sequential modes of execution are supported. DSMIT provides both the ASCII and graphical user interfaces of SMIT.

DSMIT is a Licensed Program Product (LPP) originally introduced on AIX 3.2.5 and later enhanced and released for AIX V4. In the most recent version, Version 2.2, DSMIT provides an ongoing secure operation including the secure modification of the security configuration and updates of passwords and keys. DSMIT security is based on well established cryptographic routines and DSMIT-specific (modeled after Kerberos 5) communication protocols. DSMIT security features include integrated sign on, authentication, data integrity, data confidentiality, and logging.

Like SMIT, the DSMIT menus and dialogs are easily extendable. Furthermore, DSMIT provides a command line interface that allows the user to run commands, scripts, and programs of their choice on distributed systems. The command line interface allows the user to exploit the capability of DSMIT beyond the provided menus and dialogs without adding additional menus or dialogs. The command line interface also supports interactive commands in the sequential mode of execution. For example, you can run ksh and perform interactive tasks over the secure DSMIT connection.

DSMIT also extends the function of SMIT to heterogeneous systems, with agents available for managing SunOS 4.1.3, Solaris 2.3, and HP-UX 9.0, although the DSMIT agents have not remained current with new releases of Solaris and HP-UX.

6.1.3 VSM Overview

VSM (Visual System Manager) is a graphical user interface that enables you to perform system management tasks through the direct manipulation of objects (icons). Due to VSMs drag and drop graphical interface, you do not need to have a complete understanding of the AIX commands.

VSM was originally introduced as part of AIX 3.2.5 and was enhanced on AIX Version 4. VSM is composed of independent application programs that currently include:

- Device Manager
- Print Manager
- Storage Manager (as shown in Figure 13)
- Users and Groups Manager
- Install and Update Software Manager
- Set Date and Time
- Schedule a Job
- Remove or View Scheduled Jobs
- Maintain Installed Software
- RAID Device Manager
- NIM Install Manager

Figure 13 shows a sample VSM dialog.

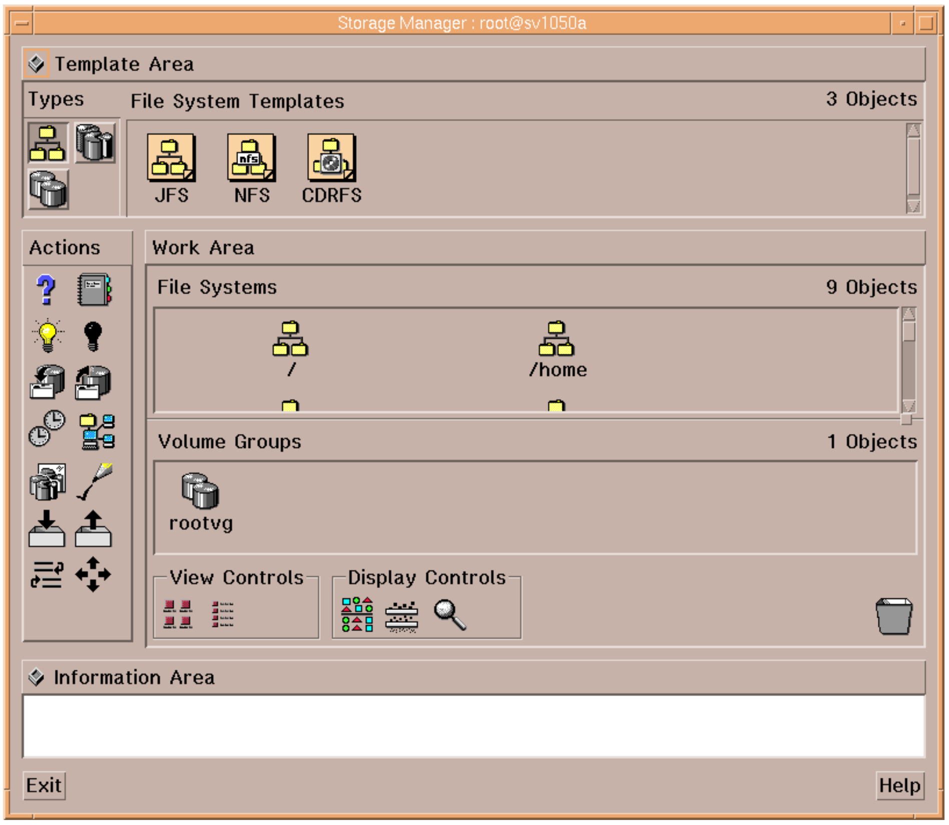


Figure 13. Sample VSM Storage Manager

6.2 Web-Based System Manager Architecture

Web-Based System Manager enables a system administrator to manage an AIX machine either locally from a graphics terminal or remotely from a PC or RS/6000 client. Information is entered through the use of GUI components on the client side. The information is then sent over the network to the Web-Based System Manager server, which runs the commands necessary to perform the required action.

Web-Based System Manager is implemented using the Java programming language. The implementation of Web-Based System Manager in Java provides:

- Cross-platform portability. Any client platform with a Java 1.1-enabled Web browser is able to run a Web-Based System Manager client object.

- Remote administration. A Web-Based System Manager client is able to administer an AIX machine remotely through the Internet.
- A richer and more flexible GUI environment than is available with either HTML forms or Java Script.

Java programs can be delivered as applets that require a Web browser to download the executable code or as stand-alone applications that are stored locally and run independently of a browser or viewer.

Web-Based System Manager has been packaged in a browser-based applet mode, and for local execution, an application mode has been implemented. The application mode uses the AIX Java Virtual-Machine that, in turn, executes the Java applications as threads on the system.

Note: When referring to Java applications, the term *application* is used differently than the conventional use of application as in word processing application or in the discussion on application-oriented user interfaces below. Java application refers to the manner in which Java code is invoked.

6.2.1 Web-Based System Manager Components

Web-Based System Manager includes the following components:

- Backups
- Devices
- File Systems
- Network (interfaces for configuring network communications)
- Printer Queues
- Processes
- Registered Applications
- Software (installable software, software installed, and objects related to installation)
- Subsystems
- System (user interface, console, date/time, language, OS characteristics, system logs, and dump devices)
- Users
- Volumes

6.2.2 Web-Based System Manager User Interface

The Web-Based System Manager user interface is an Object-Oriented User Interface (OOUI). OOUIs are distinguished from traditional, application-oriented user interfaces, in that the user focuses on readily identifiable things on which the user works. In an application-oriented environment, the user focuses on a tool for manipulating the work. Some examples may clarify the distinction. In a document processing context with an application-oriented interface, the user focuses on the tool (a word processing program). While in OOUI, the user focuses on the object of the task itself (the document). In a system management context, an application-oriented interface would require the user to learn management tools (for example, a Print Manager application), while an OOUI would enable the user to directly manipulate a representation of the managed object (for example, a printer or group of printers).

In the evolution of AIX system management user interfaces, SMIT was an application-oriented interface, and VSM was a mixed application/object-oriented interface. Web-Based System Manager is intended to significantly increase the object-orientation of system management of AIX.

The reasons for this approach, as opposed to an application-oriented user interface are:

- By focusing on objects rather than tools for manipulating objects, OOUIs are more direct and require less learning than application-oriented GUI's.
- OOUIs (especially if implemented using object-oriented programming techniques) have a more consistent user interface than application-oriented interfaces, further reducing the amount of learning required by the user.
- OOUIs follow the current trends in user interface development. User interface styles such as CDE, OS/2, and Windows reflect a trend of increasing object-orientation.

6.2.3 Web-Based System Manager Launch Interfaces

Web-Based System Manager has been implemented in a modular fashion so that it can be accessed from a variety of launch points. Some launch points are:

- A Web-Based System Manager launch page running inside a Java-enabled browser.

- A Web-Based System Manager application icon in the CDE application manager.

The launch pad icon in the CDE Application Manager loads the Web-Based System Manager environment with launch icons for all of the Web-Based System Manager applications. Multiple applications can be started without the need to restart the Web-Based System Manager environment each time.

The remote launch pad icon in the CDE application manager enables the administrator to login to another AIX 4.3 host and manage it with Web-Based System Manager.

The command line allows Web-Based System Manager to be in X Windows from an aixterm window or in the CDE desktop from a dtterm window.

When an applet is invoked from a Web-Based System Manager launch interface it appears as a Java frame (essentially a child widow) above the launch interface. Additional dialogs are always opened as child windows. The initial Web-Based System Manager frame appears in Figure 14.

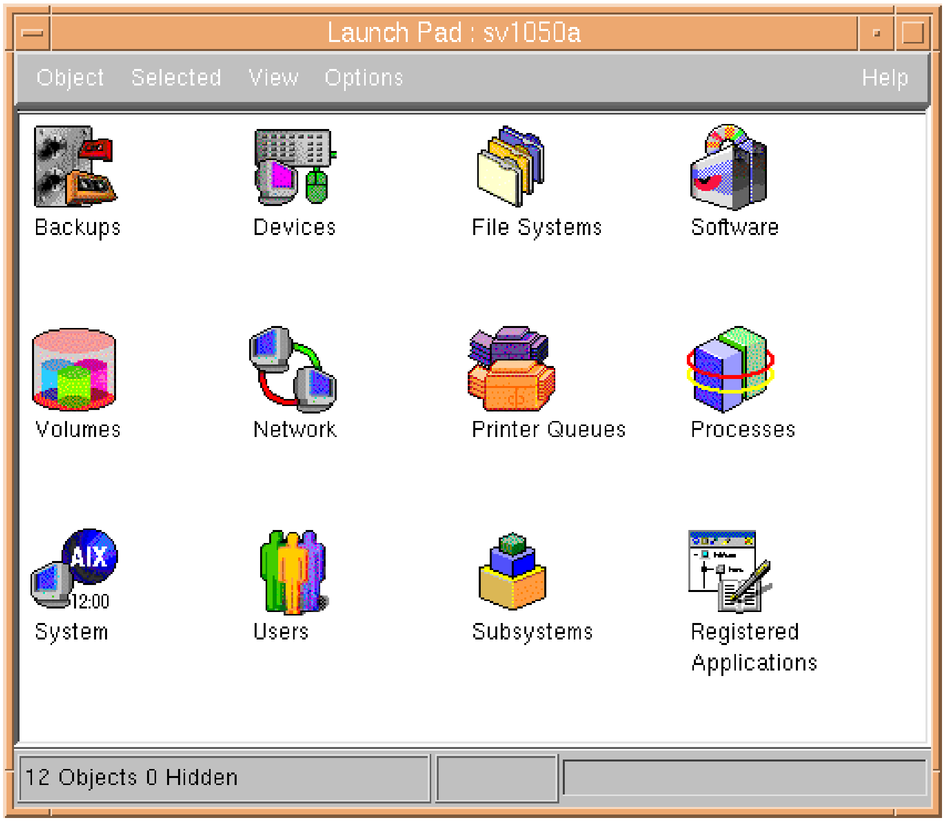


Figure 14. Web-Based System Manager Launch Interface

6.2.4 Web-Based System Manager User Interface Objects

Many system administration and configuration tasks are performed by interacting with simple objects. Simple objects represent individual managed objects that cannot be further decomposed into a collection of objects.

Each instance of a simple object in the system is represented as an icon in a container's view area. Double-clicking on a simple object opens the object so that administrative tasks may be performed. The Web-Based System Manager user interface consists of the following hierarchy of objects:

Container Objects

Container objects include other container objects and simple objects representing elements of system resources to be configured and managed.

Objects
Objects include the following user interface classes:

Property Notebook Objects
Property notebooks (tabbed dialogs) are used for displaying and changing settings associated with a managed resource or container. Property notebooks are useful because they can organize a large collection of system settings into individual pages. They are used in Web-Based System Manager for viewing configuration settings and for configuration change tasks in which there is no predefined order of steps for the user to perform.

TaskGuide Objects
TaskGuides are dialogs designed to assist the user in performing complex tasks. Unlike property dialogs, TaskGuides lead the user through a task in an ordered series of steps.

6.2.4.1 Container Objects

Simple objects are viewed and manipulated in container objects. Containers consist of a view area, objects within the view, and a group of actions. Actions apply to the container view area and individual objects in the view area.

Web-Based System Manager containers are specialized, that is, each type of Web-Based System Manager simple object (for example, user) is viewed within its own container type (for example, the Users container). The classification of containers provides rules for actions to be applied to included objects and any specialized views of objects within the container type.

Containers are viewed and manipulated in their own primary windows. Container windows perform the functions of CDE and Windows NT file manager folders. All the container windows support the following behaviors:

Open	Iconized containers may be opened to reveal their contents.
Close	Containers may be closed, returning control to the parent container (or in the case of the top-level container, exiting the application).
Find	Find an object by specifying its name.
Reload Now	Rediscover the objects and their current states.
Stop Loading	Halt the loading, or reloading, of a container.
Select All	Select all objects in a container.
Deselect All	Remove selection from all objects in a container.

Scroll When containers are sized such that their entire contents cannot be shown, they can be scrolled up/down and right/left.

6.2.4.2 Open Action

Opening a container causes the container view area to be populated with objects.

6.2.4.3 TaskGuides

TaskGuides are the IBM/Lotus equivalent of Wizards. Wizards are a Microsoft tool for assisting users in performing complex tasks. They direct the user through the task using questions, instructions, prompts, and controls specific to each step in the task. TaskGuides are used for rarely-performed, and otherwise complex, tasks, such as printer and queue configuration, installing software, and so forth.

Further information on the use and requirements for TaskGuides may be found in the User Assistance section below.

6.2.4.4 Generic Dialogs

Generic dialogs are used to represent objects where notebook or TaskGuide functions are not necessary.

6.2.5 User Interface Elements

The user interface elements of the Web-Based System Manager container are described in the following sections:

Note: The figures are provided as an example to illustrate the user interface elements. The rendering of details of the title bar, menu bar, and so on, vary depending upon the client system on which the applet is running (Figure 15).

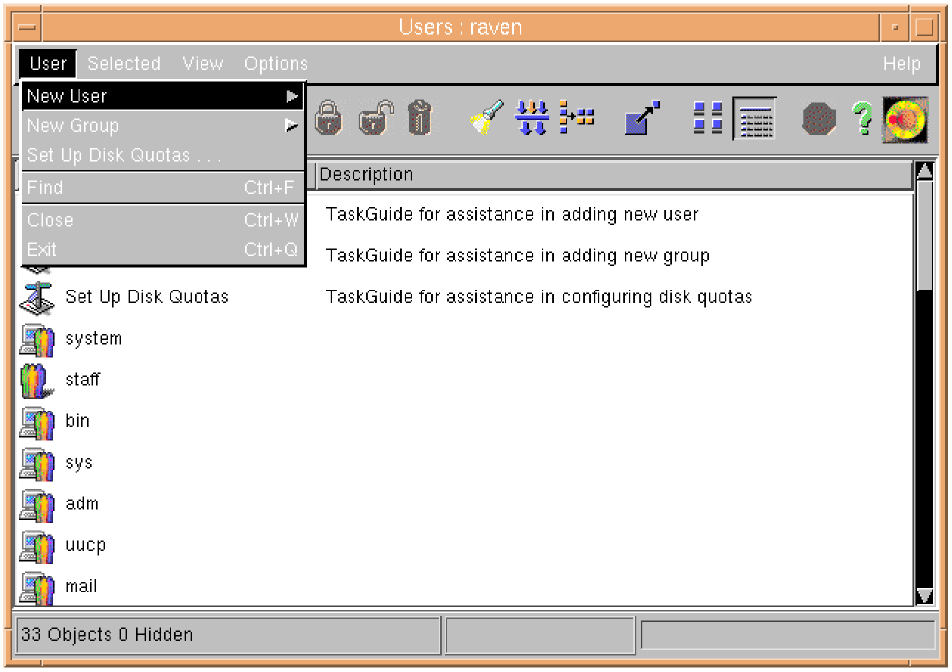


Figure 15. Web-Based System Manager User Menu

The menu bar may be shown or hidden at the user's option. A hidden menu bar may be retrieved through a pop-up menu on the main view area background.

The following menu items are meant to reflect actions that are common across most Web-Based System Managers containers. Additional choices are included for functions that have specific object types.

6.2.5.1 Menu

The object menu contains actions that globally apply to the current container. In each container type, the object menu is titled with the name of the type of object included in the menu. For example, in the users container the object menu is titled User, in the printer's container, the menu is titled Printer, and so on. The basic object menu choices are:

New Create a new instance of the object type contained in current window. This action is equivalent to opening the default template object for the container.

Switch Administrator (applet and remote mode only)
Switch to another user. A dialog box is opened for logging in as

another user. Following authentication, the new user's administrative rights are used for actions on the contents of the current container.

Find Opens a search dialog for locating objects. The result of the find action is displayed by providing selection emphasis for visible objects matched in the view area and scrolling the view area to the first object found.

The find dialog also includes a list area for displaying the objects found (because some may not be visible since they are nested in subcontainers) and provide a method of saving the results in a new subcontainer.

Close Close the container window.

Exit Exit Web-Based System Manager. An exit menu choice is present on all Web-Based System Manager containers. When Exit is selected from secondary containers, it generates a confirmation dialog, otherwise, Web-Based System Manager exits.

6.2.5.2 Selected Menu

The selected menu contains actions that are applied against one or more selected objects in the view area. The contents of the Selected menu will differ depending on the type of object container. The selected menu lists only these actions that apply to an object or the set of objects selected. Actions for an object that are temporarily unavailable (for example, *start* when the object has already been started) are dimmed.

Note: The pop-up menu for an object in the view area is roughly equivalent to the selected menu. Additionally, the enabled/disabled menu choices for a pop-up menu on a given object will be equivalent to the enabled/disabled menu choices on the selected menu when the same object has selection emphasis.

Figure 16 shows an example of the Web-Based System Manager Selected menu.

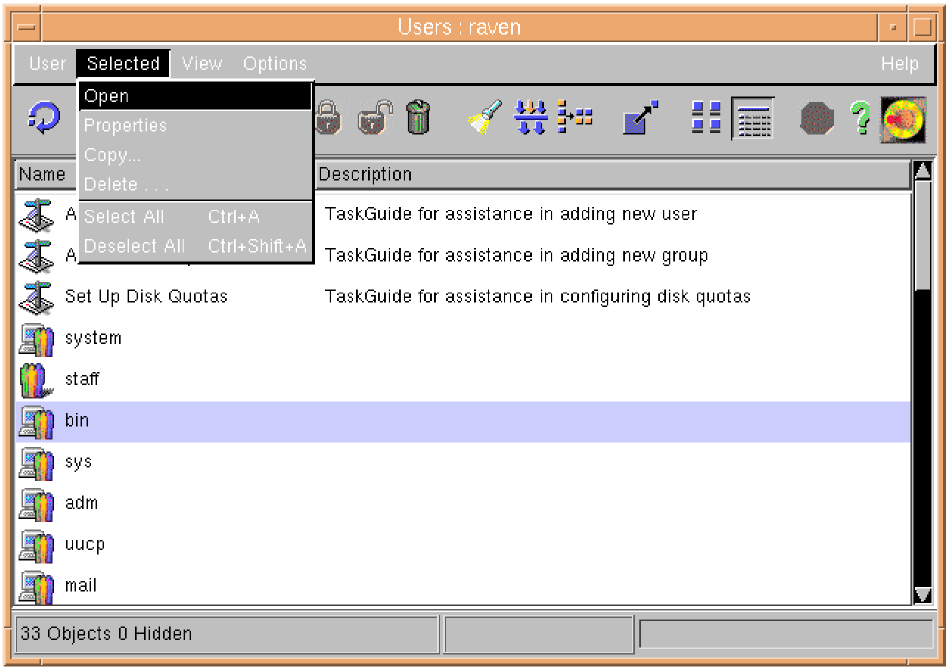


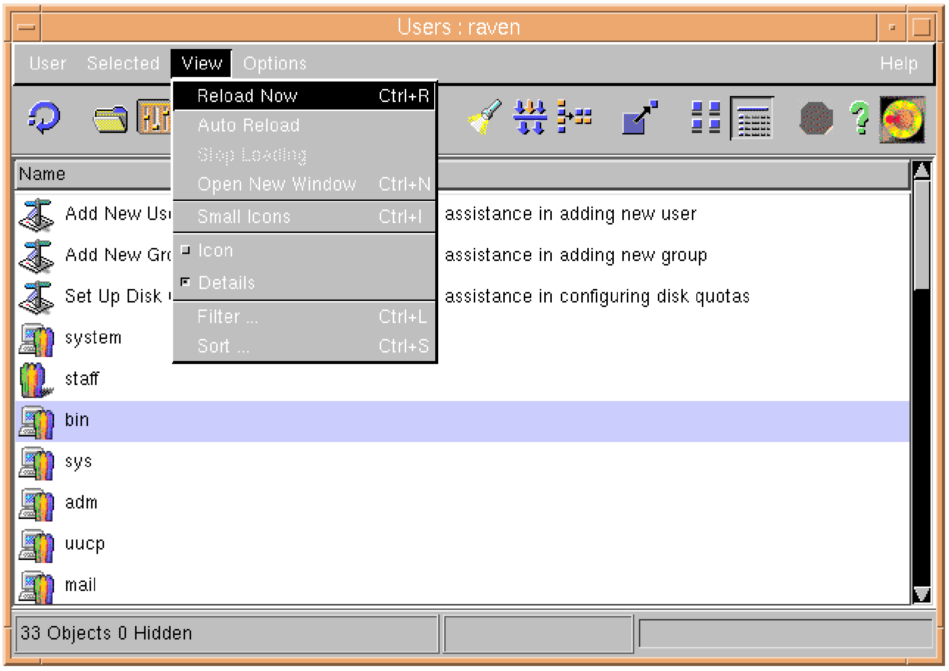
Figure 16. Web-Based System Manager Selected Menu

The basic Selected menu choices are:

Open	Opens a selected container or TaskGuide.
Properties	Opens a properties notebook dialog for the selected object.
Delete	Deletes the selected objects.
Select all	Selects all of the objects in the container.
Deselect all	Deselects all of the objects in the container.

6.2.5.3 View Menu

The view menu contains options that change the way objects are presented in the view area. The view menu is shown in Figure 17.



Add New Us… assistance in adding new user
Add New Gr… assistance in adding new group
Set Up Disk … assistance in configuring disk quotas
system
staff
bin
sys
adm
uucp
mail

33 Objects 0 Hidden

Figure 17. Web-Based System Manager View Menu

The View menu choices are:

Reload Now
Updates the view area with the latest data from the target system. This is analogous to the Netscape *Reload* function.

Stop Loading
Halts an update of the view.

Open New Window
Opens another instance of the current container window. This is equivalent to the Netscape *New Web Browser* action and the CDE file manager *Open New View* action.

Large/Small Icons
Selects either small or large icons for the display.

Icon View
Displays icons in a grid arrangement.

Tree
Changes the presentation of icons to tree view.

Details
Changes the presentation to a tabular view that displays small icons in the first column, object names in the second column, and other relevant properties in one or more additional columns (for example, object description, status, and so on). The exact information displayed in the details view will vary depending upon the application.

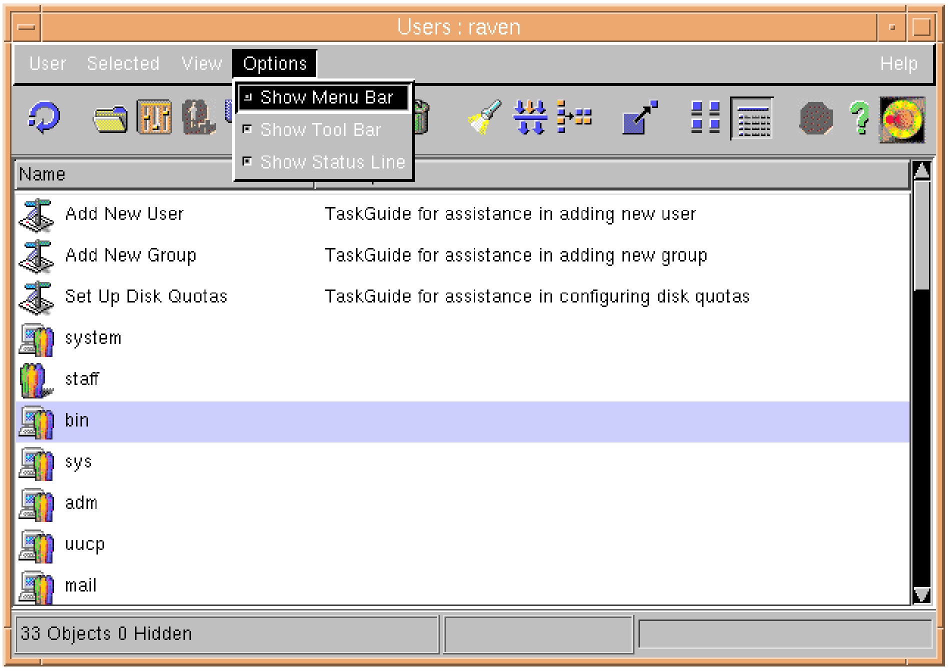
Tree Details
Changes the presentation of icons to a tree that also lists properties of each node.

Filter
Opens a dialog box for filtering objects based on user-specified criteria. (Filter is available only in icon and detail views).

Sort
Open a dialog box for sorting the objects based on user-specified criteria. (Filter is available only in icon and detail views).

6.2.5.4 Options Menu

The Options menu contains choices that specify the inclusion, or exclusion, of main user interface elements in the primary window, such as tool bar, status line, and so on. These menu items are selected by a check box for each menu choice, as shown in Figure 18.



Name	
Add New User	TaskGuide for assistance in adding new user
Add New Group	TaskGuide for assistance in adding new group
Set Up Disk Quotas	TaskGuide for assistance in configuring disk quotas
system	
staff	
bin	
sys	
adm	
uucp	
mail	

33 Objects 0 Hidden

Figure 18. Web-Based System Manager Options Menu

The Options menu choices are:

Show Menu Bar (check box)
Default position is checked. Unchecking removes the menu bar from the window. The menu bar is restored through a pop-up menu choice on the view area background.

Show Tool Bar (check box)
Default position is checked. Unchecking removes the tool bar from the window and subsequent subcontainers windows.

Show Status Line (check box)
Default position is checked. Unchecking removes the status line from the container and subsequent containers.

6.2.5.5 Help Menu

See Section 6.2.7.2, “Container Help Menu Contents” on page 214.

6.2.5.6 Pop-Up (Context) Menus

Pop-up menus are available for each object in a view area. When the cursor is positioned over an object, the Selected menu for that object is presented in a pop-up menu. When the cursor is over the main view area background, the pop-up menu contains the Options menu contents.

When a group of dissimilar objects is selected, the pop-up menu for the collection reflects only the actions that are applicable to all of the collection.

6.2.5.7 Dynamic Status Icon

A dynamic status icon is used to indicate the status of:

- Communications
- Processing on the target system

6.2.5.8 Tool Bar

Frequently used actions are represented on a tool bar. The tool bar can be displayed, or hidden, at the user's option.

The contents of the tool bar consists of some icons common to all containers (for example, Reload Now, Stop Loading, View Type) and other icons unique to specific container types. For example, the Users container includes a *Change Password* icon.

6.2.5.9 Main View Area

The main view area displays the objects and containers within the current container.

6.2.5.10 Command Buttons

The following command buttons are included on the background panel of the dialog. They are:

OK	Applies all of the parameters specified on each of the tab pages visited and closes the dialog box.
Apply	Applies all of the parameters specified on each of the tab pages visited and leaves the dialog box open.
Cancel	Closes the dialog box without applying any parameters.
Help	Launches a help window for the tab page currently visible.

6.2.5.11 Status Line

The status line is used to display current status information about the container. The status line may be shown, or hidden, at the user's option. Examples of information displayed in the status line are: number of objects shown in the view area, number of objects hidden, and loading status.

6.2.5.12 Container Views

Container views present a variety of representations of a group of objects that can be altered according to user needs or preferences. Many containers are able to present more than one view. Because different view types may be more or less appropriate for different object types, there is no one default view for all object containers.

Examples of standard views are listed in the following:

Icon Icon view arranges icons for managed objects in a grid. This view is useful for displaying a large number of objects in a small area (Figure 19).

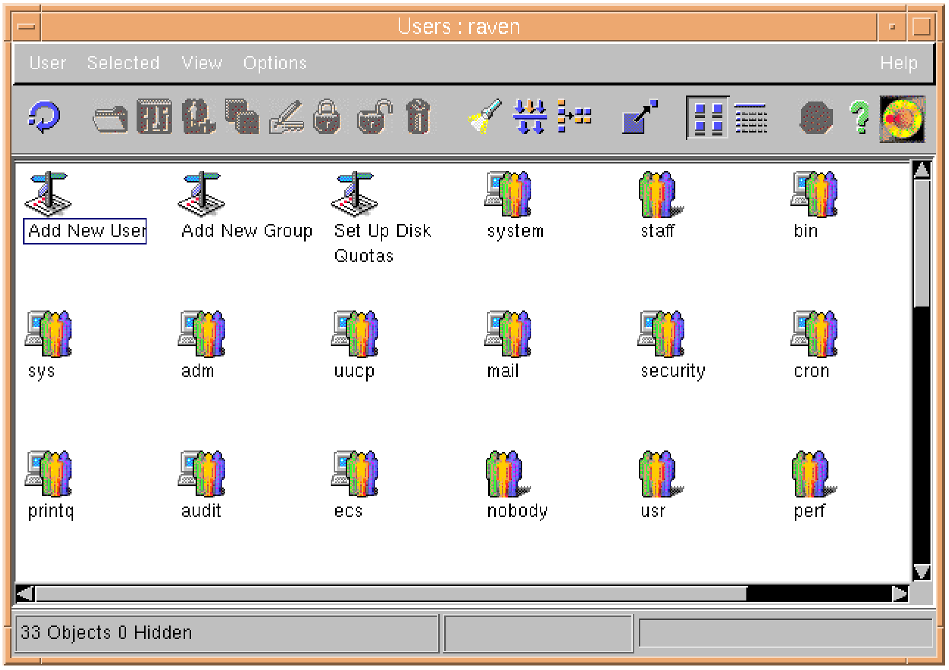
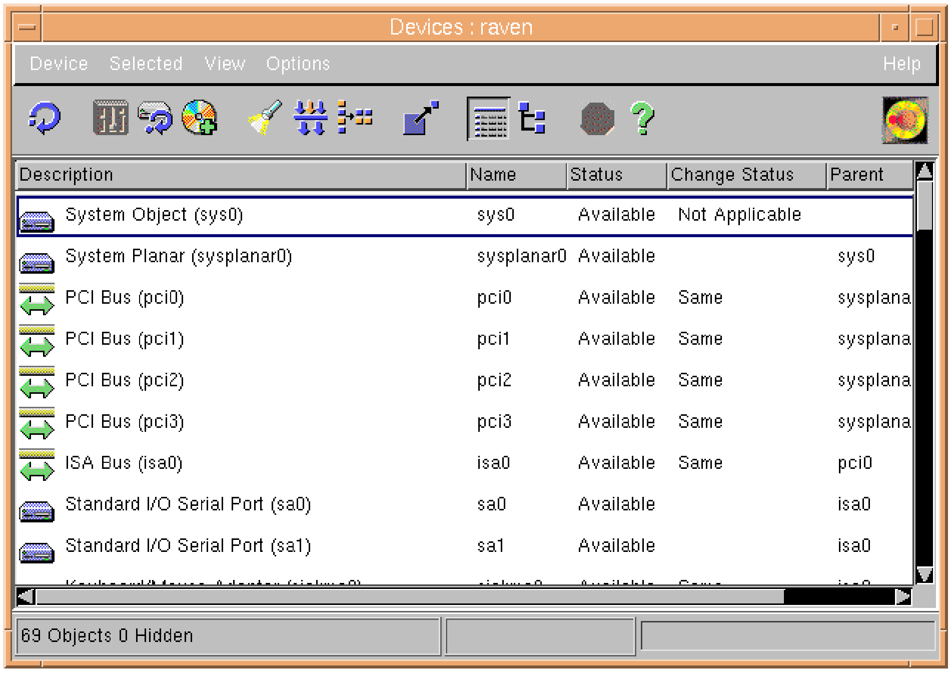


Figure 19. Web-Based System Manager Icon View

Details The icons in the view are displayed in a grid or table with the object icons, or names, in the columns on the far-left and additional property information in the remaining columns. See Figure 20 for an example of the details view for system devices.

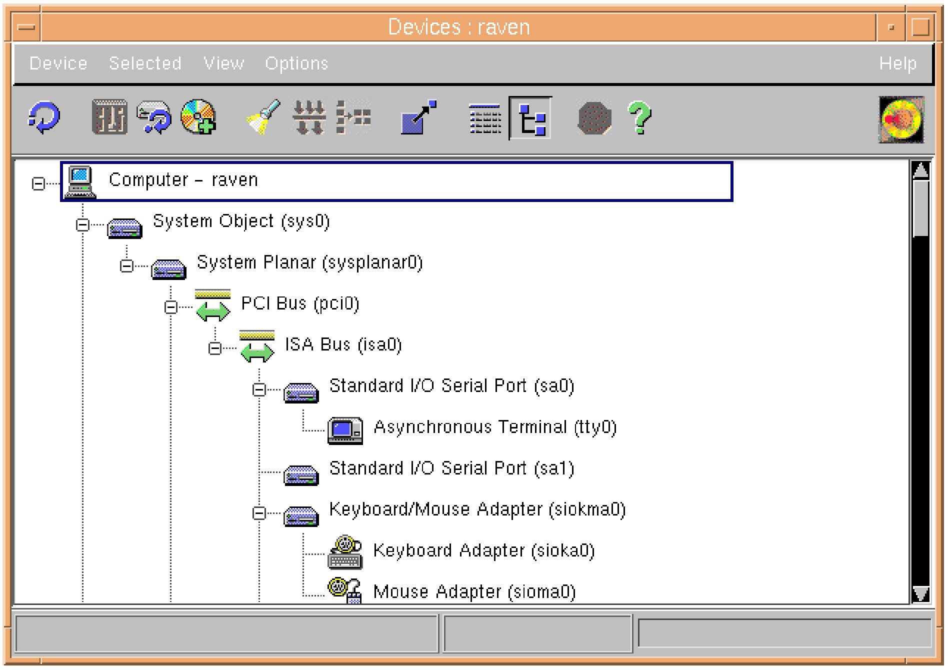


Description	Name	Status	Change Status	Parent
System Object (sys0)	sys0	Available	Not Applicable	
System Planar (sysplanar0)	sysplanar0	Available		sys0
PCI Bus (pci0)	pci0	Available	Same	sysplana
PCI Bus (pci1)	pci1	Available	Same	sysplana
PCI Bus (pci2)	pci2	Available	Same	sysplana
PCI Bus (pci3)	pci3	Available	Same	sysplana
ISA Bus (isa0)	isa0	Available	Same	pci0
Standard I/O Serial Port (sa0)	sa0	Available		isa0
Standard I/O Serial Port (sa1)	sa1	Available		isa0

69 Objects 0 Hidden

Figure 20. Web-Based System Manager Details View

Tree A tree view displays a hierarchical relationship with parent and child nodes. The tree view is useful for displaying users and groups, printers and queues, bundles and file sets, and devices. See Figure 21 for an example of a tree view for system devices.



Computer – raven
System Object (sys0)
System Planar (sysplanar0)
PCI Bus (pci0)
ISA Bus (isa0)
Standard I/O Serial Port (sa0)
Asynchronous Terminal (tty0)
Standard I/O Serial Port (sa1)
Keyboard/Mouse Adapter (siokma0)
Keyboard Adapter (sioka0)
Mouse Adapter (sioma0)

Figure 21. Web-Based System Manager Tree View

Note: The Web-Based System Manager UI architecture includes single-rooted and multiple-rooted tree views and trees of an arbitrary number of levels.

6.2.6 Message Boxes

Where it is necessary to alert the user to various conditions, a message box is used for the following purposes:

- Informational messages
- Warning messages
- Critical conditions
- Confirmation prompts

6.2.7 User Assistance

User assistance includes online information designed to aid users in performing unfamiliar tasks. Web-Based System Manager user assistance includes: online books, help, TaskGuides, hover help, and context help. These are described below.

6.2.7.1 Help

General help for each Web-Based System Manager application is provided through a container-level help for that application. The general help also contains an overview section common to all Web-Based System Manager applications.

All help text is written in HTML and is accessed through the Web browser on the client system.

6.2.7.2 Container Help Menu Contents

Container help menu choices are provided for the following:

- Contents (contents of extended or reference help)
- What's this? (places the application in context-sensitive help mode)
- Search for help on topic (Web page for accessing the help search engine)
- How to use Help
- About Web-Based System Manager (product information)

6.2.7.3 Context Sensitive Helps

Context-sensitive help is provided in a child window that pops up above the user interface element for which help is requested. It is available when the user:

- Selects **What’s this?** from the application help menu. This enters context-sensitive help mode.
- Clicks on a tool bar icon containing a question mark to get help on the contents of a window.
- Presses the **help** button on a dialog.

Context-sensitive help is available for:

- Fields with a dialog
- Objects in a view area

6.2.7.4 Hover Help

Web-Based System Manager displays hover help when the user pauses the mouse pointer over a tool bar icon.

6.2.7.5 Online Books

Web-Based System Manager users have access to the complete AIX online documentation library through hypertext links in extended help.

6.2.8 Navigation

Although the usual method of navigation through Web-Based System Manager is by use of a mouse, it also possible through the keyboard.

6.2.8.1 Keyboard Navigation

To meet the needs of a wide range of users of different abilities and skills, Web-Based System Manager supports keyboard navigation. Specific key assignments for keyboard navigation are similar to Windows and Netscape Navigator.

The following keyboard navigation methods are supported:

Focus Traversal

Tab and Shift-Tab are used to move forward and backward among controls.

Menu Shortcuts

Short cuts (or accelerators) are keyboard equivalents for menu commands that are executed by key combinations (such as CTRL-F for Find).

6.2.8.2 Mouse Model

For a three-button mouse, the mouse button functions are:

- Button 1 - select, drag, activate.
- Button 3 - context (pop-up) menu

For a two button mouse, the mouse button functions are:

- Button 1 - select, drag, activate
- Button 2 - context menu

A single-click of button 1 is used for selecting icons and activating button controls.

A double-click of button 1 activates view area icons with their default behaviors. For container objects, the default behavior is to open-contents-view. For simple objects (that is, without contents), the default behavior is open-properties-dialog. For objects that have both properties and contents (for example, UNIX groups), the double-click action is to open-contents-view. Pop-up and Selected menu choices (*Open*=contents, *Properties*=properties dialog) for each action are provided.

6.2.9 Selection and Multiple Selection

A single selection defines to which object the actions in the selected menu, or pop-up menu, apply. Specific actions are enabled, or disabled, depending upon the type of object selected. A single-click of button 1 is used to select an object.

A multiple selection is enabled for various types of objects and actions. Specific actions in the selected and pop-up menu will be enabled or disabled depending upon whether or not multiple selection is allowed for the collection of objects selected. The menu choices enabled are the intersection of the enabled states of the objects in the selected collection.

Most standard multiple selection interaction techniques are supported, including range selection, use of Ctrl-Select to modify a selection, and use of shift-select to select a contiguous range of objects.

6.3 Web-Based System Manager Enhancements (4.3.1)

Web-Based System Manager was introduced with AIX version 4.3.0 as a technology evaluation release. It did not provide all of the function required for system management but was intended to demonstrate to customers the direction of system management products.

The version of Web-Based System Manager shipped with AIX version 4.3.1 contained significant performance enhancements over the previous release. The improvements were mostly due to improvements in the underlying Java run time system.

6.4 Web-Based System Manager Enhancements (4.3.2)

The following sections describe the enhancements to Web-Based System Manager that were introduced by AIX version 4.3.2.

6.4.1 Security Enhancements

AIX 4.3.2 has enhanced the function of Web-Based System Manger to allow remote administration sessions to be carried out using the Secure Socket Layer (SSL) protocol. This allows all data transmitted on the network between the Web-Based System Manager client and the machine being managed to be encrypted and, therefore, prevent unauthorized systems from viewing the data.

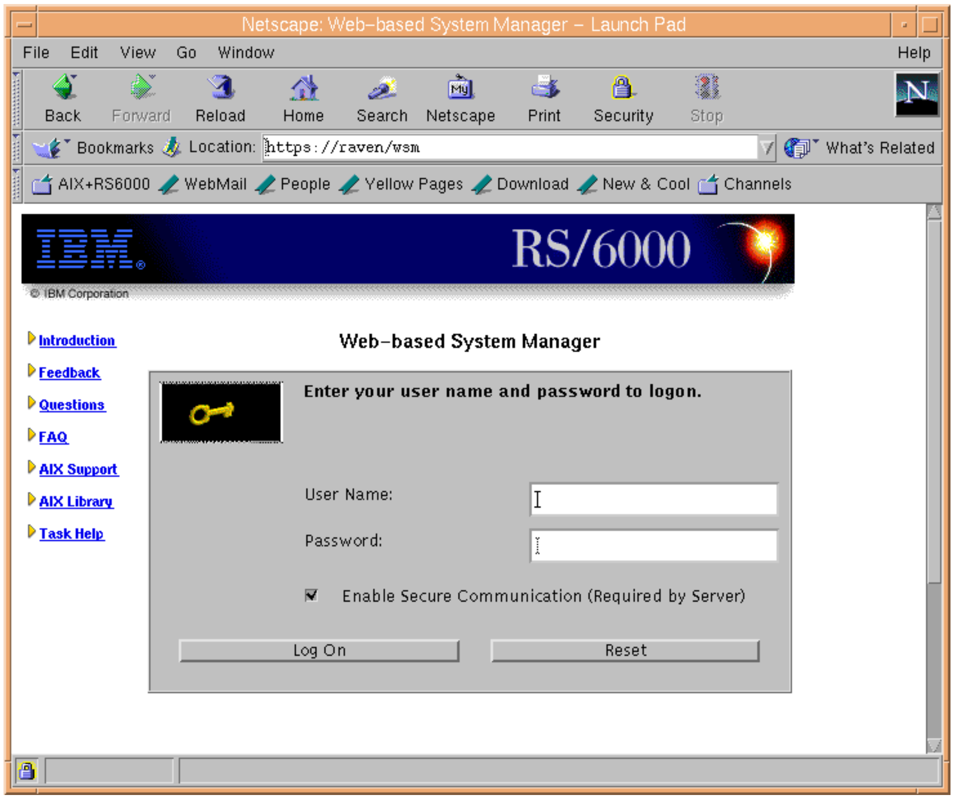
The software required to implement this function is included on the AIX Bonus Pack that ships with AIX 4.3.2. The required package is sysmgt.websm.security. Users in the United States and Canada can additionally install the package sysmgt.websm.security-us, which provides stronger encryption facilities.

The configuration process for using the SSL protocol involves generating a public and private key for each machine to be managed. The certificates can be obtained from an external Certificate Authority or generated on a designated Web-Based System Manager server for use in a private network. The keys are installed on the machine being managed and, additionally, on any AIX machines that will use the Web-Based System Manager client to manage the machine in client/server mode. In this scenario, all communication between the client and server takes place using the SSL protocol.

In applet mode, where the Web-Based System Manager client is run in a browser, the client is required to download the Web-Based System Manager servers public key in order to verify the applet files that are being downloaded. For maximum security, the client should connect to the server using the HTTPS protocol.

The encryption facilities provided work in conjunction with a Web server that uses SSL to support the HTTPS protocol. The Lotus Domino Go Web server, that is also provided on the Bonus Pack can be configured to accept requests using the HTTPS protocol, either in addition to, or instead of, the HTTP protocol. Similarly, the Web-Based System Manager server running on the managed machine can be configured to respond to HTTPS requests either in addition to, or instead, of HTTP requests.

A client session with a server that has been configured with the optional security is shown in Figure 22.


Netscape: Web–based System Manager – Launch Pad
File Edit View Go Window Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: https://raven/wsm What's Related
AIX+RS6000 WebMail People Yellow Pages Download New & Cool Channels
IBM RS/6000
© IBM Corporation
Introduction
Feedback
Questions
FAQ
AIX Support
AIX Library
Task Help
Log On
Reset

Web–based System Manager

Enter your user name and password to logon.

User Name:

Password:

[✓] Enable Secure Communication (Required by Server)

Figure 22. Example of Secure Mode Connection Using HTTPS

The only visible differences when using the secure version are:

- The URL of the initial Web-Based System Manager login page specifies the HTTPS protocol.
- The browser indicates that the Web page being viewed has been obtained using a secure connection. The Netscape Navigator browser shipped on the Bonus Pack indicates this with a locked padlock icon in the lower left corner of the window.
- Web-Based System Manger child windows have the message Secure Connection displayed in the status bar at the base of the window. See Figure 23.

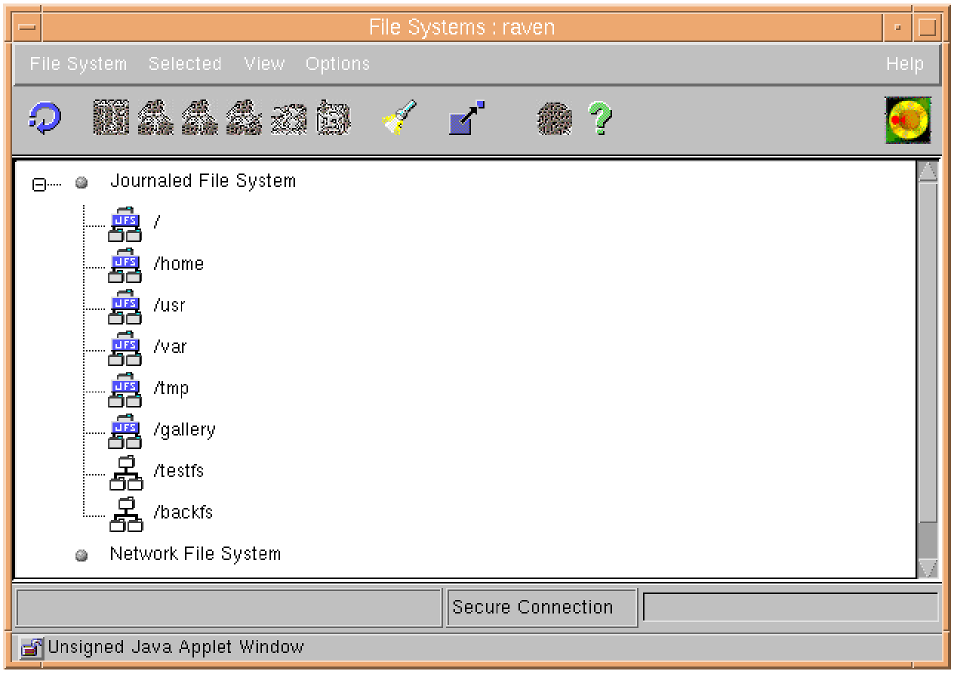


Figure 23. Example of Container Window In Secure Mode

6.4.2 Diagnostics Enhancements

Web-Based System Manager has been enhanced to allow diagnostic and service aid functions to be carried out on devices that support these actions. The menu presented depends on the capabilities of the selected device. For example, it is possible to perform format or certify operations on certain models of disk drives, as shown in Figure 24.

It is also possible to perform Error Log Analysis when running diagnostics on the selected device. The AIX error log can be searched for errors logged against the selected device for errors between 1 and 60 days old.

If a device is marked with a warning triangle (containing an explanation point), the Run Diagnostics selection can be used to determine what is wrong with the device, or if it has just been removed from the system.

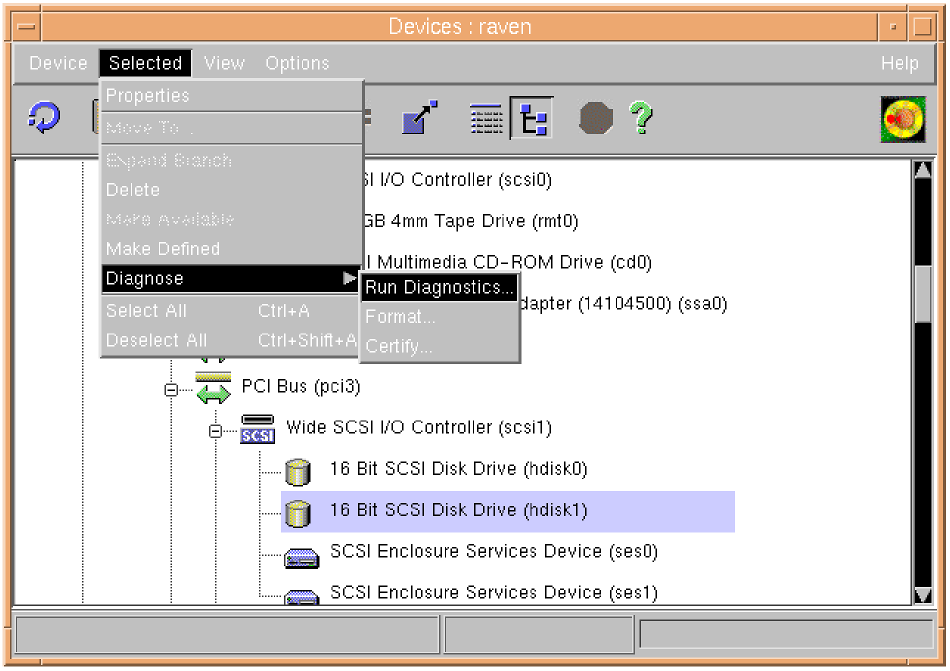


Figure 24. Example of Diagnostics Menu

Using the previous selections, the menu shown in Figure 25 is presented. Here, you may select how the diagnostics are run.

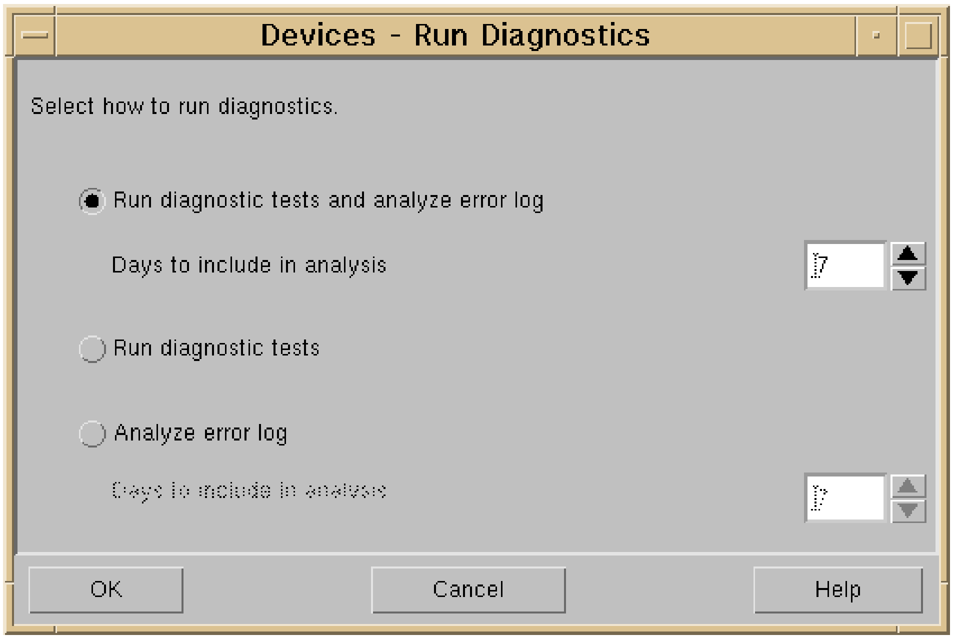
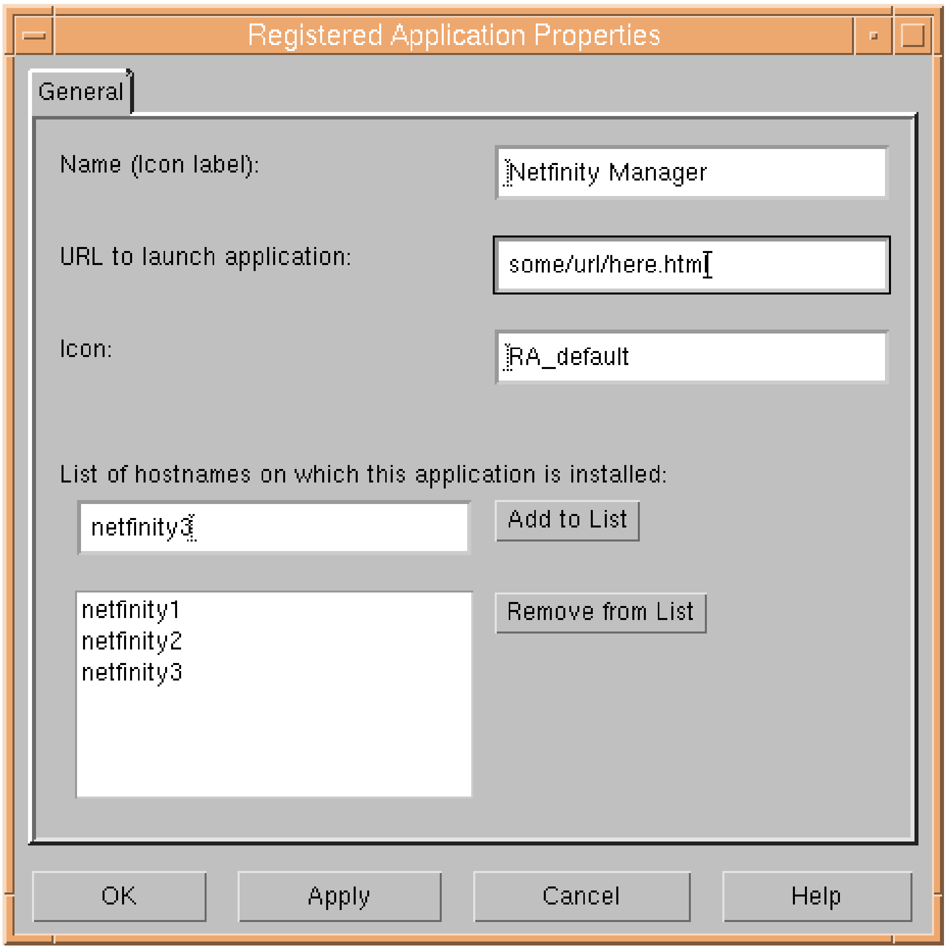

Devices - Run Diagnostics
Select how to run diagnostics.
Run diagnostic tests and analyze error log
Days to include in analysis
7
Run diagnostic tests
Analyze error log
Days to include in analysis
7
OK
Cancel
Help

Figure 25. Example of ELA Day Selection Menu

6.4.3 Registered Applications

This function allows a system administrator to register remote applications with Web-Based System Manager. It only supports an application that can be accessed using a URL. For example, it allows a system administrator to register the Netfinity Manager application with Web-Based System Manager. The dialog screen, shown in Figure 26, prompts the user to enter the URL to start the application and shows a list of machines that have the application installed.



Name (Icon label): Netfinity Manager

URL to launch application: some/url/here.html

Icon: RA_default

List of hostnames on which this application is installed:

netfinity3

netfinity1
netfinity2
netfinity3

Figure 26. Registered Applications Dialog Box

The registered application then appears on the Registered Applications container as an icon, shown in Figure 27.

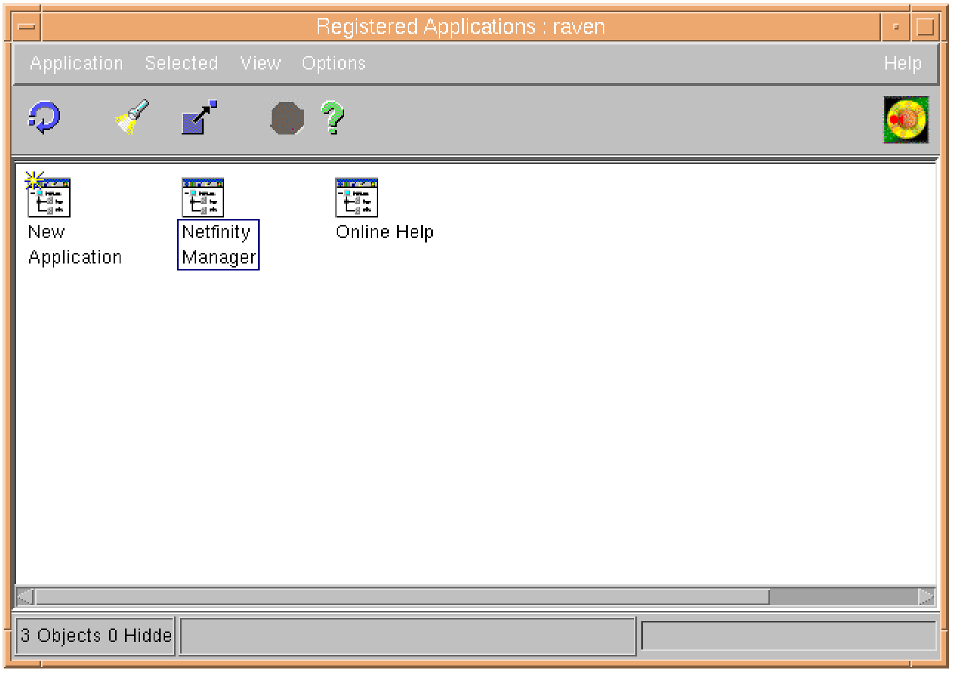
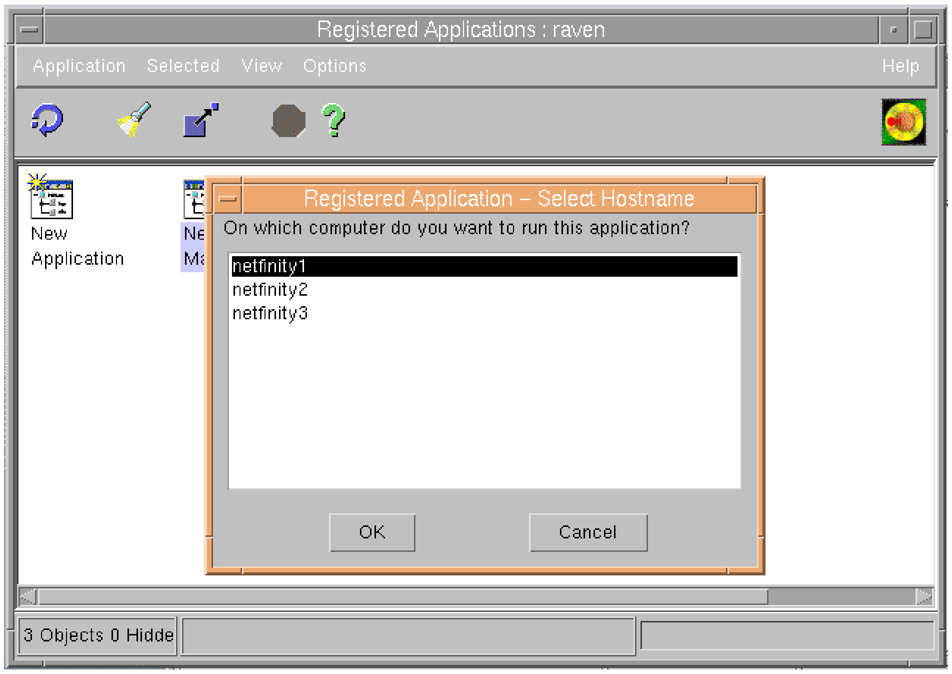


Figure 27. Registered Applications Container

When the icon is opened, the user is prompted to select the required target machine if the application is registered with multiple machines. This is shown in Figure 28. Web-Based System Manager then starts the Netscape Web browser with an initial URL of the registered application on the target machine.


Registered Applications : raven
Application Selected View Options Help
New Application
Registered Application – Select Hostname
On which computer do you want to run this application?

netfinity1
netfinity2
netfinity3

OK
Cancel
3 Objects 0 Hidde

Figure 28. Registered Application Host Selection Dialog

6.5 Web-Based System Manager Enhancements (4.3.3)

Following is the list of new enhancements on Web-Based System Manager presented with AIX Version 4.3.3. The enhancements include both enhancements to existing applications and new applications, namely:

- Volumes Application enhancements (5.9, “Mirroring and Striping Support (4.3.3)” on page 188 for striping and mirroring, 6.5.1, “Volumes Application Enhancements” on page 225 for large-disk VG and big VG support)
- File Systems Application enhancements (6.5.3, “File Systems Application Enhancements” on page 226)
- Workload Management (7.9, “Web-Based System Manager Interface” on page 356)
- NIS+ (6.5.2, “NIS+” on page 226)
- IPSec (8.3.3, “Web-Based System Manager Panel Enhancements” on page 400)

- Quality of Service (8.24.4.1, “Web-Based System Manager Integration” on page 482)
- SMB Server Release 2 (8.28.2, “AIX Fast Connect Release 2 New Function” on page 490)

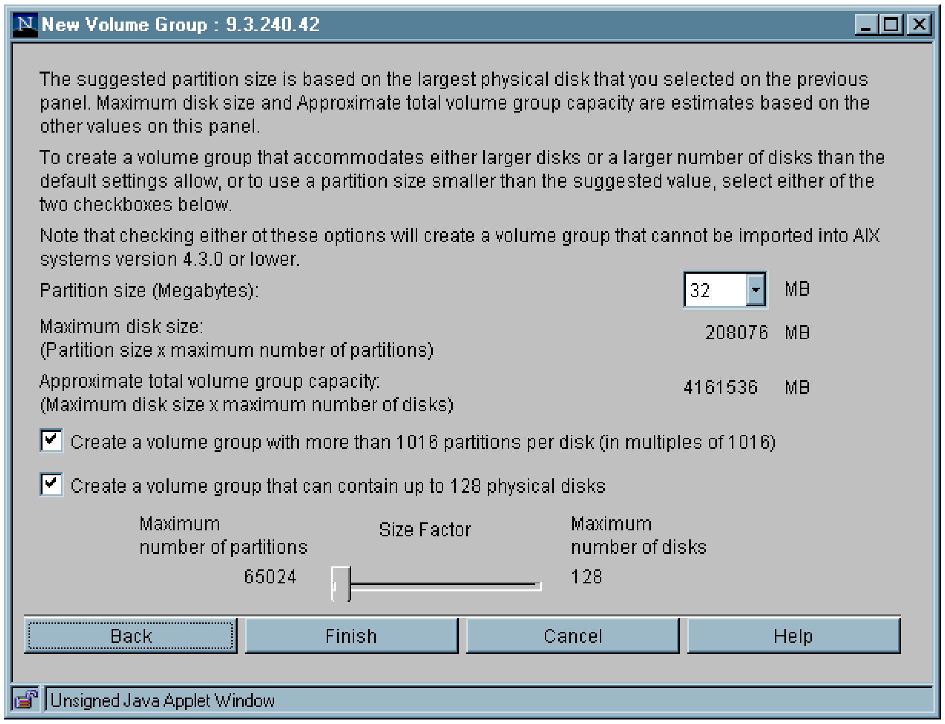
Also a lot of modifications have been done to existing applications. Most of them are bug fixes and user interface changes for better look and feel.

See each section referenced for descriptions and GUI images of the enhancements.

6.5.1 Volumes Application Enhancements

Volumes Application supports creating large-disk VG (VG that support more than 1016 PPs per PV) and big VG (VG that has more than 32 PVs).

Figure 29 shows the configuration panel which is launched from **Volume->Add Volume Group->Advanced Method** menu.


New Volume Group : 9.3.240.42

The suggested partition size is based on the largest physical disk that you selected on the previous panel. Maximum disk size and Approximate total volume group capacity are estimates based on the other values on this panel.

To create a volume group that accommodates either larger disks or a larger number of disks than the default settings allow, or to use a partition size smaller than the suggested value, select either of the two checkboxes below.

Note that checking either ot these options will create a volume group that cannot be imported into AIX systems version 4.3.0 or lower.

Partition size (Megabytes): 32 MB

Maximum disk size:
(Partition size x maximum number of partitions) 208076 MB

Approximate total volume group capacity:
(Maximum disk size x maximum number of disks) 4161536 MB

[✓] Create a volume group with more than 1016 partitions per disk (in multiples of 1016)

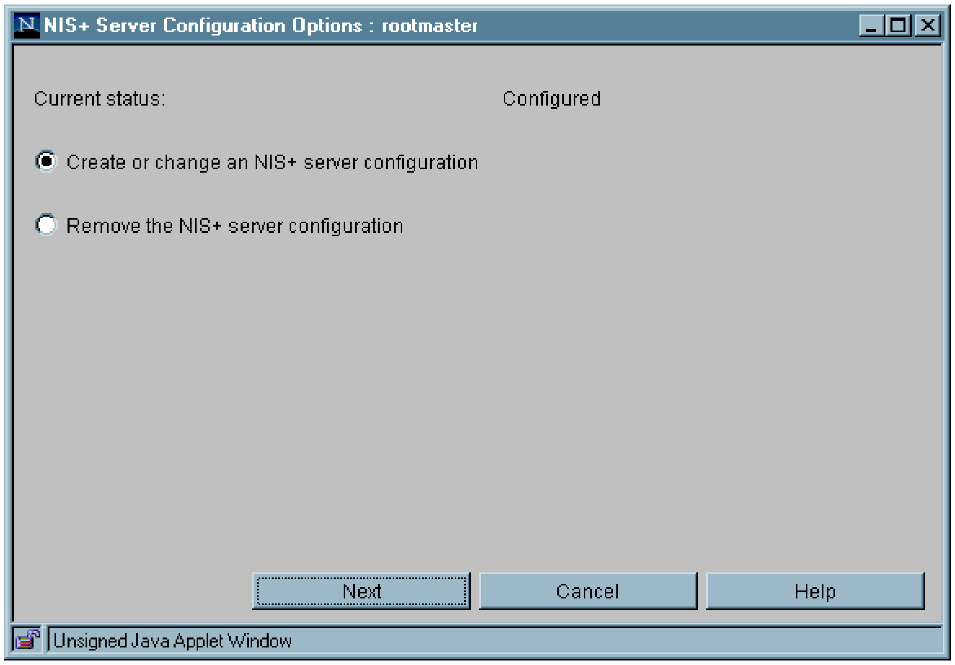
[✓] Create a volume group that can contain up to 128 physical disks

Maximum number of partitions: 65024
Size Factor
Maximum number of disks: 128

Figure 29. Volume Application BIG VG and Large-Disk VG Support

6.5.2 NIS+

AIX 4.3.3 provides NIS+ configuration panels on Web-Based System Manager. You can configure, change delete the NIS+ configuration and populate NIS+ tables using Web-Based System Manager panels. The panel shown in Figure 30 can be launched by double-clicking NIS+ Server icon on **Network** panel.


NIS+ Server Configuration Options : rootmaster

Current status: Configured

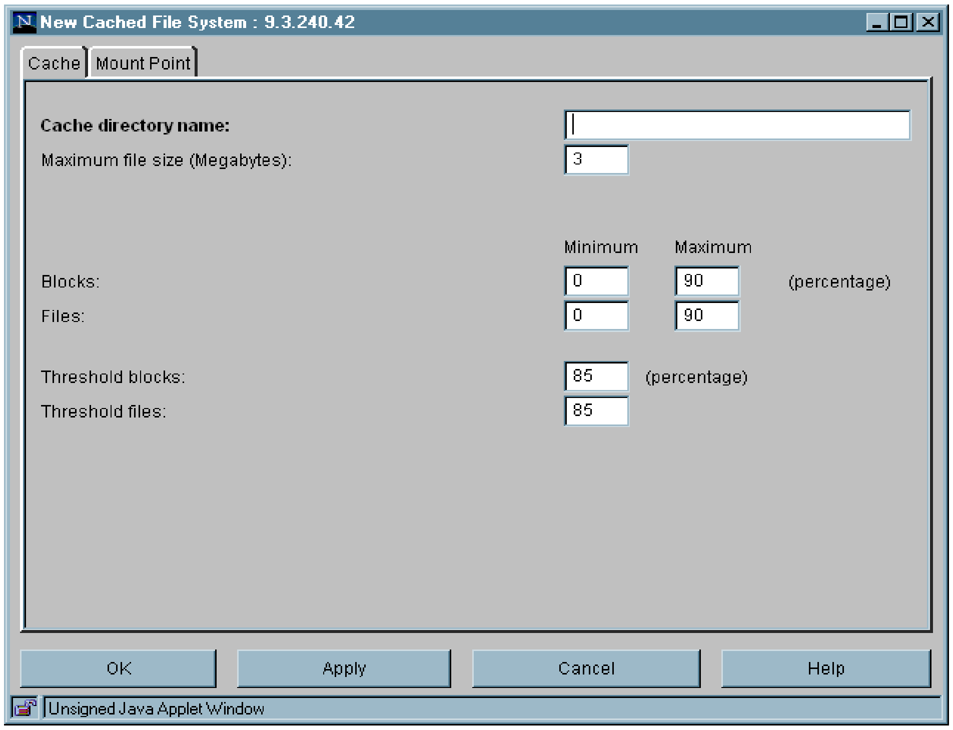
[✓ Create or change an NIS+ server configuration]

Remove the NIS+ server configuration

Figure 30. NIS+ Server Configuration Panel on Web-Based System Manager

6.5.3 File Systems Application Enhancements

The Web-Based System Manager file systems application now supports cache file system. You can create, change properties of, delete a cache file system through Web-Based System Manager file systems application. The Cached File Systems icons appears below the CD-ROM File Systems icon and the following panel can be launched by selecting **File System->New File System->Create Cached File System** menu on **File Systems** panel (Figure 31).


New Cached File System : 9.3.240.42
Cache
Mount Point
OK
Apply
Cancel
Help
Unsigned Java Applet Window

Cache directory name:			
Maximum file size (Megabytes):	3		
	Minimum	Maximum	
Blocks:	0	90	(percentage)
Files:	0	90	
Threshold blocks:	85	(percentage)	
Threshold files:	85		

Figure 31. New Cached File System Web-Based System Manager Panel

6.6 Daylight Savings Time

Before AIX Version 4.3, daylight savings time could be selected through SMIT, but the date of change for this characteristic was restricted to USA standard. Once the daylight savings YES or NO question had been answered, a list of available time zones was presented, but there was no option within SMIT for specifying the start and end dates for daylight savings.

By default, the daylight saving time starts on the first Sunday of April at 2:00 am and stops on the last Sunday of October at 2:00 am. AIX Version 4.3 now has the capability of overriding these settings by specifying the start time and stop time in the TZ environment variable.

Additional fields have been added to the SMIT time setting menu to set the TZ variable. A back-end script performs the actual setting of the TZ variable.

6.7 Login Performance

The original design of the UNIX/AIX login system dates back to the early days of UNIX when the number of users to be catered for was relatively small. As such, the login process was perfectly adequate. With the commercial acceptance of UNIX, however, the number of users per system has grown dramatically with tens of thousands of users now being seen on some servers. This increase in the number of users has highlighted some of the deficiencies in the original design of the login process that are now beginning to affect system performance.

A problem exists, which once a user has entered their name and password, the system must then search through the /etc/passwd and /etc/security/passwd files trying to find a match for that user and, if successful, must also update a number of other files. All the files are searched sequentially, and the time consumed can be substantial if there are a large number of records to search through. In extreme cases, if a user’s entry is near the end of the files, it is possible for the login attempt to time out before completion. Also, the amount of CPU time being consumed by the login process is a cause for concern. Login CPU usage has been recorded as high as 47 percent on some systems.

The three major bottlenecks that have been identified are:

- Reading the /etc/passwd file
- Reading the /etc/security/passwd file
- Updating the /etc/security/lastlog file

A limited solution was used in previous versions of AIX to address some of these issues by creating a hashed version of the /etc/passwd file. The `mkpasswd` command took the /etc/passwd file as input and created the /etc/passwd.dir and /etc/passwd.pag files. Then during login, if these hashed files existed, they were read instead of the ASCII file. This partial solution provided some relief, but there were still other areas that could also be improved.

A further improvement has been introduced in AIX Version 4.3 that provides indexed access to the data files instead of sequential reads. Indexes are created using the user name and user ID as keys with offsets into the data file. The data files remain as ASCII files, but the design allows for them to be upgraded to binary files at a later date if this is found to be necessary.

The /etc/passwd and /etc/security/passwd files have been indexed, and the /etc/security/lastlog file has been indexed with a corresponding change in the way that this file is processed.

6.7.1 Indexing of the /etc/passwd File

The actual /etc/passwd file has not been changed. However, two new files have been created, /etc/passwd.nm.idx and /etc/passwd.id.idx, through the `mkpasswd` command. These files are indexes created using the username (string) and the user ID (number) as keys. A record in the index file contains the key, offset of the corresponding record in the data file (/etc/passwd), and the status of the record. A negative offset value implies the corresponding record is deleted.

A hook was also added at the point where the file is read to check for the existence of the index file. If the corresponding index exists, then the index read mechanism is called with the key as a parameter.

6.7.2 Indexing of the /etc/security/passwd File

The /etc/security/passwd file is a text file that contains one stanza for each user. It is searched one line at a time looking for the user name. Similar to the /etc/passwd file, the user name that is physically located at the top of the file is at an advantage compared to the user name at the bottom of the file. This file is also indexed by the `mkpasswd` command. The index is based on the username string as the key and provides an offset into the file where the stanza can be located. Once the stanza is located, it is then searched sequentially for the relevant information.

6.7.3 Indexing and Locking /etc/security/lastlog File

The lastlog file is a text stanza file similar to the /etc/security/passwd file that contains one stanza per user. It is accessed in a similar manner, sequentially looking for a username. However, unlike the /etc/passwd and /etc/security/passwd files, this file is accessed for update, which means that file-locking and crash-recovery must be taken into account. In the existing design, locking is done at a file-level, and the whole file is backed up, representing a major overhead. Note that there are no external commands or system and library calls to access this file. It is accessed internally by the tsmlogin() module to display lastlog information.

In AIX Version 4.3, the /etc/security/lastlog file remains a text file but has been changed in the following ways:

Indexed access

An index called /etc/security/lastlognm.id is built using the username string. This index provides the offset to a stanza. Once a stanza is located, the lines in it are processed sequentially as in AIX Version 4.2. The index is created by the `mkpasswd` command.

Fixed record length

Since a user stanza is updated upon login, if the file needed to be reorganized after every update (because of the variable length text fields), this would cause the record (or stanza) offsets for all stanzas after the changed stanza to be changed by a fixed delta. This would keep the data and index files synchronized but would be expensive in processing time. The extra processing was avoided by keeping some unused space in each variable length field. By padding the fields with spaces, they can shrink or grow within limits, and a record can be updated without having to rearrange the file and rebuild the index.

Record level lock

Locking is now done at the record-level instead of the file-level by using advisory locks for updates. This prevents having to read and write the entire file. There is no longer a need for the /etc/olastlog file, and it has therefore been removed.

6.7.4 mkpasswd Command

The index system is created when the `mkpasswd` command is executed. It deletes any existing outdated indexes (except for the lastlog index) and builds new indexes. New flags allow complete control of this enhancement.

6.8 LDAP Exploitation for User Management (4.3.3)

There has been an increasing demand for a facility where the data is centrally maintained on a server system and remotely accessed by one or more clients in real time. LDAP is evolving as a standard protocol across the industry and it is suitable for distributed security authentication because it provides a ready made client/server implementation.

In AIX 4.3.3 security routines are enabled to make LDAP calls to store and retrieve data in order to manage user and group information. System performance improves in environments with large number of users because of advantages inherent with searching through a database versus a text file, even if it has indexes.

Unlike other directory services like NIS and NIS+, all the data related to AIX users is stored into the LDAP database including, for example, login restriction and system usage limits.

Security issues has been taken in special account to avoid LDAP clients to gain non-authorized information on user data. The LDAP server does not provide the LDAP schema or content to LDAP client except those who authenticate themselves as being system administrator. There are two layers of protection:

1. Password Protection: a password is specified by the system administrator when the schema is configured and the LDAP APIs use this password to bind with the server. The password is distributed to clients which store it locally in clear text but in file located in a protected directory.
2. SSL Secure Port Protection: the password protection provides security but information may still be captured snooping the communication network. To provide further security, the LDAP can be configured to allow only SSL connections for AIX information.

6.8.1 The mksecldap Command

The `mksecldap` command provides both client and server configuration in order to use LDAP for security authentication and data management.

The syntax for server configuration is:

```
mksecldap      -s -a adminDN -p adminpasswd {-d ldapaixdn}\
               {-k ssl key file path -w ssl key password}
```

The syntax for client configuration is:

```
mksecldap      -c -h hostlist {-d ldapaixdn} {-u [ALL|userlist]} \
               -a adminDN -p adminpasswd \
               {-k ssl key file path -w ssl key password } \
               {-t maxtimeout} \
               {-C <cache size> } {-P <number of thread>}
```

The switches meaning is the following:

-c	Client configuration
-s	Server configuration
-a	Administrator’s user name in distinguished name format
-p	Administrator’s password
-d	Distinguished name of AIX security information subtree. If not specified, the default cn=aixsecdb is used

-k	Path to the SSL key file
-w	Password needed to access SSL key file
-h	List of the LDAP servers that provide AIX security data
-u	When used on a client, it updates local /etc/security/user file in order to define LDAP provided authentication. If an ALL keyword is provided, all AIX users will be authenticated using LDAP.
-t	Time out on LDAP request. If time out occurs, local files are used for authentication.
-C	Size of local LDAP data cache used by `secldapclntd`
-P	Number of threads created `secldapclntd`

On the server, the command configures both LDAP and DB2 database. In order to allow LDAP to provide both public access to non-AIX data and restricted access to AIX security data, a secondary LDAP server is started with a separate database instance. The primary LDAP server points to the secondary for AIX data. The LDAP database is populated with all the users defined on the server and their related information.

On the client, the command creates the /etc/security/ldap/ldap.cfg file that contains the needed data to connect to the LDAP servers and starts the `secldapclntd` daemon that receives all the AIX security command's requests and forwards them to the LDAP servers. The command also updates the /etc/security/user file to define which users must use LDAP authentication, when the -u switch is used.

6.8.2 Server Configuration

The server configuration is very simple. The basic information you have to provide is the name of the administrator, the password and, optionally, the LDAP directory name. For example, you may execute:

```
mksecldap -s -a cn=root -p rootpwd
```

If you want to use the SSL security, you have to install from the Bonus Pack the gskrf301.base fileset. Then you must create the SSL certificate (see Appendix A, "SSL File Creation for LDAP" on page 685) and then execute, for example:

```
mksecldap -s -a cn=root -p rootpwd -k /etc/keyfile -w sslpwd
```

The `mksecldap` command takes care of configuring both LDAP and DB2 without interfering with existing LDAP configuration or data. Since many DB2 operations are required, it may take several minutes to complete.

When command finishes, LDAP is ready to provide AIX user information. Initially, the database only contains the users and groups defined in the server machine.

The configuration of the server part of LDAP does not make the RS/6000 a client of LDAP database. In order to make the machine access the LDAP server, an explicit client configuration must be made.

Note

By default, the `mksecldap` command created the DB2 database in the /home file system. The command may fail due if there is not enough free space.

6.8.3 Client Configuration

On client side, the `mksecldap` command requires the names of the LDAP servers, the user, and password to access the database. Optionally you can use SSL for additional security providing the key file with the appropriate password. For example:

```
mksecldap -c -h LDAPsrv -a cn=root -p rootpwd
```

or, for SSL encryption:

```
mksecldap -c -h LDAPsrv -a cn=root -p rootpwd -k /etc/keyfile.kdb -w sslpwd
```

If you have not used the default directory name on the server, you need to provide also the directory’s distinguished name using the -d switch.

In the previous examples, no users have been added to the system, but LDAP access has been activated. You must configure the local machine to accept LDAP users.

User authentication using LDAP is made by defining in /etc/security/user the authentication method to be LDAP. This is made by changing the SYSTEM attribute of the default stanza, or of a specific user, to be equal to LDAP and providing a new registry named LDAP.

The following example shows two stanzas in /etc/security/user where only ldapuser is authenticated by LDAP. Note that ldapuser may not be present in /etc/security/passwd: the password file is looked only if LDAP query fails or times out.

```
ldapuser:
        SYSTEM=LDAP
        registry=LDAP
local:
        admin = false
```

Once user information is provided by LDAP, all requests regarding the user are made to the `secldapclntd` daemon, that takes care of retrieving the data from the LDAP servers. It is the only daemon that is allowed to contact the LDAP server for AIX security. In order to improve performance, it is a multithreaded process and caches collected data. If the daemon is not active, the client cannot contact the LDAP server.

The configuration file of the daemon is /etc/security/ldap/ldap.cfg and looks like the following example (if you do not use SSL, the last two lines are not present):

```
ldapservers:LDAPsrv
ldapadmin:cn=root
ldapadmpwd:rootpwd
ldapaixdn:cn=aixsecdb
ldapsslkeyf:/etc/keyfile.kdb
ldapsslkeypwd:sslpwd
```

All user configuration may be made by `mksecldap` command using the -u switch. You can provide a list of users or define all users to require a lookup in the LDAP database for authentication or you can use the ALL value to use LDAP for all users, as in the following example:

```
mksecldap -c -h LDAPserver -a cn=root -p rootpwd -u ALL
```

Note

If you ever need to stop the `secldapcltnd` daemon, do not use kill -9 since some IPC configuration will not be cleaned and you will not be able to restart the daemon. Use the USR1 signal instead as in the following example:

```
kill -s 30 <pid>
```

6.8.4 User Administration

The LDAP database contains all the information related to the users. The AIX commands that are used to administer user data like `mkuser`, `lsuser`, `mkgroup`, `lsgoup` have been enhanced to use LDAP. A new switch -R is used to select the authentication method.

For example, it is possible to create a new user and then view its parameters in the following way:

```
# mkuser -R LDAP login=false nologin
# lsuser -R LDAP nologin
nologin id=211 pgrp=staff groups=staff home=/home/nologin shell=/usr/bin/ksh login=false
su=true rlogin=true telnet=true daemon=true admin=false sugroups=ALL tpath=nosak ttys=ALL
expires=0 auth1=SYSTEM auth2=NONE umask=22 SYSTEM=compat loginretries=0 pwdwarntime=0
account_locked=false minage=0 maxage=0 maxexpired=-1 minalpha=0 minother=0 mindiff=0
maxrepeats=8 minlen=0 histsize=0 fsize=2097151 cpu=-1 data=262144 stack=65536
core=2097151 rss=65536 nofiles=2000
```

The values of the variables associated to the user have the same meaning as if present in the local files, so there is no semantic difference between LDAP provided data and local data. For example, a user that has been defined with remote login disabled cannot perform a remote login in any machine using LDAP security.

Note that user creation with `mkuser -R LDAP` does not modify the /etc/security/user file: the command only updates the LDAP database. If you want the user to login you must either have LDAP authentication enabled in the default /etc/security/user stanza or you must add a stanza in the file that instructs the system to use LDAP.

6.9 Microcode Packaging

In versions of AIX prior to Version 4.3, some IBM microcode entities resided in filesets that were prerequisites of other filesets. This meant that our OEM customers had to ship these filesets even if they were not required. In some cases, customers who developed their own additional features preferred to use their own microcode instead of the IBM-supplied microcode. In order for these OEM customers to replace the IBM microcode, it was necessary for them to modify AIX.

To rectify this situation, the following filesets were modified to remove the IBM microcode so that our OEM customers can ship AIX without having to perform any changes:

- bos.sysmgt.nim.master
- devices.mca.8fc8
    - Common token-ring Software
    - Token-Ring high-performance adapter diagnostics
    - Token-Ring high-performance adapter microcode
    - Token-Ring high-performance adapter software
- devices.mca.df9f
    - Direct-Attached disk diagnostics

    - Direct-Attached disk software
- devices.mca.ffe1
    - 128-port asynchronous adapter diagnostics
    - 128-port asynchronous adapter microcode
    - 128-port asynchronous adapter software

The devices.mca packages were a source of error because even though some systems do not have an MCA bus, other rspc packages regarded these MCA packages as prerequisites. The files needed by the other rspc packages were moved out of the devices.mca.* packages and into a separate fileset.

6.10 Online Alternate Disk Installation

The `alt_disk_install` command gives users another way to update AIX to the next release or maintenance level without having to schedule an extended period of system downtime.

The update can be performed in two ways:

mksysb image

Installing a mksysb requires a 4.3 mksysb image or 4.3 mksysb tape. The `alt_disk_install` command is called, specifying a disk or disks that are installed in the system but are not currently in use. The mksysb is restored to those disks, such that, if the user chooses, the next reboot will boot the system on a 4.3 system.

Note: If needed, the `bootlist` command can be run after the new disk has been booted, and the bootlist can be changed to boot back to the older version of AIX.

Cloning

Cloning allows the user to create a backup copy of the root volume group. Once created, the copy may be used either as a back up, or it can be modified by installing additional updates. One possible use might be to clone a running production system, then install updates to bring the cloned rootvg to a later maintenance level. This would update the cloned rootvg while the system was still in production. Rebooting from the new rootvg would then bring the level of the running system up to the newly installed maintenance level. If there was a problem with this level, simply changing the bootlist back to the original disk and rebooting would bring the system back to the old level.

Currently, you can run the `alt_disk_install` command on 4.1.4 and higher systems for both of these functions. The bos.alt_disk_install.rte fileset must be installed on the system to do cloning to an alternate disk, and the bos.alt_disk_install.boot_images fileset must be installed to allow a mksysb install to an alternate disk.

The mksysb image that is used must be created before installation and include all the necessary device and kernel support required for the system on which it is installed. No new device, or kernel support, can be installed before the system is rebooted from the newly-installed disk.

Note: The level of mksysb that you are installing must match the level of the bos.alt_disk_install.boot_images fileset. At this time, 4.3.3, 4.3.2, 4.3.1, and 4.3.0 mksysb images are supported. AIX 4.3.2 boot images are available only on the 4.3.2 installation media.

When cloning the rootvg volume group, a new boot image is created with the `bosboot` command. When installing a mksysb image, a boot image for the level of mksysb and platform type is copied to the boot logical volume for the new alternate rootvg. When the system is rebooted, the `bosboot` command is run in the early stage of boot, and the system will be rebooted again. This is to synchronize the boot image with the mksysb that was just restored. The system will then boot in normal mode.

At the end of the install a volume group, altinst_rootvg, is left on the target disks in the varied off state as a place holder. If varied on, it will show as owning no logical volumes, but it does, in fact, contain logical volumes. Their definitions have been removed from the ODM because their names now conflict with the names of the logical volumes on the running system. It is recommended that you do not vary on the altinst_rootvg volume group but just leave the definition there as a place holder.

When the system reboots from the new disk, the former rootvg will not show up in an `lspv` listing. The disks that were occupied by the rootvg will show up as not having a volume group. However, you can still use the `bootlist` command to change the bootlist to reboot from the old rootvg if necessary.

When the system is rebooted from the new altinst_rootvg, then `lspv` will show the *old* rootvg as old_rootvg so you will know which disk or disks your previous rootvg was on. There is also a -q option in alt_disk_install that will allow you to query to see which disk has the boot logical volume so you can set your bootlist correctly for cases when old_rootvg has more than one disk.

The alternate root file system is mounted as /alt_inst, so other file systems also have that prefix (/alt_inst/usr, /alt_inst/var). This is how they must be accessed if using a customization script.

Note: If you have created an alternate rootvg with `alt_disk_install`, but no longer want use it, or you want to run `alt_disk_install` commands again, do not run `exportvg` on altinst_rootvg. Simply run the `alt_disk_install -X` command to remove the altinst_rootvg definition from the ODM database.

The reason you cannot run the `exportvg` command (or the `reducevg` command) is that the logical volume names and file systems now have the real names, and exportvg removes the stanzas for the real file system from /etc/filesystems for the real rootvg.

If `exportvg` is run by accident, be sure to recreate the /etc/filesystems file before rebooting the system. The system will not reboot without a correct /etc/filesystems file.

6.10.1 alt_disk_install Command Syntax

The following is an example of the `alt_disk_install` command:

```
alt_disk_install -d device || -C [-i image.data ] [-s script ] [-R
resolv_conf ]
[-D] [-B] [-V] [-r] [-p platform ] [-L mksysb_level ]
[-b bundle_name [ [ -I installp_flags ] [-l images_location ] [-f
fix_bundle ]
[-F fixes ] [-e exclude_list ] [-w filesets ] target_disks...

alt_disk_install -X
```

The following is a description of `alt_disk_install` flags:

-d device
The value for device can be:

Tape device - for example, /dev/rmt0.

Path name of mksysb image in a file system.

Note: -d and -C are mutually exclusive.

-C Clone rootvg.

Note: -d and -C are mutually exclusive.

-i image.data
Optional image.data file to use instead of default image.data from mksysb image or image.data created from rootvg. The image.date

file name must be a full path name. For example, /tmp/my_image.data.

-s script Optional customization script to run at the end of the mksysb install or the rootvg clone. This file must be executable. This script is called on the running system before the /alt_inst file systems are unmounted, so files can be copied from the running system to the /alt_inst file systems before the reboot. This is the only opportunity to copy or modify files in the alternate file system because the logical volume names will be changed to match rootvg's, and they will not be accessible until the system is rebooted with the new alternate rootvg. You must use a full path name for script.

-R resolv_conf
Specifies the path name of the resolv.conf file you want to replace the existing one after the mksysb has been restored, or the rootvg has been cloned. You must use a full path name for resolv_conf.

-D Turn on debug (set -x output).

-V Turn on verbose output. This will show the files that are being backed up for rootvg clones. This flag will show files that are restored for mksysb alt_disk_installs.

-B Specifies not running bootlist after the mksysb or clone. If set, then the `-r` flag cannot be used.

-r Specifies to reboot from the new disk when the alt_disk_install command is complete.

-p platform
This is the platform to use to create the name of the disk boot image. The default value is the platform of the running system obtained with the `bootinfo -T` command on 4.1 or the `bootinfo -p` command in 4.2. This flag is only valid for mksysb installs (-d flag).

-L mksysb_level
This level is combined with the platform type to create the name of the boot image to use (IE rspc_4.3.0_boot). This must be in the form V.R.M. The default is 4.3.0. The mksysb image is checked against this level to verify that they are the same.

The following flags are only valid for use when cloning the rootvg (-C):

-b bundle_name
Path name of optional file with a list of packages, or filesets, that will be installed after a rootvg clone. The -l flag must be used with this option.

-I installp_flags
The flags to use when updating, or installing, new filesets into the cloned alt_inst_rootvg. The default flags are -acgX. The -l flag must be used with this option.

-l images_location
Location of the installp images, or updates, to apply after a clone of the rootvg. This can be a directory full path name or a device name (for example, /dev/rmt0).

-f fix_bundle
Optional file with a list of APARs to install after a clone of the rootvg. The -l flag must be used with this option.

-F fixes Optional list of APARs (such as IX123456) to install after a clone of the rootvg. The -l flag must be used with this option.

-e exclude_list
Optional exclude list to use when cloning rootvg. The rules for exclusion follow the pattern matching rules of the `grep` command. The exclude_list must be a full path name.

-w filesets
List of filesets to install after cloning a rootvg. The -l flag must be used with this option.

The following are supplied as parameters

Target Disks
Specifies the name, or names, of the target disks where the alternate rootvg will be created. The disk, or disks, must not currently contain any volume group definition. The `lspv` command should show these disks as belonging to volume group `None`.

6.10.2 Using alt_disk_install

The following are examples of using the `alt_disk_install` command:

1. To clone the running 4.2.0 rootvg to hdisk3 and apply updates from /updates to bring the cloned rootvg to a 4.2.1 level:

   ```
   #  alt_disk_install -C -F 4.2.1.0_AIX_ML -l /updates hdisk3
   ```

   The bootlist will then be set to boot from hdisk3 at the next reboot.

2. To install a 4.3 mksysb image on hdisk3, then run a customized script (/home/myscript) to copy some user files over to the alternate rootvg file systems before reboot:

```
# alt_disk_install -d /mksysb_images/4.3_mksysb -s /home/myscript
hdisk3
```

6.10.3 Alternate Disk Installation Enhancements (4.3.1)

The `alt_disk_install` command gives users another way to update AIX to the next release or maintenance level without having to schedule an extended period of system downtime. The function is included in the bos.alt_disk_install package, which is shipped on the AIX media. The package is not installed by default during system installation.

AIX 4.3.1 has enhanced the alternate disk installation function by splitting the task into three distinct phases. A system administrator now has greater control over the task by being able to perform each phase in isolation from the others. It is not required to perform them all at once.

The three phases of the alternate disk install are:

1. Phase One
   - Create the altinst_rootvg, logical volumes and file systems.
   - Restore or copy files to the /alt_inst file systems.
2. Phase Two
   - Copy a resolv.conf file to the /alt_inst/etc file system if specified.
   - Copy NIM client configuration information if specified.
   - For clones, install any new filesets, updates, and fixes.
   - Run any user specified customization script.
3. Phase Three
   - Manipulate the ODM databases and /etc/filesystems file.
   - Build the boot image.
   - Unmount the /alt_inst file systems and rename the logical volumes and file systems.

Phase two can be performed with phase one or phase three and can also be performed on its own multiple times if required before phase three is run.

The phases performed are controlled by the new -P option that has possible values as shown in Table 24.

Table 24. Possible Values of Phase Value

Flag Value	Result
1	Phase one performed
2	Phase two performed
3	Phase three performed
12	Phases one and two performed
23	Phases two and three performed
all	Phases one, two, and three performed

The target disk name is required for all phases, even though the altinst_rootvg has already been created.

Some standard alt_disk_install options, such as the reboot option and the no bootlist option, are not allowed in phase one or phase two.

Specifying an exclude list (-e exclude_list) is not allowed in phase two or phase three.

If a flag is used in a wrong phase, a warning is displayed, but the install does not terminate.

6.10.4 Alternate Disk Installation Enhancements (4.3.2)

In this section, the enhancements made at the AIX 4.3.2 introduction are given.

6.10.4.1 New alt_disk_install 4.3.2 Usage

Listed below are the new command formats for the various tasks given.

Create Alternate Disk: mksysb (-d) or clone (-C):

```
alt_disk_install {-d <device> | -C} [-i <image.data>] [-s <script>]
                 [-R <resolv_conf>] [-D] [-B] [-V] [-r]
                 [-p <platform>] [-L <mksysb_level>]
                 [-b <bundle_name>] [-I <installp_flags>]
                 [-l <images_location>] [-f <fix_bundle>]
                 [-F <fixes>] [-e <exclude_list>] [-w <filesets>]
                 [-n] [-P <phase_option>] <target_disks...>
```

Determine Volume Group Boot Disk (-q):

```
alt_disk_install -q <disk>
```

Rename Alternate Disk Volume Group (-v):

```
alt_disk_install -v <new volume group name> <disk>
```

Wake-Up Volume Group (-W):

```
alt_disk_install -W <disk>
```

Put-to-Sleep Volume Group (-S):

```
alt_disk_install -S
```

Clean Up Alternate Disk Volume Group (-X):

```
alt_disk_install -X [<volume group>]
```

6.10.4.2 Scenarios for Command Enhancements

To see which disks belong to the original rootvg volume group after the system has been rebooted from the alternate disk, an entry has been added to the database, so that when the system is rebooted from the alternate disk, and the `lspv` command is executed, a volume group name old_rootvg will be listed for the original rootvg disks.

```
# lspv
hdisk0        00006091aef8b687    old_rootvg
hdisk1        00076443210a72ea    rootvg
```

This volume group will be set to not varyon at reboot, and should only be removed with:

```
# alt_disk_install -X old_rootvg
# lspv
hdisk0        00006091aef8b687    None
hdisk1        00076443210a72ea    rootvg
```

New function was added to the -X flag to allow for specified volume group name information removal. It is recommended that you always use `alt_disk_install -X` when removing any information about an alternate volume group (altinst_rootvg, old_rootvg, and so on). alt_disk_install manages changes to the ODM that are required. Using `exportvg` or `reducevg` could cause serious problems to your system (like removing base entries in /etc/filesystems). `alt_disk_install -X` and `alt_disk_install -X <new volume group name>` will not remove actual data from the volume group. Therefore, you can still reboot from that volume group, if you reset your bootlist.

To see which disk is the boot disk, of the old_rootvg volume group, after a reboot from the alternate disk, the -q flag has been added to `alt_disk_install` to determine the boot disk, from the user specified disk and its associated volume group. The command syntax is:

```
# alt_disk_install -q <disk>
```

Given any disk in the volume group, and the command will return the actual boot disk (contains hd5), for that volume group.

```
# lspv
hdisk0         00006091aef8b687    old_rootvg
hdisk1         00076443210a72ea    rootvg
hdisk2         0000875f48998649    old_rootvg
# alt_disk_install -q hdisk0
hdisk1
```

In this case, the boot disk for old_rootvg is actually hdisk1. Therefore, you could reset your bootlist to hdisk1 and reboot to the original rootvg volume group.

```
# bootlist -m normal hdisk1
# reboot -q
```

This query will work on any volume group that has a boot (hd5) logical volume.

Different names for altinst_rootvg are now possible for the case that a user would want to have multiple alternate disks on one system (one with 4.3.2, one with 4.2.1, and so on). The -v flag was added to allow non-rootvg volume group names to be changed. The syntax for this is:

```
# alt_disk_install -v <new volume group name> <disk>
```

For example, on a 4.2.1 system, run alt_disk_install to restore a 4.3.2 mksysb to hdisk2 and hdisk3. Execute alt_disk_install, with the -v flag, to rename the altinst_rootvg volume group name. Then, on the same system, run alt_disk_install to clone the 4.2.1 system to hdisk4 and hdisk5. Finally, rename the cloned altinst_rootvg.

```
# lspv
hdisk0         00006091aef8b687    rootvg
hdisk1         00000103000d1a78    rootvg
hdisk2         000040445043d9f3    None
hdisk3         00076443210a72ea    None
hdisk4         0000875f48998649    None
hdisk5         000005317c58000e    None
# alt_disk_install -d /dev/rmt0 hdisk2 hdisk3
...
# lspv
hdisk0         00006091aef8b687    rootvg
```

```
hdisk1         00000103000d1a78    rootvg
hdisk2         000040445043d9f3    altinst_rootvg
hdisk3         00076443210a72ea    altinst_rootvg
hdisk4         0000875f48998649    None
hdisk5         000005317c58000e    None
# alt_disk_install -v alt_disk_432 hdisk2
# lspv
hdisk0         00006091aef8b687    rootvg
hdisk1         00000103000d1a78    rootvg
hdisk2         000040445043d9f3    alt_disk_432
hdisk3         00076443210a72ea    alt_disk_432
hdisk4         0000875f48998649    None
hdisk5         000005317c58000e    None
# alt_disk_install -C hdisk4 hdisk5
...
# lspv
hdisk0         00006091aef8b687    rootvg
hdisk1         00000103000d1a78    rootvg
hdisk2         000040445043d9f3    alt_disk_432
hdisk3         00076443210a72ea    alt_disk_432
hdisk4         0000875f48998649    altinst_rootvg
hdisk5         000005317c58000e    altinst_rootvg
# alt_disk_install -v alt_disk_421 hdisk4
# lspv
hdisk0         00006091aef8b687    rootvg
hdisk1         00000103000d1a78    rootvg
hdisk2         000040445043d9f3    alt_disk_432
hdisk3         00076443210a72ea    alt_disk_432
hdisk4         0000875f48998649    alt_disk_421
hdisk5         000005317c58000e    alt_disk_421
# alt_disk_install -q hdisk3
hdisk2
# bootlist -m normal hdisk2
# sync
# reboot -q
After the system reboot, perform the following steps:
# lspv
hdisk0         00006091aef8b687    old_rootvg
hdisk1         00000103000d1a78    old_rootvg
hdisk2         000040445043d9f3    rootvg
hdisk3         00076443210a72ea    rootvg
hdisk4         0000875f48998649    alt_disk_421
hdisk5         000005317c58000e    alt_disk_421
```

A way to wake up a volume group for data access between the alternate disk and the original rootvg and also a way to put the volume group back to sleep is provided.

```
The syntax for the wake-up function is:
       # alt_disk_install -W <disk>
```

Note, the volume group that will experience the wake-up will be renamed altinst_rootvg.

The booted volume group's version of AIX must be later or equal to the version of AIX on the volume group that will undergo the wake-up. This may mean that you will need to boot from the altinst_rootvg and wake-up the old_rootvg.

```
# oslevel
4.1.0.0
# lspv
hdisk0         000040445043d9f3    rootvg
hdisk1         00076443210a72ea    None
# alt_disk_install -d /dev/rmt0 hdisk1
...
# lspv
hdisk0         000040445043d9f3    rootvg
hdisk1         00076443210a72ea    altinst_rootvg
# reboot -q
```

After rebooting...

```
# oslevel
4.3.0.0
# lspv
hdisk0         000040445043d9f3    old_rootvg
hdisk1         00076443210a72ea    rootvg
# alt_disk_install -W hdisk0
# lspv
hdisk0         000040445043d9f3    altinst_rootvg
hdisk1         00076443210a72ea    rootvg
```

At this point, you will find the altinst_rootvg volume group varied on and the /alt_inst file systems will be mounted.

Now, to put the volume group to sleep the command syntax is:

```
# alt_disk_install -S
# lspv
hdisk0         000040445043d9f3    altinst_rootvg
hdisk1         00076443210a72ea    rootvg
```

The altinst_rootvg is no longer varied-on and the /alt_inst file systems are no longer mounted. If desired and the bootlist is reset, this volume group is now ready for booting. If it is necessary for the altinst_rootvg volume group name to be changed back to old_rootvg, this can be done with the -v flag.

```
# alt_disk_install -v old_rootvg hdisk0
# lspv
hdisk0         000040445043d9f3    old_rootvg
hdisk1         00076443210a72ea    rootvg
```

6.11 Printer Support

The AIX spooler subsystem was already significantly enhanced over the basic UNIX spooler. For AIX Version 4.3, the following additional enhancements have been included:

6.11.1 Remote Printing Robustness

Both `rembak` and `lpd` have new flags that allow you to build a log file. A log file is helpful in determining why a daemon failed. Use the following commands to start error logging:

```
startsrc -s lpd -a "-D /tmp/lpddebug"
/etc/qconfig -D /tmp/remback_debug flag backend=/usr/lib/lpd/rembak
```

Support in SMIT was updated to allow `rembak` error logging to be enabled when adding a remote queue.

6.11.2 Remote Print Job Count

On a print server, when a job is received from the print client, `lpd` receives a control file and one or more data files. The control file contains information on the job to be printed, including the name of the corresponding data files. The datafiles contain the actual data to be printed. The control file is used to generate the arguments to the `enq` command. It is subsequently deleted. The datafiles are copied to the spooling directory so they can be processed by `qdaemon`.

The LPR/LPD protocol (RFC 1179) specifically designates the naming convention for print data files sent from print clients to print servers. Part of the name is a three-digit job ID. Due to this specification, problems could arise when a single print client sent more than 1000 print jobs to a print server. The datafiles became non-unique and the older file with the same name would get deleted. This could cause major problems for high-volume printing installations.

AIX version 4.3.0 has modified the function of the print server. When files are received by `lpd`, (before they are copied to the spooling directory) a time stamp is appended to the data file names, thus generating unique file names. This enables a large number of jobs to be submitted. The print server still conforms to the specification of the LPD protocol, which does not stipulate what happens to the data file once it has been received by the server.

6.11.3 Additional Printer Support

AIX Version 4.3 now includes native support for five additional Lexmark dot-matrix printers:

- Lexmark 2380 Model 3
- Lexmark 2381 Model 3
- Lexmark 2390 Model 3
- Lexmark 2391 Model 3
- Lexmark Forms Printer Model 4227

More printer support was specifically introduced in AIX 4.3.1:

- Hewlett-Packard 4000
- IBM InfoPrint 20

More printer support was specifically introduced in AIX 4.3.2 for IBM, Lexmark and Hewlett-Packard printers:

- IBM InfoPrint 32
- Lexmark Optra Color 40
- Lexmark Optra Color 45
- Lexmark Optra Color 1200
- Lexmark Optra K 1220
- Hewlett-Packard 8000
- Hewlett-Packard 8500 Color

More printer support is introduced in AIX 4.3.3 for Hewlett-Packard, IBM and Lexmark printers:

- Hewlett-Packard 2500C Color Printer
- Hewlett-Packard Color LaserJet 4500
- Hewlett-Packard LaserJet 8100
- Hewlett-Packard LaserJet 5000 D640 Printer
- IBM InfoPrint 40
- Lexmark Optra E310 Laser Printer
- Lexmark Optra M410 Laser Printer
- Lexmark Optra Se Laser Printer
- Lexmark Optra T Laser Printer Family

- Lexmark Optra W810 Laser Printer

The HP 8000 and HP 8500 Color printers and the associated AIX print drivers support A3 paper size.

For all printers, this includes:

- Printer backend colon files for each printer data stream supported by the printer.
- ODM parameters for the printer device driver and diagnostics.
- New message catalog entries for the printer name and new printer attributes (if required).
- Packaging files to make support for the printer's separately installable packages.

More information and colon files can be obtained from Lexmark. In the USA, call 1-800-Lexmark (1-800-539-6275) or visit their Web site at the following URL:

`http://www.lexmark.com`

6.11.4 Print Job Administration Enhancements (4.3.2)

The print queue administration commands have been enhanced to support print queues with more than 1000 jobs. Previous editions of AIX would allow more than 1000 jobs in a print queue. Cancelling or altering a job when the queue size grew more than 1000 became difficult, because job numbers would repeat, and specifying a specific job number would not guarantee that the job selected would be unique and the one desired.

The formatting of the output of the `qchk` command, when used with the -W flag, has been changed to show the six-figure print job number. The lpstat command has been changed to also accept the -W flag to show information in wide format. Use of the -W flag results in output where the lines are over 106 characters in length. It can be quite confusing to read the output when using a screen that is only 80 characters wide. To maintain compatibility for any user scripts that parse the output of these commands, the default format for both remains unchanged from previous versions.

```
# qchk
-W
Queue               Dev            Status       Job Files              User
     PP   %  Blks  Cp Rnk
------------------- -------------- --------- ------ ------------------ --------
-- ---- --- ----- --- ---
ps                   lp0           DOWN
                                   QUEUED      2228 /etc/passwd        root
                1   1   1
                                   QUEUED      2229 /etc/passwd        root
                1   1   2
```

```
                                     QUEUED      2230 /etc/passwd        root
                1   1   3
                                     QUEUED      2231 /etc/passwd        root
                1   1   4
#
```

The enq, cancel, qpri, qcan, and lprm commands have been altered to accept six-figure job numbers.

The enhancement applies only to jobs submitted to local print queues. Jobs submitted to remote printers will still have three digit print job numbers. This is because of a restriction in the lpd protocol.

Although AIX 4.3.2 now generates six digit job numbers for local jobs, the response time of the qchk and lpstat commands is identical to that on AIX 4.3.0. The time taken to list the jobs on the queue is proportional to the number of jobs. It is suggested that you maintain a queue size less than 1000 unless absolutely necessary, because larger queue sizes will impact performance.

6.12 System Resource Controller Subsystem Enhancements (4.3.2)

Two major enhancements have been introduced to the System Resource Controller (SRC) subsystem in AIX 4.3.2. They are aimed at increasing the reliability and scalability of both the various subsystems that are controlled by SRC and the SRC itself. The following sections explain these enhancements:

6.12.1 Recoverable SRC Daemon

The SRC is a subsystem controller that facilitates the management and control of complex subsystems. The SRC provides a single set of commands to start, stop, trace, refresh, and query the status of a subsystem. If a subsystem should fail for any reason, the SRC can automatically restart it.

If the SRC itself were to fail for any reason, it would be restarted due to its entry in /etc/inittab, as shown in the following example:

```
srcmstr:2:respawn:/usr/sbin/srcmstr # System Resource Controller
```

The respawned SRC is, however, unable to control or monitor the subsystems started by the previous instance of SRC since they will have been inherited by the init process when the original SRC terminated. As a result, the lssrc command will show such a subsystem as inoperative, even though it is still running. In addition, the startsrc command can be used to start a second instance of the subsystem, even though the subsystem definition explicitly forbids multiple instances.

The SRC in AIX 4.3.2 has been enhanced to allow a respawned `srcmstr` daemon to monitor and control the subsystems started by the previous instance of the daemon. This has been achieved using the following enhancements.

The SRC now keeps an external list of the subsystems under its control in the file /var/adm/SRC/active_list. This file is for use by the SRC system only, therefore the format is unpublished. A respawned `srcmstr` daemon will read the contents of this file to update its internal list of the currently running subsystems. This will allow the `lssrc` command to correctly determine the status of the running subsystems, even though they were not necessarily started by the current instance of the `srcmstr` daemon.

A respawned `srcmstr` daemon uses a new kernel extension to register interest in the termination of certain processes. This allows a respawned `srcmstr` daemon to be informed of the termination of subsystems started by the previous instance of the daemon. A child process is created to communicate with the kernel extension. The child process in turn communicates with the srcmstr daemon. The presence of the child process, called `srcd`, indicates that the `srcmstr` daemon has been restarted.

```
# ps -ef | grep src
    root  4650     1   0   Aug 21      -  0:00 /usr/sbin/srcmstr
    root 24680  4650   0   Aug 21      -  0:00 srcd
    root 25894  7030   2 10:03:38  pts/1  0:00 grep src
#
```

If a subsystem fails for any reason while under the control of a respawned `srcmstr` daemon, it will be restarted if the subsystem policy requires it. In this case the exit code of the subsystem is not available to SRC due to the method used by the kernel extension to detect process termination. This is still preferable however to the previous function of SRC that would not detect subsystem failure at all in this situation.

If you run the `lssrc` command with the -S flag, you receive a list of the subsystem attributes. The *action* is set to -R (for respawn) or -O (for once). The value of action must be -R to have the subsystem restarted. Also, there is a retry limit. If the subsystem fails more than once within the configured wait time (20 seconds by default), it will not be restarted.

The action and waittime attributes can be set using the `mkssys` command or changed with the `chssys` command.

This new feature of the `srcmstr` daemon can be disabled if required by specifying the -B option when starting the daemon. This is usually performed by an entry in /etc/inittab.

6.12.2 Thread-Safe Routines in libsrc

In previous versions of AIX, some of the libsrc subroutines are neither threadsafe nor reentrant. This prevents other libraries and applications that call these libsrc subroutines from achieving thread-safety and reentrance requirements.

The libsrc subroutines of interest in a threaded environment are those that support communication with an SRC subsystem. In other words, the routines that are used by an application, that may be multithreaded, to interrogate the SRC.

The libsrc subroutines that update the subsystem configuration data, and those that are used by SRC commands to process input parameters, are not required in a threaded environment. This is because the applications that use these routines, the srcmstr daemon itself along with `lssrc` and related commands, are not threaded applications.

The threadsafe and reentrant routines are shown in Table 25. The new function has been implemented by changing the internals of some routines and by providing new threadsafe and reentrant versions of other routines. The new routines are indicated by the _r extension on the name. Where a new routine has been implemented, the original non-threadsafe version has been retained for use by non-threaded applications and for binary compatibility with previous versions of AIX.

Table 25. Threadsafe Routines in libsrc

New threadsafe routines	Existing routines made threadsafe
`src_err_msg_r`	`srcsrpy`
`srcrrqs_r`	`srcstathdr`
`srcstattxt_r`	`srcstop`
`srcstat_r`	`srcstrt`
`srcsrqt_r`	
`srcsbuf_r`	

6.13 TTY Remote Reboot (4.3.2)

AIX 4.3.2 has added the ability to communicate with a system that has stopped responding on the network but is still processing device interrupts. The feature allows a system administrator to force a machine to take a predetermined action when a user defined character sequence is entered on a serial port. The feature can only be enabled on native serial ports. Only one serial port on a machine can be configured for remote reboot. Serial ports configured on 8, 16, 64, and 128 port adapter cards are not supported.

The feature is configured by setting two ODM attributes that have been added to native serial ports. The new attributes are reboot_enable and reboot_string. The reboot_enable attribute has possible values of no, reboot, and dump. The reboot_string attribute is used to store a case sensitive user defined string up to 16 characters in length. It is advised to choose an unusual character sequence that would never normally be typed. For example ReEbOoTmE. This allows the serial port to be used for a normal login session, if required, without the possibility of accidentally rebooting the system.

Table 26. Settings of reboot_enable Attribute

Value of reboot_enable Attribute	**Result**
no	Remote reboot disabled. No action taken if reboot_string is entered.
reboot	Machine will reboot when reboot_string is entered and confirmed.
dump	Machine will dump system image to dump device when reboot_string is entered and confirmed.

The settings appear on the **SMIT Add a TTY**, and **Change / Show Characteristics of a TTY** panels as:

```
REMOTE reboot ENABLE                                   no
REMOTE reboot STRING                                  [#@reb@#]
```

Interrupts must be enabled on the port for this feature to be active. One way to insure interrupts are enabled is to enable login on the port; with the port enabled, getty is running and holding the device open, although the user does not need to be logged in for the system to recognize the reboot string.

If the user defined reboot string is entered when reboot_enable is set to reboot or dump, the user defined string is erased from the screen and replaced with the symbol > that is the confirmation prompt. If the user presses the **1** key on the keyboard, then the predefined action specified by reboot_enable will occur. If the user presses any other key, the user defined string reappears on screen; the subsequent character is appended, and no other action is taken.

An error log entry is made when the remote reboot facility is enabled or disabled on a serial port. An entry is also made when the facility is used to reboot a machine or force a system dump. The entry is created when the machine next starts the errorlog daemon indicating the action taken and the name of the tty device used to initiate the action.

```
---------------------------------------------------------------------------
LABEL:          TTY_RRB
IDENTIFIER:     1960E672

Date/Time:       Fri Aug 28 14:54:41
Sequence Number: 20
Machine Id:      000044091C00
Node Id:         aix4xdev
Class:           O
Type:            INFO
Resource Name:   Remote Reboot

Description
SYSTEM REBOOTED USING TTY REMOTE REBOOT.

User Causes
SYSTEM REBOOTED USING TTY REMOTE REBOOT.

Detail Data
TTY LOGICAL NAME
tty0
---------------------------------------------------------------------------
```

The remote reboot function is intended to be used on remote server machines that do not have a service processor. Ordinarily, the serial port with remote reboot enabled would be connected to a modem to allow remote support staff to reboot the machine if it fails to respond on the network in the normal manner.

It is the system administrator's responsibility to provide physical security on any serial port with remote reboot enabled. This is because any user can determine the reboot string by using the `lsattr` command with the appropriate logical device name. It is not possible to enable a password protected reboot string, as this would require the code checking the password to use the crypt() function. Since the code checking the string is running at the highest interrupt priority, any increase in the time taken to service the interrupt may cause other device interrupts to be lost with unpredictable results.

6.14 Network Install Manager Enhancements

The Network Install Manager (NIM) subsystem has been enhanced in AIX 4.3.2 and AIX 4.3.3 to offer greater control over NIM operations. The system has been changed to allow more concurrent NIM operations and restrict the number of concurrent NIM operations.

6.14.1 Restrict Concurrent Group Operations (4.3.2)

A NIM machine group allows an administrator to use a single command to initiate the same NIM action on many machines at the same time. Depending on the NIM operation and numbers of machines involved, this can sometimes lead to resource constraints. For example, many machines performing a BOS install action could saturate a network segment.

Two new settings are available when performing NIM operations on group resources. Together they allow the administrator to specify how many concurrent operations should be attempted on machines in the group and for how long the NIM server should continue to initiate the operations.

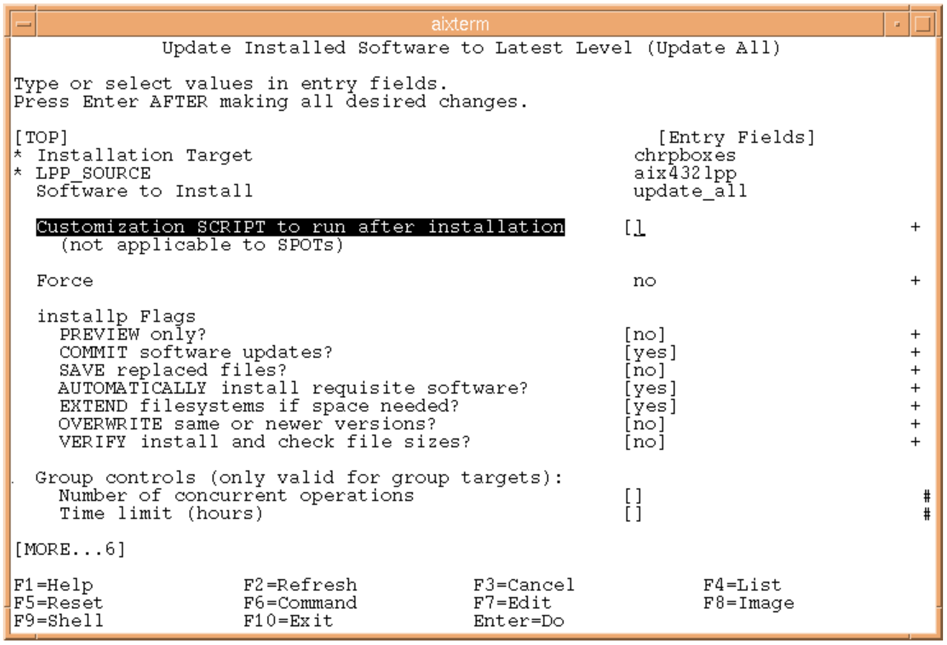
For example, this would allow the administrator of a NIM environment with a machine group of 100 machines to initiate a NIM operation on the group and to specify that no more than 10 machines in the group should have the operation in progress at any one time. This ensures that the network bandwidth is not exhaustively consumed. When a NIM operation completes on a client machine, the NIM server initiates an operation on the next machine in the group until all group members have been processed, or the time limit has been exceeded. The options are in place for the duration of the NIM operation. Subsequent NIM operations on the group can use different values if desired.

The options are valid only for certain operations when a NIM group is used as the target. The NIM operation will fail with an error message if the options are used for individual machine, LPP, or SPOT targets. The options appear near the end of the following NIM SMIT panels that initiate operations likely to generate large amounts of network traffic:

- Install the Base Operating System on Stand-alone Clients
- Install and Update from LATEST Available Software
- Update Installed Software to Latest Level (Update All)
- Install and Update Software by Package Name (includes devices and printers)
- Install Software Bundle (Easy Install)
- Update Software by Fix (APAR)

- Install and Update from ALL Available Software
- Install mksysb on an Alternate Disk
- Clone the rootvg to an Alternate Disk

Figure 32 shows an example of the new NIM settings within SMIT.



```
            Update Installed Software to Latest Level (Update All)

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

[TOP]                                                   [Entry Fields]
* Installation Target                                 chrpboxes
* LPP_SOURCE                                          aix432lpp
  Software to Install                                 update_all

  Customization SCRIPT to run after installation     []                  +
    (not applicable to SPOTs)

  Force                                               no                  +

  installp Flags
    PREVIEW only?                                    [no]                 +
    COMMIT software updates?                         [yes]                +
    SAVE replaced files?                             [no]                 +
    AUTOMATICALLY install requisite software?        [yes]                +
    EXTEND filesystems if space needed?              [yes]                +
    OVERWRITE same or newer versions?                [no]                 +
    VERIFY install and check file sizes?             [no]                 +

  Group controls (only valid for group targets):
    Number of concurrent operations                  []                    #
    Time limit (hours)                               []                    #

[MORE...6]

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

Figure 32. Sample NIM SMIT Panel Showing Group Controls

6.14.2 Resource Lock Contention (4.3.2)

The lock granularity of the NIM subsystem has been improved to allow more operations in parallel. Previous versions of NIM would lock an object for the duration of some operations, thus preventing any other operation on the same object.

The locking methodology has been changed to lock the object only for critical parts of the operation. This will allow other operations on the object to complete when in the past they may have waited or timed out. For example, this change will allow a showlog operation to be carried out on a machine resource, which a customer operation is also being carried out. Previously, the machine object would be locked for the entire duration of the customer operation.

6.14.3 Administration Enhancements (4.3.2)

The NIM sections of SMIT and Web-Based System Manager have been updated to offer function that was previously only available using the NIM command line interface. This includes support for ATM network types and IEEE 802.3 Ethernet networks.

6.14.4 bosinst.data Resource Handling (4.3.3)

The bosinst.data file is of vital importance in the restore of system backup images, either using NIM and attached tapes or CDs. Syntax errors or elements missing on the file may cause the restore to fail. New features in AIX 4.3.3 give more means to reduce errors.

6.14.4.1 File Checking

A new command has been added in the bos.sysmgt.sysbr fileset that enables you to verify the correctness of the bosinst.data file. The root user and system group can execute the following script:

```
/usr/lpp/bosinst/bicheck <filename>
```

where filename is the bosinst.data file you want to verify: the command returns error code 0 if no error has been found or error code 1 and a message on standard error with the name of stanzas and fields which have incorrect values. All the file is read and multiple errors may be highlighted.

The `bicheck` command verifies the existence of the control_flow, target_disk_data, and locale stanzas as needed. The value for each field, if given, is confirmed to match an allowable value, or other limitations, if they exist. If a non prompted install is specified, the existence of values for required fields are confirmed. If a dump stanza exists, the values are determined to match the allowable values or be within other limitations, where they exist.

6.14.4.2 File Update During mksysb

When a system backup is created using the `mksysb` command, the data file bosinst.data that describes how the system backup will be reinstalled is a copy of the bosinst.data created when the system was first installed.

If disks have been added to the rootvg, the changes are not reflected in the bosinst.data file and when the system backup is reinstalled problems will arise. During a non prompted install, the incorrect data can lead to a failed install.

A common case in which the install fails is when a second disk is added to the rootvg, so that logical volumes can be mirrored to a separate disk. When reinstalling a mksysb from one of these systems, the install will fail due to lack of space, unless both disks were selected to be in the rootvg.

In AIX 4.3.3 the `mksysb` and `savevg` commands are able to check if the bosinst.data file contains all the disks present in rootvg or if stanzas related to removed disks are still present and they can correct the file content.

The `mksysb` and `savevg` commands look also for existence of /save_bosinst.data_file file and they do not update an existing /bosinst.data file if the /save_bosinst.data_file is found. Otherwise, they check the /bosinst.data file and correct any missing or incorrect information in the target_disk_data stanza. If the /bosinst.data file does not exist, they copy the /var/adm/ras/bosinst.data file to /bosinst.data and correct the target_disk_data stanzas as needed. If /save_bosinst.data_file exists, but there is no /bosinst.data file, they behave as though /save_bosinst.data_file does not exist.

6.14.5 NIM Security (4.3.3)

NIM’s method of running commands on remote clients is based on standard AIX authentication and, starting from AIX version 4.3.1, Kerberos 5. Kerberos provides better security but is not always available.

In SP environments Kerberos 4 is used for authenticating remote commands. In order to improve security and to give all NIM installation benefits to SP customers, AIX 4.3.3 supports also Kerberos 4 authentication in NIM operations.

The NIM master must have Kerberos 4 authentication enabled to support clients that only have Kerberos 4 authentication. The NIM master may also have other authentication methods enabled, such as standard AIX or Kerberos 5. The NIM master attempts all authentication methods until a successful method is reached. If there are no remote execution access for the master on the client then the NIM commands fails.

The NIM master supports the clients that have one of the authentication methods that the NIM master has.

NIM is not responsible for configuring Kerberos 4 or disabling standard security AIX on the clients. Kerberos 4 authentication is implemented utilizing the PSSP 3.1 (or later) Kerberos 4 authentication commands. In secure environments, if you wish to support only Kerberos 4 on his clients, then you need to remove the /.rhosts file from the client after the defining a machine as a NIM client. NIM is not responsible for prompting for tokens in the Kerberos 4 environment. Any tokens that are required should be acquired prior to running any NIM commands.

If secure clients are reinstalled with BOS (Base Operating System), the authentication methods on the NIM master should be set for both Kerberos 4 and Standard UNIX. This is because NIM does not configure Kerberos 4 on the client after the BOS is installed. NIM relies on standard /.rhosts to guarantee that it can remotely execute commands on the client until the client is configured by the system administrator with Kerberos 4 and made into a secure client.

6.14.6 NIM Scalability (4.3.3)

NIM has been designed to install machines over the network. It was not designed to install large quantities of machines at the same time. Steps have been taken to increase the capability of installing more machines simultaneously, reducing the amount of time on locks being held, distributing resources around the network, and exporting NIM resources in a global manner.

6.14.6.1 nimesis Daemon

All NIM clients communicate with the NIM database during an install sending their status changes. A new value is sent about every minute to the master to communicate progress and also to identify the last task completed on the client. All these commands get funneled though the `nimesis` daemon.

Before AIX 4.3.3 several `nimesis` processes were launched to manage each command that comes from the client and when a large group of machines were installed at the same time this could cause many processes on the master at a single given time. Either some of the commands may fail due to lack of swap space or the NIM master may become really slow due to the mass quantity of processes required. Overloads have been reported during an install of over 30 or 40 client machines at the same time.

Clients send asynchronous message to notify NIM master of state changes or result values making the master change the status of NIM objects. If these commands are not handled by the master server, the NIM client may not be placed into the correct state at the completion of an install. The resources used by this client may not get deallocated. This may cause problems and confusion for the NIM administrator.

Starting from AIX 4.3.3, the `nimesis` daemon is a multithreaded process that handle `nimclient` commands to support installations of around 200 client machines simultaneously, with better performance than previous releases.

A new max_nimesis_threads attribute for the master object has been added that indicate how many threads the `nimesis` should use. The new `nimesis` daemon is now made by only two processes: one is similar to the previous version daemon and then a new multithreaded process that handles the `nimclient` command on the master. The first process basically accepts the connection from the client machines and then passes the socket descriptor through a stream socket to the second process. The second process consists of several circular queues that handle the buffering of client commands.

The default value of max_nimesis_threads is 20, as you can see using the `lsnim -l master` command:

```
# lsnim -l master
master:
   class               = machines
   type                = master
   comments            = machine which controls the NIM environment
   platform            = chrp
   netboot_kernel      = mp
   if1                 = itso 433c.itsc.austin.ibm.com 0020357C5FD8
   ring_speed1         = 16
   Cstate              = ready for a NIM operation
   prev_state          = ready for a NIM operation
   Mstate              = currently running
   serves              = boot
   serves              = nim_script
   master_port         = 1058
   registration_port   = 1059
   reserved            = yes
   if_defined          = chrp.mp.tok
   if_defined          = rspc.up.tok
   max_nimesis_threads = 20
```

If the max_nimesis_threads attribute is present and has a value different from zero, the new multithreaded version of the daemon is used, otherwise the non-threaded version is activated.

You control the nimesis behavior executing the `nim` command:

```
nim -o change -a max_nimesis_threads=<max number of threads> master
```

or you can use the Web-Based System Manager or SMIT interface. The following SMIT panel (fast path nim_tune_nimesis) lets you select the `nimesis` behavior disabling the tuning or defining the number of nimesis threads that handle simultaneous requests.

```
 Tune Client Request Processing

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

                                                         [Entry Fields]
* Client Request Tuning?                              [enable]                +
    Maximum simultaneous requests                     [40]                     #








F1=Help             F2=Refresh          F3=Cancel           F4=List
Esc+5=Reset         F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

In the example, the number of threads is set to 40. You can increase the value up to 150, but a value of 50 is enough to manage about 200 concurrent installations: each thread is able to handle more than one client.

6.14.6.2 nim_script Resource

The nim_script resource defines the directory containing customization scripts created by NIM.

AIX 4.3.3 supports the nim_script resource to reside on NIM servers other than the NIM master. This helps reducing the resource contention on the master during installations of large quantities of machines. This also allows the nim_script resource to potentially be better located in the network, reducing network contention.

There is only one nim_script object used by all NIM machines and NIM places it on the most logical NIM server machine for the operation according to the following ordering rules:

- If it is a bos_inst operation, locate the nim_script on the SPOT server.
- If it is a customer operation and a lpp_source is allocated, locate the nim_script on the on the lpp_source server.
- If it is a cust operation and no lpp_source is allocated and a script resource is allocated, locate the nim_script on the on the script server.
- If the prior three locations fail then allocate it on NIM master.

6.14.6.3 Resource Propagation

As customers begin creating larger NIM environments and attempt to install all the machines at the same time, NIM tends to buckle under the pressure. SP install works around this by hiding many NIM masters in their environment, usually one per frame. This allows less global traffic and keeping the traffic more local to the SP frames themselves. As part of this scheme, SP does its own resource distribution management.

In an attempt to make NIM more scalable, better facilities for distributing the install resources to other NIM servers have been added. NIM already provides the capability to propagate SPOT and lpp_source resources. Starting from AIX 4.3.3, all install resources may be propagated to other NIM servers.

The concept is that the NIM administrator may propagate the resources to NIM servers in the same subnet or at least in closer proximity to the clients being installed. NIM then allows the administrator the capability to propagate NIM install resources to any NIM client machine. This potentially reduces resource contention on one NIM server and possibly lessen the network contention in the environment.

The resources that may have *replicas* on more than one server are:

- mksysb
- script
- bosinst_data
- image_data
- installp_bundle
- fix_bundle
- resolv_conf
- exclude_files

This first implementation does not provide a way to keep the replicated resources in sync. It is designed to take a snap-shot of one resource and copy it to another location.

The new SMIT panels for script, exclude_files, installp_bundle, fix_bundle, bosinst_data, image_data and resolv_conf resources have a new Source for replication field, as shown in the following example:

```
 Define a Resource

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

                                                    [Entry Fields]
* Resource Name                                  []
* Resource Type                                   script
* Server of Resource                             []                        +
* Location of Resource                           []                         /
  Comments                                       []

  Source for Replication                         []                        +


F1=Help             F2=Refresh          F3=Cancel           F4=List
Esc+5=Reset         F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

You create the original resource without filling the new field. Then you can ask NIM to create a new replica (a copy) of the resource defining a new resource located on another NIM server and you give in the Source for Replication the name of the original resource.

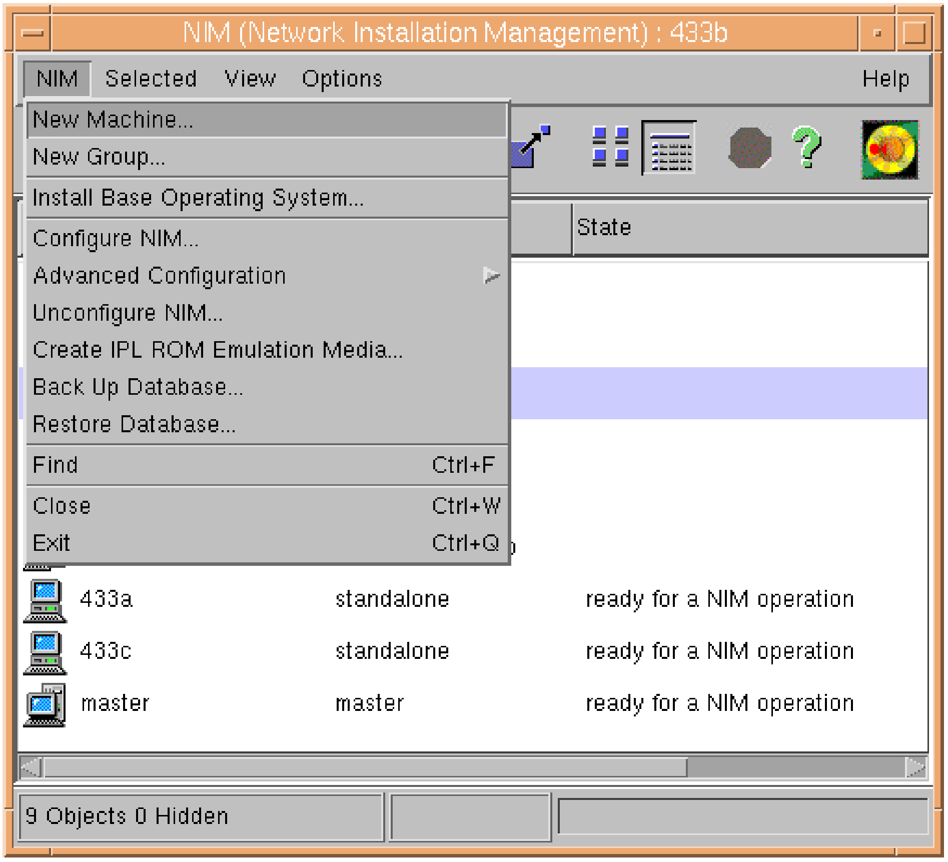
The mksysb resource allocation is a little different but the only new addition is the Source for Replication field, as shown in the following example:

```
 Define a Resource

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

                                                      [Entry Fields]
* Resource Name                                    []
* Resource Type                                     mksysb
* Server of Resource                               []                        +
* Location of Resource                             []                         /
  Comments                                         []

  Source for Replication                           []                        +
                -OR-
  System Backup Image Creation Options:
    CREATE system backup image?                     no                       +
    NIM CLIENT to backup                           []                        +
    PREVIEW only?                                   no                       +
    IGNORE space requirements?                      no                       +
    EXPAND /tmp if needed?                          no                       +
    Create MAP files?                               no                       +
    Number of BLOCKS to write in a single output   []                         #
     (leave blank to use system default)
    Use local EXCLUDE file?                         no                       +
     (specify no to include all files in backup)
                -OR-
    EXCLUDE_FILES resource                         []                        +
     (leave blank to include all files in backup)

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

The same feature has been added in Web-Based System Manager panels that allow resource allocation.

6.14.7 Web-Based System Manager Improvements (4.3.3)

New functions and task guides have been added to NIM Web-Based System Manager interface to ease NIM operations. By launching the Web-Based System Manager interface and selecting NIM from the launch pad, you can select the new task guides selecting the **NIM** menu on the upper bar, as shown in Figure 33.


NIM (Network Installation Management) : 433b

NIM Selected View Options Help

New Machine...
New Group...
Install Base Operating System...
Configure NIM...
Advanced Configuration
Unconfigure NIM...
Create IPL ROM Emulation Media...
Back Up Database...
Restore Database...
Find Ctrl+F
Close Ctrl+W
Exit Ctrl+Q

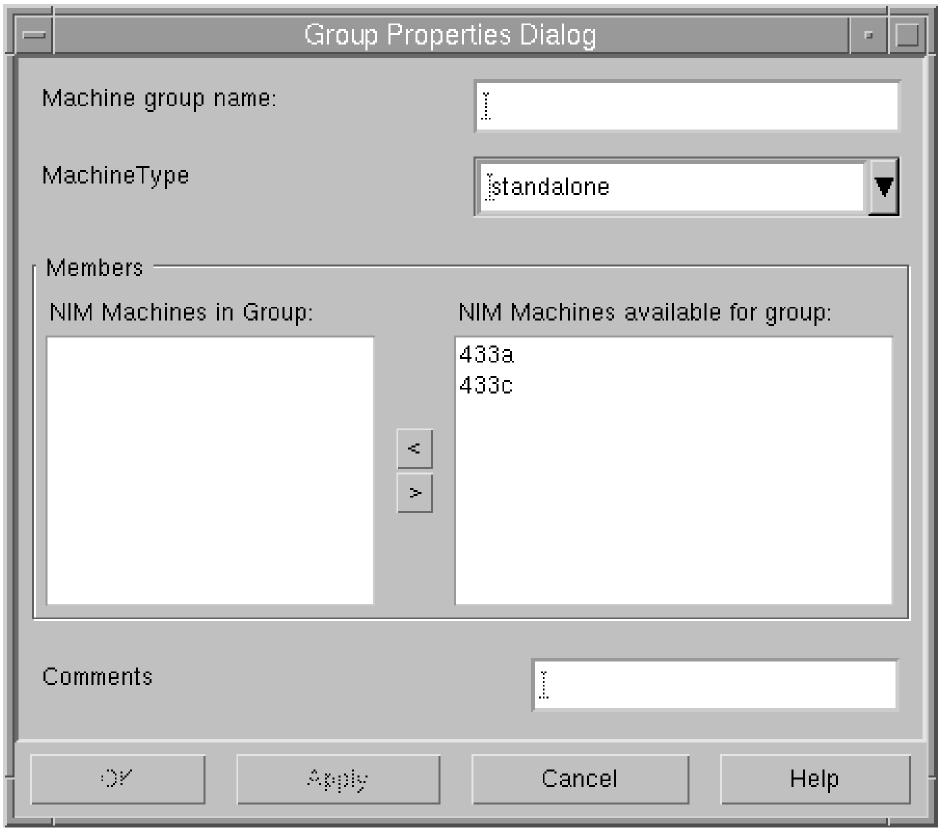
		State
433a	standalone	ready for a NIM operation
433c	standalone	ready for a NIM operation
master	master	ready for a NIM operation

9 Objects 0 Hidden

Figure 33. NIM Web-Based System Manager Panel

6.14.7.1 Machine Group Definition

Machine groups are present in NIM starting from AIX 4.3.0, but no support was present in Web-Based System Manager. In AIX 4.3.3 they can be directly used by Web-Based System Manager. Selecting **NIM** and then **New Group...** the following panel appears (Figure 34), from where you can define the new machine group and add into it any number of NIM clients among those you have previously defined:



Machine group name:

MachineType standalone

Members

NIM Machines in Group:

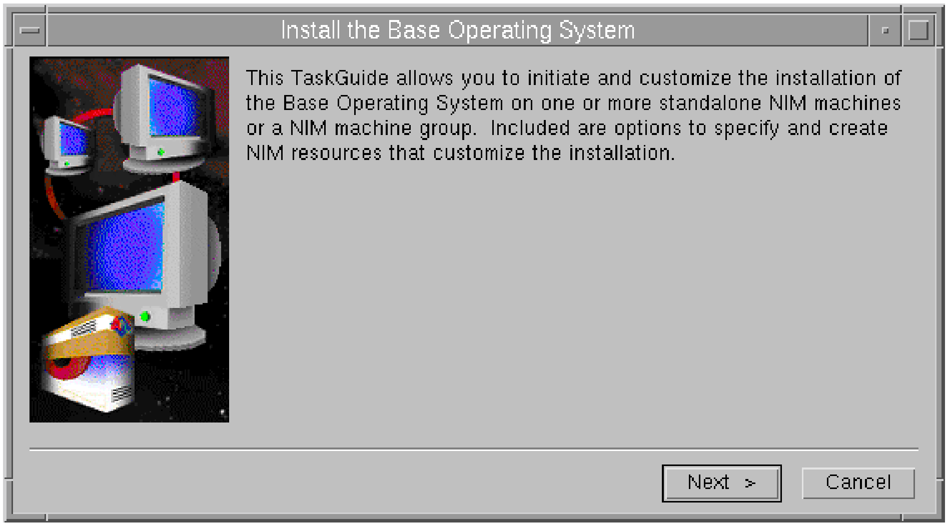
NIM Machines available for group:
433a
433c

Comments

Figure 34. NIM Web-Based System Manager Machine Group Definition

6.14.7.2 Base Operating System Install

Installation of AIX on a machine is now even simpler making use of Web-Based System Manager task guides. Select **NIM** an then **Install Base Operating System** to start the following task guide (Figure 35):



This TaskGuide allows you to initiate and customize the installation of the Base Operating System on one or more standalone NIM machines or a NIM machine group. Included are options to specify and create NIM resources that customize the installation.

Figure 35. NIM BOS Task Guide

The task guide allows you to install multiple machines or, if you have defined any, to install a machine group.

6.15 Paging Space Enhancements (4.3.2)

AIX Version 4.3.2 now provides support for up to 32 GBs of memory on RS/6000 64-bit SMP servers. Before AIX 4.3.2, paging space was allocated for the executing process at the time the memory was requested or accessed. This required backing paging space allocated for all pages in the real memory, to save the image of the page. On a large-memory machine where paging was never or rarely required, these paging space blocks were allocated but never be used. In this case, resources were wasted.

In AIX 4.3.2, the policy for paging space allocation has been modified to allow a deferred paging space allocation. The allocation of paging space is delayed until it is necessary to page out the page, which results in no wasted paging space allocation. This new paging space allocation method greatly reduces the paging space requirements for systems with large physical memory.

6.15.1 Late and Early Paging Space Allocation

There are three kinds of paging space allocation policies used with AIX. The setting of the PSALLOC environment variable determines the paging space allocation mode.

Early Allocation

If the environment variable PSALLOC is set to early, then the early allocation policy is used. This will cause a disk block to be allocated whenever a memory request is made. If there is insufficient paging space available at the time of the request, the early allocation mechanism fails the memory request. This guarantees that the paging space will be available if it is needed.

Late Allocation in Pre-AIX 4.3.2

If the environment variable is not set, then the default late allocation policy is used and a disk block is allocated only when a page in memory is initially accessed, not when it is allocated.

Late Allocation in AIX 4.3.2

AIX 4.3.2 modifies the late allocation policy so that a disk block is not allocated until it becomes necessary to page out the page from memory into paging space. In AIX 4.3.2, late allocation will not allocate any disk blocks if there is enough real memory and no paging required for a given application set.

In summary, Table 27 shows the different policies used in various AIX versions.

Table 27. Paging Space Allocation Policies

PSALLOC = early	All AIX versions	
	Paging space is allocated when memory is *requested*	
PSALLOC is not set or set to any value other than early	Pre-AIX 4.3.2	AIX 4.3.2
	Paging space is allocated when the page in memory is *accessed*	Paging space is allocated when the page in memory needs to be *paged out*

Paging space slots are only released by process (not thread) termination or by the disclaim system call. They are not released by the free call.

6.15.1.1 Early Paging Allocation Mode Considerations

If the PSALLOC environment variable is set to early, then every program started in that environment from that point on, but not including currently running processes, will run in the early allocation environment. The early allocation algorithm causes the appropriate number of paging space slots to be allocated at the time the virtual-memory address range is allocated. For example, with malloc. Interfaces, such as the malloc subroutine and the brk subroutine, will fail if sufficient paging space cannot be reserved when the request is made.

The early allocation algorithm guarantees as much paging space as requested by a memory allocation request. Thus, proper paging space allocation on the system disk is important for efficient operations. When available paging space drops below a certain threshold, new processes cannot be started, and currently running processes may not be able to get more memory. Any processes running under the default late allocation mode become highly vulnerable to the SIGKILL signal mechanism. In addition, since the operating system kernel sometimes requires memory allocation, it is possible to crash the system by using up all available paging space.

Before you use the early allocation mode throughout the system, it is very important to define an adequate amount of paging space for the system. The paging space required for early allocation mode will almost always be greater than the paging space required for the default late allocation mode. How much paging space to define depends on how your system is used and what programs you run. A good starting point for determining the right mix for your system is to define a paging space four times greater than the amount of physical memory.

Certain applications can use extreme amounts of paging space if they are run in early allocation mode. The AIXwindows server currently requires more than 250 MB of paging space when the application runs in early allocation mode. The paging space required for any application depends on how the application is written and how it is run.

6.15.1.2 Late Paging Allocation

If the environment variable PSALLOC is not set, is set to null, or is set to any value other than early, the default late paging space allocation policy is used, and a disk block is allocated only when a page is initially used, not when a memory request is made.

The default late allocation algorithm for paging space allocation assists in the efficient use of disk resources and supports applications of customers who wish to take advantage of a sparse allocation algorithm for resource management.

Some programs allocate large amounts of virtual memory and then use only a fraction of the memory. Examples of such programs are technical applications that use sparse vectors or matrices as data structures. The late allocation algorithm is also more efficient for a real-time, demand-paged kernel, such as the one in the operating system.

6.15.2 Commands Affected by Late Paging

The following commands are affected by the change in paging policy.

6.15.2.1 vmstat Command Update

The avm column reported by `vmstat` command means active virtual pages. In previous AIX versions, the description of avm states "virtual pages are considered active if they are allocated". This is not true for every release of AIX and is changed to "virtual pages are considered active if they have been accessed".

6.15.2.2 lsps Command Update

If you set the environment variable PSALLOC=early, the -s flag displays a value different from the value returned when using the -a flag for all the paging spaces. In this case, the value of -s flag displays the percentage of paging space allocated (reserved), whether the paging space has been assigned (used) or not. The -a flag specifies the percentage of paging space used. Therefore, the percentage reported by the -s flag is usually larger than that reported by the -a flag.

The following is an example. First, set the paging space allocation to early:

```
#export PSALLOC=early
```

After the system is running for some time, the paging space looks like:

```
#lsps -a
Page Space  Physical Volume   Volume Group    Size   %Used  Active  Auto
Type
hd6         hdisk0             rootvg         256MB      8
yes   yes    lv
#lsps -s
Total Paging Space   Percent Used
      256MB               9%
```

The paging space used reported by using -s (9%) is larger than using -a (8%).

Set the paging space allocation to late:

```
#export PSALLOC=
```

The `lsps` command displays the same percentage value with -a and -s flag (8%).

```
# lsps -a
Page Space  Physical Volume   Volume Group    Size   %Used  Active  Auto
Type
```

```
hd6        hdisk0         rootvg        256MB       8     yes   yes    lv

# lsps -s
Total Paging Space   Percent Used
      256MB              8%
```

6.16 Error Message Templates (4.3.2)

When an application wants to log an error to the AIX error log, it writes information about the error, specifically the error identifier, resource name, and error specific data to the /dev/error special file. The error daemon reads from the special file and logs the information in the error log. The `errpt` command, which is used to display error messages, reads the error message template repository to determine how to interpret and display the error data.

On previous versions of AIX, it is not possible for an error message template definition to contain any message text. It can only contain codepoints, which are two byte message numbers used to reference predefined message strings. The codepoints refer to messages defined by the IBM SNA Generic Alert Architecture described in *SNA Formats*, GA27-3136. The architecture imposes restrictions on the message numbering and message length that can be used. The message strings are kept in the codepoint.cat file, which is a specially formatted message catalog.

AIX 4.3.2 has updated the `errpt` command, along with the commands used for updating the error template repository, to understand an additional error template format, which can define messages from a normal format NLS message catalog as well as using the previous codepoint method.

A template may contain all NLS messages, all codepoints, or a combination of both. An NLS message is represented with a message set number, a message number, and a default text string to be printed if the associated message catalog is not present. Each error template also specifies the message catalog to be used for messages referenced by that template.

```
*!sample.cat
* sample error template using NLS messages
+ SAMPLE:
        Err_Type = UNKN
        Class = S
        Report = TRUE
        Log = TRUE
        Alert = FALSE
        Comment = "Sample of msgs in templates"
        Err_Desc = {1, 2, "SAMPLE DESCRIPTION TEXT"}
        Fail_Causes = {0, 0xeb54, ""), (1, 5, "default cause"},
           {3, 5, "default cause 2"}, {3, 6, "default cause3"}
        Prob_Causes = {2, 1, "Bad Operator"}, {2, 2, "Bad Programmer"}
        User_Causes = {2, 5, "User pressing wrong key"}
        User_Actions = {2, 6, "Read the manual"},
           {2, 7, "Take a long long long\n\t\
long, really long, look at the manual."}
        Inst_Causes = {0, 0, ""}
        Inst_Actions= {2, 8, "reinstall"}
        Fail_Actions = {2, 9, "kick it"}
        Detail_Data = 226, {0, 0x8004,""} ,ALPHA
        Detail_Data = 4, {2, 10, "regester value"} ,HEX
```

This dramatically increases the number of messages that can be used by an error template, as it is no longer restricted to the messages defined by the IBM Alert Architecture. The trade off is that the new error messages are no longer changeable.

In addition to increasing the number of messages available to an error template, this enhancement has also increased the number of detail data items up to a maximum of sixteen.

6.17 Remote File Distribution Enhancements (4.3.2)

Previous versions of AIX included Version 5.1 of the `rdist` command, which is used to distribute and maintain identical copies of files on multiple hosts. The `rdist` command on AIX 4.3.2 has been updated to Version 6.1.3, which includes some new features, namely:

- Multiple target hosts are now updated in parallel. This improves the update time when working with large numbers of hosts. This behavior can be controlled by changing the number of hosts updated in parallel, or disabling the feature, in which case, hosts are updated sequentially.
- The new version of `rdist` avoids problems when communicating with a remote host by setting a time-out value. If the remote host fails to respond within a set period during a transfer, `rdist` displays an error message and continues to update other hosts. The previous version of `rdist` would continue to wait until the remote host responded.
- Local and remote error messages are distinctly marked for better clarity.
- The amount of free space can optionally be checked to avoid filling up a file system. Before actually installing or updating a file, `rdist` will calculate whether the update would exceed the minimum amount of free space as specified on the command line. If the minimum space would be exceeded by the update, no update is performed and an error message is displayed.
- The client and server portions are split into two distinct programs, `rdist` and rdistd. This lowers the risk of security vulnerabilities since the server rdistd does not need to be setuid to root. It also allows for greater ease in maintaining different versions of `rdist`.

Version 6.1 of `rdist` implements a new protocol for communicating between machines. Both versions of the `rdist` command are shipped with AIX 4.3.2 to allow users to distribute files to machines running either version of `rdist`. The new version is shipped as `/usr/bin/rdist`, and the old version as `/usr/bin/oldrdist`.

When the `rdist` program contacts a target machine, it requests the target to start the `rdist` server side program. Version 6.1 `rdist` will start the rdistd server program. Version 5.1 `rdist` requests the target machine to run `rdist -Server`. If the version 6.1 `rdist` is run with the -Server option, then it will exec a copy of `oldrdist`. In this way, you can get compatibility with hosts running `rdist` Version 5.1 attempting to distribute files to a machine running `rdist` Version 6.1. If a host running `rdist` Version 6.1 wants to distribute files to a host running the `rdist` Version 5.1, then it must run the `oldrdist` program.

6.18 Editor Enhancements (4.3.2)

The `ed` editor program has been enhanced to examine the environment variable EDTMPDIR to determine the directory location for temporary files. This has been done to allow a system to better handle the start up of multiple `ed` sessions by avoiding a bottleneck on the inode for the default temporary directory used by `ed`.

6.19 System Backup Usability Enhancements (4.3.2)

The `mksysb` and `savevg` commands have been enhanced to include information in the backup image about block size of the tape device being used to store the backup image.

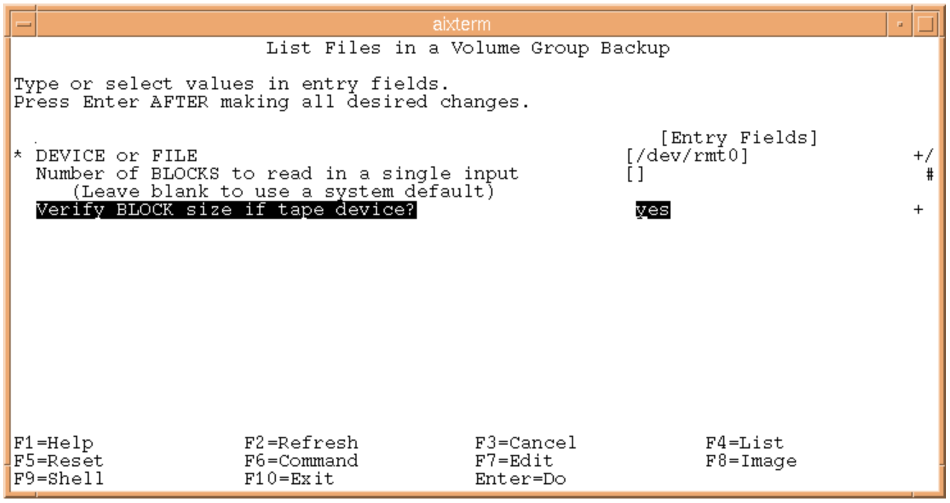
This change means that when using SMIT to list the contents of the backup, restore individual files, or restore the complete backup. The system can read the information on the tape indicating the block size used to create the image. It can then change the block size of the tape device being used to read the backup to be the same, therefore maximizing the data transfer rate. Once the operation to list, or restore the backup has completed, the system changes the block size of the tape device back to the previous setting.

The following SMIT panels have been updated to include a new option that allows the user to specify whether the system should attempt to determine the tape block size used to create the backup image:

- List Files in a System Image
- Restore Files in a System Image

- List Files in a Volume Group Backup
- Restore Files in a Volume Group Backup

An example of the change is shown in Figure 36. If the option is set to yes, and the tape being read was created on a previous version of AIX that did not include the tape block size information on the tape, then the underlying commands will set the block size to 0 and continue with the required operation.



```
                       List Files in a Volume Group Backup

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

                                                           [Entry Fields]
* DEVICE or FILE                                        [/dev/rmt0]             +/
  Number of BLOCKS to read in a single input            []                       #
     (Leave blank to use a system default)
  Verify BLOCK size if tape device?                      yes                    +









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

Figure 36. Sample SMIT Volume Group Backup Screen

6.20 Operating System Install Enhancement (4.3.2)

The function of the non-prompted install method has been improved to provide a means of protecting user defined volume groups already on the system. A non-prompted install can be carried out by supplying a customized bosinst.data file when restoring a mksysb or installing the base operating system from CD-ROM, tape, or NIM server.

The EXISTING_SYSTEM_OVERWRITE variable now has three possible values, provided in Table 28, which determine the action taken.

Table 28. Possible Values of EXISTING_SYSTEM_OVERWRITE

EXISTING_SYSTEM_OVERWRITE=	Action Taken
any	Any disk can be used for the system install. This is the behavior of the 'yes' option on releases prior to AIX 4.3.2.
no	Only disks containing no volume groups (user defined or previous rootvg) can be used.
yes	Only disks in the current rootvg, or containing no volume groups, can be used for the system install

The value of EXISTING_SYSTEM_OVERWRITE is only examined if the bosinst.data file also sets PROMPT=no and INSTALL_METHOD=overwrite.

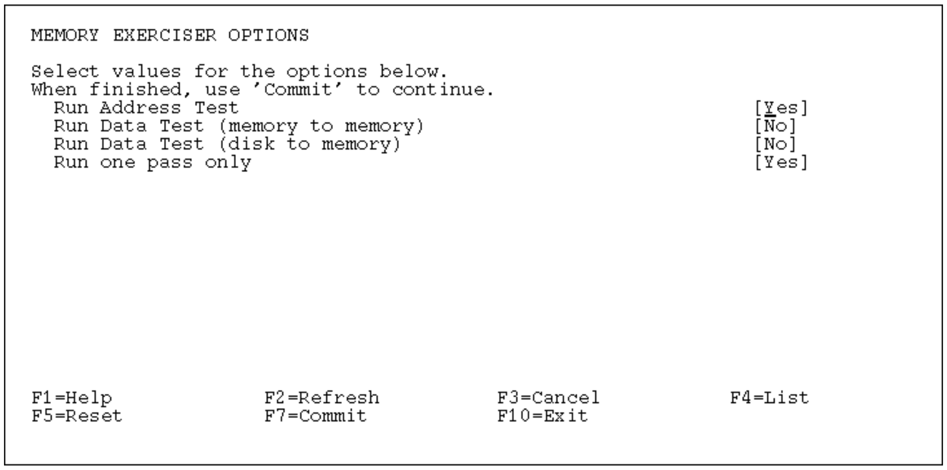
6.21 New Diagnostic Service Aid (4.3.2)

The diagnostics subsystem in AIX 4.3.2 has been enhanced by the addition of a system memory exerciser, which can be used to check system memory on CHRP systems. The tool is implemented as a service aid and is only available when running online diagnostics in service or maintenance modes. Service mode diagnostics are entered when the machine boots in service mode. Maintenance mode diagnostics are entered by first taking the machine to maintenance mode using the `shutdown -m` command.

The machine to be tested is required to have the bos.acct package installed and paging space of at least one and a half times the amount of physical memory. The service aid will exit with an error if the bos.acct package is not installed. If the paging space requirement is not met, the service aid will give a warning message informing the user that it is not possible to test the maximum amount of physical memory.

The service aid will then display the memory exerciser options screen, which allows the user to select the characteristics of the testing that will be performed.

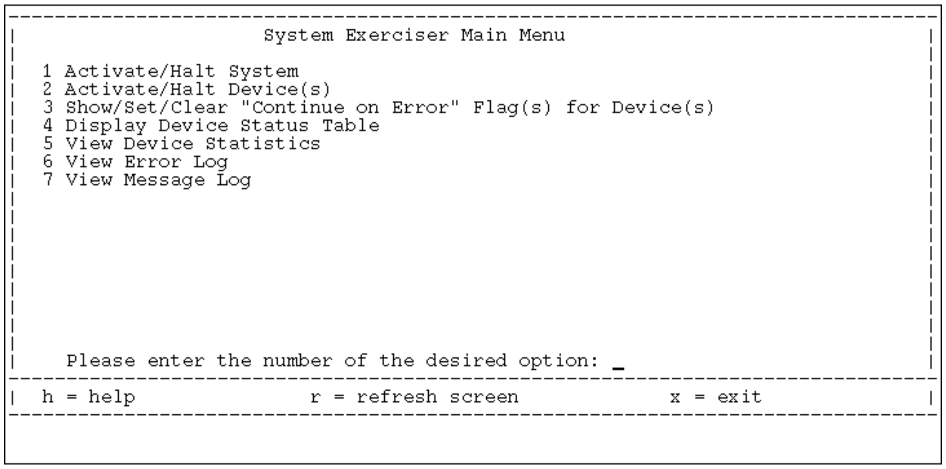
The service aid memory exerciser options screen is shown in Figure 37.



```
MEMORY EXERCISER OPTIONS

Select values for the options below.
When finished, use 'Commit' to continue.
  Run Address Test                                              [Yes]
  Run Data Test (memory to memory)                              [No]
  Run Data Test (disk to memory)                                [No]
  Run one pass only                                             [Yes]










F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F7=Commit           F10=Exit
```

Figure 37. Memory Exerciser Options Menu

Once the desired options have been selected, the system exerciser menu (Figure 38) is displayed, which allows the user to start and stop the exerciser and view the error logs.



```
--------------------------------------------------------------------------------
|                      System Exerciser Main Menu                              |
|                                                                              |
|  1 Activate/Halt System                                                      |
|  2 Activate/Halt Device(s)                                                   |
|  3 Show/Set/Clear "Continue on Error" Flag(s) for Device(s)                  |
|  4 Display Device Status Table                                               |
|  5 View Device Statistics                                                    |
|  6 View Error Log                                                            |
|  7 View Message Log                                                          |
|                                                                              |
|                                                                              |
|                                                                              |
|                                                                              |
|                                                                              |
|                                                                              |
|                                                                              |
|    Please enter the number of the desired option: _                          |
--------------------------------------------------------------------------------
|  h = help               r = refresh screen             x = exit              |
--------------------------------------------------------------------------------
```

Figure 38. System Exerciser Main Menu

6.22 Diagnostic Enhancements (4.3.3)

This section describes the enhancements to the diagnostic environment that are provided as part of AIX Version 4.3.3. For background information regarding this environment, you should review the book *Understanding the Diagnostic Subsystem for AIX*, SA23-2797 which has also been upgraded.

There is a new diagnostic CD-ROM for AIX Version 4.3.3.

6.22.1 Diagnostics Task Selection List

Previously, the diagnostic Tasks Selection List menu listed items in the order in which it was thought that they would be used, but this has proven to be difficult to use. With AIX Version 4.3.3, tasks and service aids under diagnostics have now been ordered alphabetically for improved ease of use.

You can see the new display order by using the steps that follow:

1. Go to the main diagnostic menu. You can execute `smitty`, select **Problem Determination**, then **Hardware Diagnostics**, then **Current Shell Diagnostics**, or the fast path `diag` command, and then press the Enter key to continue to the screen from which you can select the option in the next step.
2. Select the option **Task Selection(Diagnostics, Advanced Diagnostics, Service Aids, etc.)** to display the list of tasks on your screen, which should appear as in the following example:

```
TASKS SELECTION LIST                                                  801004


From the list below, select a task by moving the cursor to
the task and pressing 'Enter'.
To list the resources for the task highlighted, press 'List'.

[TOP]
  Run Diagnostics
  Run Error Log Analysis
  Run Exercisers
  Display or Change Diagnostic Run Time Options

  Add Resource to Resource List
  Backup and Restore Media
  Certify Media
  Change Hardware Vital Product Data
  Configure ISA Adapter
  Configure Reboot Policy
  Configure Remote Maintenance Policy
[MORE...30]

F1=Help             F4=List             F10=Exit            Enter
F3=Previous Menu
```

The choices that follow on subsequent screens are now listed in alphabetical order. The tasks are discussed in the article *Tasks and Service Aids* in the book *Understanding the Diagnostic Subsystem for AIX*, SA23-2797.

Documentation of Tasks

The AIX Documentation that describes the diagnostic tasks is currently task oriented. The *Tasks and Service Aids* article has grouped the tasks in the article's *Task List* section in one of the following six categories:

- Diagnostics and Error Log Analysis
- Configuration and VPD
- Communications and LANs
- Media
- Microcode
- Displays

Each task has a hypertext link to text that describes what the task is. The descriptions of every task are located following the list of all tasks. The descriptions are listed alphabetically, which is also how they appear on the screen as shown for the output of step 2 on page 277.

As shown in that screen, the **Change Hardware Vital Product Data** task follows the **Certify Media** task. The *Tasks and Service Aids* article categorizes the Certify Media task in the Media group of tasks, and the Change Hardware Vital Product Data task is in the Configuration and VPD group. If you select the link for either task, you will go to its description. The text description of the Change Hardware Vital Product Data task follows that of the Certify Media task, even though they are in separate categories.

6.22.2 New Diagnostic Event Log

With the enhanced AIX diagnostic package, you can keep track of diagnostic activity on the systems by using a new diagnostic event log. This log can be viewed by using the **Display Previous Diagnostic Results** task. Previously, the results of diagnostic tasks may not have been clear. That is, Service Request Numbers (SRNs) may not have been noticed amongst any No Trouble Found (NFT) messages; before AIX Version 4.3.3, the previous results task displayed at most 25 entries.

The diagnostic log is a tasks log that is stored in binary in the new log file /var/adm/ras/diag_log, not the errlog file. The Diagnostic Controller also writes its analysis to files in the directory /etc/lpp/diagnostics/data, and the `diagrpt` command, or Display Previous Diagnostic Results task, can be used at a later date to retrieve and display these results. If you look in the /etc/lpp/diagnostics/data directory, you can see the last 25 diagnostic results (NTFs, SRNs, and menugoals) in files named diagrptX.dat where X is a number from 1 to 25. The `diagrpt` command can be used to look at either the *.dat files or the diag_log. You can look at the diag_log directly by using the -a (detail version) or -r (summary version) flags. The -o and -s flags can be used to view the *.dat files directly. Some examples of the `diagrpt` command are shown in “Using the Enhanced diagrpt Command” on page 281. The files in /etc/lpp/diagnostics/data/*.dat are ASCII data files. When there are more than 25 entries, the files are overwritten. Therefore, you will always have the last 25 diagnostic results (NTFs, SRNs, and menugoals).

The same diagnostic results are also logged to /var/adm/ras/diag_log which is a binary file. However, unlike the *.dat files which can only show the last 25 results, the diagnostic log can show you a lot more diagnostic results. Since the diag_log is at least 100K it can hold more than 25 results before the data is wrapped.

Note that you should not use the binary form of the log data, but only the data formatted and presented to you by the `diagrpt` command, in case the format of the binary log changes.

6.22.2.1 Inside the Diagnostics Log

Before AIX Version 4.3.3, the Display Previous Diagnostic Results task displayed at most 25 entries. Now this task allows you to access the contents of the new diagnostic event log ranging from 100 KB to 1 MB. This log can be used to trace the execution of diagnostics. The following events are logged:

- Starting diagnostics
- All problem reports (SRNs)
- All NTFs
- All menu goals
- All software errors
- Exerciser errors

The log entries are similar to the AIX Error Log. They can be displayed in either a short or long version. The command `diagrpt -a` displays the long version of the log. The short version, using the command `diagrpt -r`, has the following output format:

```
ID DATE/TIME  T RESOURCE_NAME  DESCRIPTION
```

The Date/Time field has the following form: weekday month day HH:MM:SS. For example, an entry may be: Mon Jan 5 11:25:30. This is the same time stamp format as is used in the long version of the `errpt` command.

Log Identifier

The ID is a hex value that identifies the event being logged when viewing the short version of the log. The current log identifiers are:

- DC00 = Diagnostic Controller session started
- DCF0 = Diagnostic Controller reported an SRN from missing options
- DCF1 = Diagnostic Controller reported an SRN from new resource
- DCE1 = Diagnostic Controller reported ERROR_OTHER
- DA00 = Diagnostic Application reported NTF
- DAF0 = Diagnostic Application reported an SRN
- DAFE = Diagnostic Application reported an ELA SRN
- DAE0 = Diagnostic Application reported ERROR_OPEN
- DAE1 = Diagnostic Application reported ERROR_OTHER

Log Types

Each log entry has a *Type* (T field) associated with it. The following Types are logged:

- Type I = Informational Message
- Type S = SRN Callout
- Type N = No Trouble Found
- Type E = Error Condition

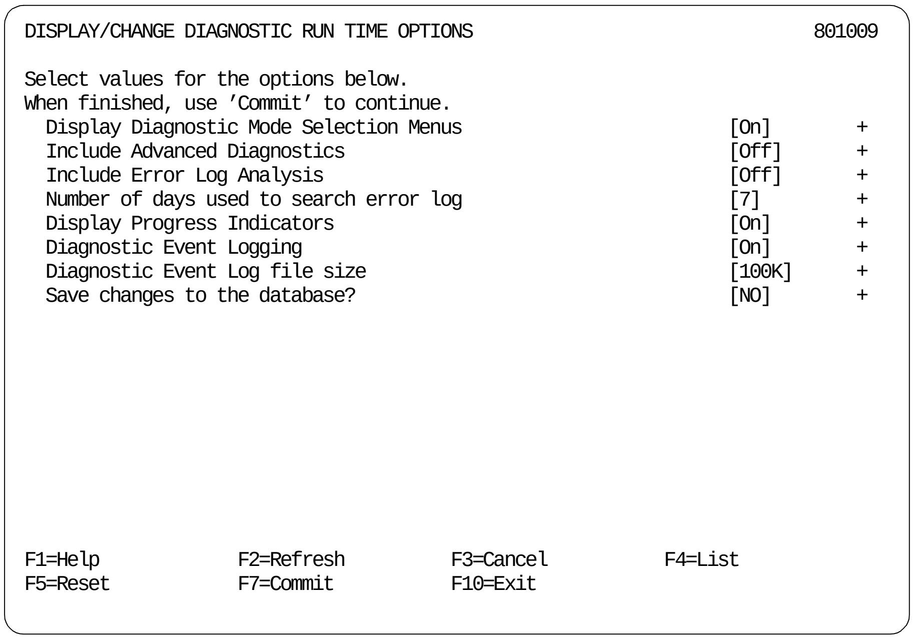
6.22.2.2 Log Options

The log has a toggle option to start or stop logging diagnostic events. You can also control the size of the log. After the maximum size of the log is reached, the log will wrap, overwriting previous events. To support multiple sessions of Diagnostics, the log is locked using the ODM lock routines before events are written to it. The log only logs events in online concurrent or online service mode, and is turned off when running from standalone diagnostics.

You can change the log options by the following method:

1. Go to the main diagnostic menu. You can execute `smit`, select **Problem Determination**, then **Hardware Diagnostics**, then **Current Shell Diagnostics**, or the fast path `diag` command and then press the **Enter** key to continue to the screen with the usual menu choices.

2. Select the option **Task Selection(Diagnostics, Advanced Diagnostics, Service Aids, etc.)** to display the list of tasks.
3. Select the task **Display or Change Diagnostic Run Time Options** to get to a screen that looks similar to the following example:



```
DISPLAY/CHANGE DIAGNOSTIC RUN TIME OPTIONS                            801009

Select values for the options below.
When finished, use 'Commit' to continue.
  Display Diagnostic Mode Selection Menus                 [On]        +
  Include Advanced Diagnostics                            [Off]       +
  Include Error Log Analysis                              [Off]       +
  Number of days used to search error log                 [7]         +
  Display Progress Indicators                             [On]        +
  Diagnostic Event Logging                                [On]        +
  Diagnostic Event Log file size                          [100K]      +
  Save changes to the database?                           [NO]        +









F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F7=Commit           F10=Exit
```

4. Make the required changes to the options Diagnostic Event Logging and Diagnostic Event Log file size.
5. Press the **F7** key to save your changes and exit this screen.

6.22.2.3 Using the Enhanced diagrpt Command

The `diagrpt` command is shipped in the bos.diag.util fileset, and its syntax is:

```
diagrpt [[-o][-s mmddyy]] | [ [-a]|[-r] ]
```

Previously, the syntax of the `diagrpt` command was:

```
usage: diagrpt [ -o ][ -s mmddyy ]
```

The following examples show the use of the `diagrpt` command:

- The short version of the command, using the -r flag, displays one event per line. You can use this with the `head -5` command to display the five most recent entries, as shown:

```
# /usr/lpp/diagnostics/bin/diagrpt -r|head -5
```

```
ID     DATE/TIME            T RESOURCE_NAME    DESCRIPTION
DC00   Wed Aug 11 09:59:51  I diag             Diagnostic Session was started
DA00   Wed Aug 11 04:00:52  N sysplanar0       No Trouble Found
DC00   Wed Aug 11 04:00:48  I diag             Diagnostic Session was started
DA00   Wed Aug 11 04:00:45  N sysplanar0       No Trouble Found
```

• The -o flag can be used, as in the AIX 4.3.2 release, to display some more detail about what the most recent diagnostic activity was:

```
# /usr/lpp/diagnostics/bin/diagrpt -o
TESTING COMPLETE on Wed Aug 11 04:00:52 CDT 1999
801010

No trouble was found.

The resources tested were:

- sysplanar0       00-00           CPU Planar
```

• You can use the wc command to show that the log contains 523 entries:

```
# /usr/lpp/diagnostics/bin/diagrpt -r|wc -l
     523
```

• You can use the long version, invoked with the -a flag, to display all information contained in each event log entry as in the (different) example:

```
# diagrpt -a
---------------------------------------------
IDENTIFIER:     DC00

Date/Time:                Mon Jan  5 11:25:30
Sequence Number:          1
Event Type:               Informational Message

Resource Name:            diag
Diag Session:             23781

Description:
Diagnostic Session was started.
---------------------------------------------
IDENTIFIER:     DAF0

Date/Time:                Mon Jan  5 11:25:30
Sequence Number:          2
Event Type:               SRN Callout

Resource Name:            fd0
Resource Class:           diskette
Resource Subclass:        siofd
Resource Type:            fd
```

```
Location:               01-D1-00-00

Diag Session:           23781
Test Mode:              Console, Advanced, Normal
Problem Determination, Option Checkout

SRN:                    935-102

Description:
Diskette Drive select test failed.

Probable FRUs:
80% fd0               00-00-0D-00       Diskette Drive
20% sysplanar0        00-00             System Planar
----------------------------------------------
```

6.22.3 Diagnostic Test Enhancements

Some improvements have been made to the system hardware tests. These include:

- The FRU part number is now displayed instead of the FRU confidence level, if available. This reduces the amount of cross reference lookup by a CE to get the correct replacement part.
- A restructure of the Central Electronics Complex (CEC) tests has been done, to improve the analysis of microcode generated error logs.
- The Operator Panel tests have been placed in their own Diagnostic Application. Previously, the Operator Panel tests were included in the system planar Diagnostic Application. Normally, the system planar diagnostics are run to test and do error log analysis for system problems. The Operator Panel Test just had previously added unneeded menus that you had to answer while investigating such problems. Since this has been changed, the system planar can be tested more quickly. An entry for the Operator Panel now appears on the Diagnostic Selection and Resource Selection menu.

6.22.4 Customer Engineer Diagnostic Login Capability

In previous versions of AIX, only the root user could run diagnostics. The CE diagnostic login capability will allow a Customer Engineer (CE) or service support representative to login to a system as non-root user and run diagnostics.

An administrative role has been created that allows diagnostics to be run. Diagnostics authenticates the user by checking if the user has this administrative role. Users in the Run Diagnostics role can change the system configuration, update microcode, for example.

6.22.4.1 Creating a Customer Engineer User

You must either have root user authority or be a user with administrative role *RunDiagnostics* to run diagnostics. If you are not a root user, you must also have *system* as primary group. You can create a user with the appropriate authority by executing the command `smitty mkuser` and entering the minimally required fields, as in the following example:

```
 Add a User

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

[TOP]                                                  [Entry Fields]
* User NAME                                         [diagce]
  User ID                                           []                      #
  ADMINISTRATIVE USER?                               false                 +
  Primary GROUP                                     [system]               +
  Group SET                                         []                     +
  ADMINISTRATIVE GROUPS                             []                     +
  ROLES                                             [RunDiagnostics]       +
  Another user can SU TO USER?                       true                  +
  SU GROUPS                                         [ALL]                  +
  HOME directory                                    []
  Initial PROGRAM                                   []
  User INFORMATION                                  []
  EXPIRATION date (MMDDhhmmyy)                      [0]
  Is this user ACCOUNT LOCKED?                       false                 +
[MORE...36]

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

If a user does not have the appropriate authority, they will receive an error message similar to the following if they try to access the system diagnostic tools:

```
$ diag
User is not authorized to run diagnostics.
```

6.22.5 Hardware Diagnostic Exerciser

A diagnostic exerciser has been added for processors to enhance problem determination. It provides a means to verify both memory and processor repairs previously detected by error log analysis.

The exercisers are used to test hardware and verify part replacement. Fatal hardware errors and data mis-compares are reported by the exercisers. Recoverable errors are not reported by the exercisers. Fatal errors are logged in the AIX error log and data mis-compare errors are logged in the diagnostics event log. Recoverable errors may be logged in the AIX error log if thresholds are exceeded. Error Log Analysis should be run to determine a SRN. Mis-compare errors should be rare. Note the following important points:

- The exercisers are only supported on CHRP platforms.
- In general the exercisers will try to verify the hardware by writing data patterns to the device, reading it back and comparing the data written with the data read.
- The exercisers are not supported from standalone diagnostics.
- 100% of the real memory will not be exercised since memory access is through the AIX Virtual Memory Manager (VMM). The `vmstat` and `lsps` commands are used to determine how much memory and paging space is available and then that amount of memory is allocated for use by the exerciser using the IPC shared memory facility.
- It is recommended that no customer applications be running while the systems exerciser is running since the system performance will be degraded.
- You need 1 MB of free storage in /tmp. You can use `smit` to increase the size of the /tmp file system.
- If there are no free physical partitions on the hard disk you select, the exercise will be limited to *read* only. If there are free partitions, you are given the option to set up a scratch logical volume to perform a read/write/compare exercise. The default is one logical partition which in most cases is sufficient.

Exerciser Options and Duration

There are two exerciser options:

- Extended

  If the extended exercise option is selected the exerciser main menu is displayed and once the exercisers are started they will run continuously until you choose to terminate the exercise.

- Short

  The short exercise will take 5-10 minutes depending on processor speed, memory, and I/O configuration. For the short exercise the Exerciser Main Menu screen is bypassed, the exercisers started automatically, and the status screen is displayed. When the exercise completes without error(s) an `Exerciser complete, Run Error Log Analysis` message is displayed, and when you respond by pressing Enter, the Task Selection Menu is displayed.

If error(s) are detected, you have the option to further investigate the error(s) before returning to the Task Selection Menu.

6.23 Performance Toolbox Agent Repacking (4.3.2)

The Performance Toolbox is a Motif-based AIX licensed program product (LPP) that consolidates AIX performance tools into a toolbox framework. Users can easily access tools for system and network performance tuning, monitoring, and analysis. It consists of two major components: Performance Toolbox Manager and Performance Toolbox Agent.

The Performance Toolbox Manager has three packages:

perfmgr.local	This package contains the commands and utilities that allow monitoring of only the local system.
perfmgr.network	This package contains the commands and utilities that allow monitoring of remote systems as well as the local system.
perfmgr.common	This package contains the commands and utilities that are common between the network support and the local support.

The Performance Toolbox Agent has one package:

perfagent.server	This package contains the performance agent component required by Performance Toolbox as well as some local AIX analysis and control tools.

The packaging of the previous Performance Toolbox contained two filesets: perfagent.server and perfagent.tool causing installation difficulty. To improve this process, those pieces that are required to be built with the AIX kernel are moved into the perfagent.tools fileset. Then the agent becomes mainly an interface routine to those pieces.

The perfagent.tools fileset is shipped with the AIX 4.3.2 base. For AIX 4.3.2, The Performance Toolbox Agent will prereq perfagent.tools. So the .tools fileset must be installed first.

Note: The online *PTX Guide and Reference for AIX 4.3* (perfagent.html.en_US) is available by ordering APAR IX80484 (PTF

U458736). In AIX Version 4.3.0, this document is shipped with the Performance Aide CD. In AIX Version 4.3.2, it is included in the base AIX documentation.

Table 29 lists the various minimum file set levels required with a particular AIX level.

Table 29. AIX Level and Required File Sets

AIX Version	File Set
AIX 4.1.5	perfagent.tools 2.1.6.* perfagent.server 2.1.6.*
AIX 4.2.1	perfagent.tools 2.2.1.* perfagent.server 2.2.1.*
AIX 4.3.1	perfagent.tools 2.2.31.* perfagent.server 2.2.31.* (replaced by 32)
AIX 4.3.0	perfagent.tools 2.2.32.0 perfagent.server 2.2.32.0 (prereqs 3.3.32.0 perfagent.tools)

6.24 Performance Toolbox Enhancements (4.3.3)

On AIX 4.3.3 following performance toolbox enhancements are provided.

6.24.1 Kernel Statistics Access From User Mode

Sometimes it is useful to access kernel statistics information from non-privileged processes. An example of such process is the database manager process that requires kernel statistics information for its own resource management. A new kernel extension, libspmi_kex, is introduced to allow user mode process to access kernel statistics. The libspmi_kex kernel extension supplies functions to open /dev/kmem and /dev/mem for binary read. The functions are used only by the functions in libspmi.a shared library. Processes can access kernel statistics information through libspmi.a shared library.

6.24.2 SPMI-Based Top Clone

A new performance monitoring program based on libspmi.a shared library is introduced. The program, `topas`, is a cursed based application that displays *top* processes, disk, and memory usage. As its name implies, `topas` program is a clone of the freeware program `top`. The following screen output shows screen images of `topas`.

```
Topas Monitor for host:    rootmaster          EVENTS/QUEUES    FILE/TTY
Thu Aug 12 19:13:48 1999   Interval:  2        Cswitch     483  Readch     1178
                                               Syscall    4781  Writech     197
Kernel    6.2   |##                          | Reads         6  Rawin         0
User      0.2   |                            | Writes        0  Ttyout      197
Wait     44.5   |##############              | Forks         0  Igets         0
Idle     49.0   |###############             | Execs         0  Namei      2300
                                               Runqueue    0.0  Dirblk     2627
Interf   KBPS   I-Pack  O-Pack   KB-In  KB-Out Waitqueue   1.7
tr0       0.3     2.0     0.5     0.1     0.2
lo0       0.0     0.0     0.0     0.0     0.0  PAGING           MEMORY
                                               Faults      325  Real,MB     511
Disk    Busy%     KBPS     TPS KB-Read KB-Writ Steals        0  % Comp     28.0
hdisk0    0.0    894.0   223.5   894.0     0.0 PgspIn        0  % Noncomp  20.0
hdisk1    0.0      0.0     0.0     0.0     0.0 PgspOut       0  % Client    0.0
hdisk5    0.0      0.0     0.0     0.0     0.0 PageIn      223
hdisk3    0.0      0.0     0.0     0.0     0.0 PageOut       0  PAGING SPACE
hdisk2    0.0      0.0     0.0     0.0     0.0 Sios        223  Size,MB     288
                                                                % Used      0.6
find     (18398) 10.0% PgSp: 0.1mb root                         % Free     99.3
topas    (26004)  0.5% PgSp: 0.4mb root
gil      (1548)   0.0% PgSp: 0.0mb root
ksh      (24814)  0.0% PgSp: 0.2mb root           Press "h" for help screen.
X        (4464)   0.0% PgSp: 2.9mb root           Press "q" to quit program.
```

6.24.3 Performance Toolbox for AIX (PTX) Scaling

The Performance Toolbox Agent (PTX) has been upgraded with the ability to process performance metrics by activity, rather than by a fixed name. This capability allows system metrics to be logged only when your specified thresholds are exceeded. Another improvement to PTX is the ability to create a file which contains shell commands to be executed when recording files are deleted. This feature allows you to merge, rename or move recording files automatically.

The process table size internal to the System Performance Measurement (SPMI) Library was previously limited to 100 processes, but this has been expanded. The PTX agent can initialize contexts for more than 256 processes in 4.3.3, but you cannot monitor all of them at the same time. Remote monitoring is usually limited to less than 150 simultaneous metrics. The SPMI library internals and metric definitions now uses the procsinfo structure instead of the procinfo and userinfo structures.

6.25 Performance Related Enhancements (4.3.3)

There are a number of AIX performance enhancements. Some of the changes, such as the consolidation of common code in the `netpmon`, `tprof`, and `filemon` tools into one common library, may not have a significant impact on how you use these tools. However, other tools have had usability enhancements also and so these changes are described in this section.

6.25.1 sar Command: New -d Flag

AIX 4.3.3 has added an additional option to the `sar` command: the -d option provides useful statistics such as throughput, average queue depth, and so on. Many of these statistics were previously being provided with the `iostat` command. For compatibility with other UNIX systems, this information is also being provided with the `sar` command. The following example uses the find command to create some disk activity. The `sar` interval parameter used here is two seconds and the number parameter is three. The following example shows the output of new -d flag; disk I/O data is shown three times at two second intervals.

```
# find / -ls >/dev/null &
[1]     4494
# sar -d 2 3

AIX nisclient 3 4 006152004C00    08/22/99

19:09:17     device    %busy    avque    r+w/s   blks/s   avwait   avserv

19:09:19     hdisk0      0      0.0        0        0      0.0      0.0
                cd0      0      0.0        0        0      0.0      0.0


19:09:21     hdisk0     61      1.0       90      358      0.0      0.0
                cd0      0      0.0        0        0      0.0      0.0


19:09:23     hdisk0     70      0.0       95      382      0.0      0.0
                cd0      0      0.0        0        0      0.0      0.0


Average      hdisk0     43      0.3       61      246      0.0      0.0
                cd0      0      0.0        0        0      0.0      0.0
#
```

6.25.2 Feedback Directed Program Restructuring

Feedback Directed Program Restructuring (FDPR) is an optimization tool that improves program code locality. The tool receives input files in XCOFF format, instruments them, executes them for profiling information and then reorders them in order to get a better cache ratio. In AIX 4.3.3, the usability and reliability of this tool has been improved by adding automatic selection of options based on the type of execution used.

The FDPR tool contains many optimization options which you can choose for optimizing a program. Some of the options are essential for certain applications. For example, if you had a C++ application which used the Throw and Catch mechanism, then you had to use the -tb fdpr flag to preserve trace back tables in order for the reordered code to run properly. Now -tb can be omitted and the trace back tables will be automatically included.

6.25.3 svmon Performance Tool Enhancements

The svmon tool has been enhanced with usability, scalability and speed improvements. Additionally, it supports the new Workload Management function offered in AIX Version 4.3.3.The svmon tool is part of the perfagent.toolspackage, which contains many performance analysis tools.

An updated svmon man page is in B.2, “svmon” on page 711.

6.25.3.1 What does svmon do

The AIX operating system manages lots of entities, such as processes, files, and users, that consume virtual memory resources. The goal of svmon is to provide you with a snapshot of Virtual Memory consumption of some entities. An application developer can use svmon’s statistics to optimize an application, or an administrator may be able to identify a problem with an application. VMM skill is important to fully interpret and utilize svmon output data.

The virtual memory consumption is expressed by means of:

- The number of pages in real memory (and the pinned pages herein)
- The number of pages reserved on paging space
- The number of pages allocated in real memory or on paging space

6.25.3.2 New svmon Filter Options

The svmon man page describes all the flags and also the different types of reports that svmon can generate. By default, all segments are processed, but you can use the following new flags to filter the analyzed segment:

-w	Only working segments are analyzed
-f	Only persistent segments are analyzed
-c	Only client segments are analyzed

A new flag will sort the entities in decreasing order.

-v	Sort the entities by number of pages in virtual space

During the detailed report (-D), the option -b can be used for debugging purposes. This can display frame information of all frames of the segment (page number, frame number, pinned status, modified bit status, and reference bit status). An example of its use, assuming you want detailed statistics for segment 11211, is:

```
# svmon -D 11211 -b|pg

Segid: 11211
Type:  persistent
Address Range: 0..0

           Page      Frame    Pin        Ref        Mod
              0     151167     N          N          N
#
```

Assuming that you had used the ps and grep commands to determine that the process ID of the cron command is 10338, you could display the working segments in uses as in the following example:

```
# svmon -P 10338 -w

-------------------------------------------------------------------------------
     Pid Command        Inuse      Pin     Pgsp  Virtual   64-bit    Mthrd
   10338 cron            3624     1426      178     2621        N        N

  Vsid     Esid Type Description           Inuse   Pin Pgsp Virtual Addr Range
     0        0 work kernel seg             2701  1425  178  2431   0..32767 :
                                                                    65475..65535
  9029        d work shared library text     747     0    0    40   0..65535
  1181        2 work process private         125     1    0   125   0..46 :
                                                                    65309..65535
 1f1ff        f work shared library data      51     0    0    25   0..109
#
```

6.25.3.3 New svmon Report Displays

The svmon command now can produce seven types of reports. The three new types are:

-W to supports workload management.This option displays memory usage statistics for the specified workload management class name(s).

-U to display the memory consumption of a user

-C to display the memory consumption of a command

As an example of these new flags, the following command shows how you no longer have to find cron's PID, but you can quickly obtain a snapshot of its memory usage in one step:

```
# svmon -C cron

===============================================================================
Command cron            Inuse      Pin     Pgsp  Virtual
                         3645     1426      178     2621

...............................................................................
SYSTEM segments         Inuse      Pin     Pgsp  Virtual
                         2701     1425      178     2431

  Vsid     Esid Type Description          Inuse   Pin Pgsp Virtual Addr Range
     0        0 work kernel seg            2701  1425  178  2431   0..32767 :
                                                                   65475..65535
...............................................................................
EXCLUSIVE segments      Inuse      Pin     Pgsp  Virtual
                          196        1        0      150

  Vsid     Esid Type Description          Inuse   Pin Pgsp Virtual Addr Range
  1181        2 work process private        125     1    0   125   0..46 :
                                                                   65309..65535
 1f1ff        f work shared library data     51     0    0    25   0..109
 1b1db        1 pers code,/dev/hd2:58        10     0    -     -   0..9
 180b8        - pers /dev/hd2:4105            3     0    -     -   0..2
 12212        - pers /dev/hd2:14935           1     0    -     -   0..0
 11211        - pers /dev/hd9var:21           1     0    -     -   0..0
 170b7        - pers /dev/hd2:2               1     0    -     -   0..0
 160b6        - pers /dev/hd4:25              1     0    -     -   0..0
  90c9        - pers /dev/hd9var:17           1     0    -     -   0..0
 1b07b        - pers /dev/hd3:2               1     0    -     -   0..0
  b0ab        - pers /dev/hd2:4098            1     0    -     -   0..0

...............................................................................
SHARED segments         Inuse      Pin     Pgsp  Virtual
                          748        0        0       40

  Vsid     Esid Type Description          Inuse   Pin Pgsp Virtual Addr Range
  9029        d work shared library text    747     0    0    40   0..65535
 19099        - pers /dev/hd4:757             1     0    -     -   0..0
#
```

6.25.3.4 Svmon Scaling

The svmon command has also been changed internally, such as by the addition of six new global page counters, to help enable it to scale on large systems. The new counters are:

- Number of pages in use for working segment
- Number of pages pinned for working segment
- Number of pages in use for persistent segment
- Number of pages pinned for persistent segment
- Number of pages in use for client segment

- Number of pages pinned for client segment

6.25.3.5 rmss Kernel Extension

`Svmon` requires some system calls that are not part of the kernel. They are delivered in a dedicated kernel extension named rmss.ext. The command `svmon` is actually divided in two binaries. The first binary (/usr/bin/svmon) only loads the kernel extension rmss.ext and then execs the second binary /usr/lib/perf/svmon_back.

6.25.4 Trace Based Tools Scaling

The system `trace` utility has been changed to optionally provide one buffer set per CPU. Two other tools associated with trace also have been improved:

- `tprof` - This tool has been enhanced to produce separate statistics per processor when running off-line and in conjunction with a new trace option to produce one trace log file per processor.
- `pprof` - This is a lightweight, trace-based tool which collects a system's process and thread information. Reports are generated in several formats, including a family view. The family view displays all parent-child relationships for all processes and threads. This tool is especially helpful in pinpointing system degradation when caused by multiple processes.

6.25.4.1 Trace Changes

With the growing number of processors in a system, there may be an increase in buffer contention that affects system performance. Hence, AIX Version 4.3.3 optionally provides separate buffers per CPU so as to maximize trace output by minimizing buffer contention. Up to 1024 CPUs can now have their own trace buffer.

If this function is invoked, the `trace` command will produce one file per CPU, one file per set of buffers, plus the base file, which is /var/adm/ras/trcfile by default. Thus the trace is still able to be accessed with a single file. The base file will tell the utilities what type of trace this is. The files may then be merged by `trcrpt` when you generate the report.

Trace Documentation

The AIX documentation for the changes in the `trace` command has been included in the file /usr/lpp/bos/README. One of the important items in this file is the statement:

NOTE: It is strongly recommended that the -B flag is used to tell `trace` to allocate buffers in separate segments rather than using memory from the kernel heap.

Syntax Changes in trace, trcrpt and snap

Since this change will primarily be used by the performance measurement tools, the change has not yet been formally documented. Currently, the command line is the only way to use the new trace options directly. The main changes in the `trace`, `tcrpt` and `snap` commands are described here:

- Both `trace` and `trcrpt` now have a -C flag for separate CPU tracing. The format is -C cpuid-list. The cpuid-list can be a list of CPU ids, beginning with 0, separated by commas or spaces. The list format is the same as for the -j and -k flags. In addition, however, the list might be the reserved word "*" or "all", for all CPUs.

  If -C is specified, the `trace` command will allocate separate buffer(s) per CPU, and create separate files as well. By default, the files will be named trcfile, trcfile-0, trcfile-1, and so forth. Be aware of three other flags:

  -c This flag saves the old trace file in a .old file, so it should not be used with the new -C flag, since the new CPU specific files will not be saved.

  The files will be created in the same directory, /var/adm/ras by default. If you want to transfer these files, they may be placed in a different directory than the one in which they were created, but they must not be renamed.

  -o If -o filename is specified along with -C, then -0 through -(n-1) is appended to the name specified with -o for a n-way CPU system.

  For example,

  ```
  trace -C all -o /tmp/tlog
  ```

  would produce files named /tmp/tlog, /tmp/tlog-0, /tmp/tlog-1, and so on. If you don’t specify a file name, the trace command's stream output option, -o, will not be supported.

-f If you use -f for a single mode trace, as opposed to the wraparound default option, tracing stops when the last buffer fills. As each buffer fills, a buffer wrap hook, 006, will be produced, so trace readers may elect to stop processing data as soon as the first wrap hook is seen, or they may continue.

- `trcrpt` merges the information traced. The information is merged based on the time stamps in the entries. When a multi-CPU trace is being processed, the CPU ID is displayed along with each reported entry, as though -O cpuid=on was specified. Note that for `trcrpt`, -C all or -C * is unnecessary. Note that there is no ambiguity here since the command used to start the trace is shown as part of the report header. For example, if you execute

  ```
  trace -a -C 0,1,2
  ```

  this specifies to trace CPUs 0, 1, and 2. If you then execute `trcrpt` (no options), the report will consist of all CPUs traced, and the header shows trace -a -C 0,1,2. Also, if all CPUs were traced, but `trcrpt -C 5` is given, then only the trace for CPU 5 is shown. An error is produced if there is no trace information to report for the CPUs specified.

  -v This verbose option for the `trcrpt` command now gives information about which CPUs were traced and the associated file names.

- `snap` has been be updated to gather multiple trace files, with the addition of the -T option. Hence, you should use something similar to

  ```
  snap -gT ./trace.log1 -d /junk/new-trc/snap
  ```

  If no file name is specified for the "-T" option, then use "." so that the default root trace file, /var/adm/ras/trc, and any CPU files are saved if the trace was a multi CPU trace. If you only use `snap -g`, then you will get only the base file and be warned by the following message:

  ```
  WARNING: This is a multicpu trace. Specify -T flag to copy all CPU
  files.
  ```

Using a Multiple CPU Trace

You can execute a command sequence similar to the following example:

```
# trace -C all -a -o/junk/new-trc/trace.log1
# cp /etc/motd /tmp/junk
# trcstop
# trcrpt -C0 trace.log1 > ./trcrpt.C0-only.out
# trcrpt trace.log1 > ./trcrpt.all-cpus.out
# ls -l
total 37504
-rw-rw-rw-   1 root     system     23552 Aug 03 15:02 trace.log1
```

```
-rw-rw-rw-   1 root     system   1311816 Aug 03 15:03 trace.log1-0
-rw-rw-rw-   1 root     system   1311816 Aug 03 15:03 trace.log1-1
-rw-r--r--   1 root     system   5797652 Aug 03 15:18 trcrpt.C0-only.out
-rw-r--r--   1 root     system   10744715 Aug 03 15:26 trcrpt.all-cpus.out
```

This example shows how this new trace function generates a trace file for each CPU, and that the trace data can be filtered for a particular CPU. Thus `trcrpt.C0-only.out` is approximately half the size of `trcrpt.all-cpus.out`.

The smaller file without a dash numerical suffix is known as the base file. This file contains only meta data; that is initial conditions. These initial conditions in the base file are the same as those written to a standard trace file, except a base file for a multi CPU trace contains the extra 00A 0x10 hooks. 00A is the standard utility hook, and 0x10 is the data word in this hook that identifies this as the hook mapping CPU to file. The initial conditions specify system environment data as of when the trace was started. For example, they contain a list of the processes running on the system when trace was started, the time conversion data, and loader data if -n was specified.

Trace Issues

Note that use of the -C flag creates trace buffers and trace log files for each processor that is traced. The buffer and file sizes specified apply to each buffer and file created. Therefore, if trace -C all is specified on a system with eight processors, eight sets of buffers will be allocated. In this case, eight files would also be created along with a trace header file pointing to the files for each CPU. The files created would be /var/adm/ras/trcfile, /var/adm/ras/trcfile-0 through /var/adm/ras/trcfile-7. Because of the extra pinned memory overhead, If -C is specified, no trace information will appear in a system dump, if one is taken. The kernel debuggers, such as `kdb`, will continue to format trace data given that the separate buffers option is not used. Also, the trace behavior of allocating one set of buffers will continue to be the default.

6.25.4.2 Tprof

Given the change in the `trace` command such that you now can obtain CPU specific trace data, the `tprof` command has also been updated to process this multi CPU data.

An updated `tprof` man page is in B.1, “tprof: Traces from Multiple CPUs” on page 697.

`tprof` operates in two modes, the first is called the online mode which has `tprof` execute the system `trace` command and a specific program, if specified, which is to be profiled. After the `trace` has completed, `tprof` processes the trace data and produces report files. In the offline mode, the trace data has already been gathered and `tprof` simple reads from this file and as before processes the trace data and produces report files. If multiple trace files exist from multiple CPUs, then `tprof` can also provide you with CPU specific data.

Note that the `gennames` tool is used in conjunction with the new offline mode capability found in tprof. It consolidates and simplifies the information needed by `tprof` to process symbol, loader and extension information.

If the -C flag is specified, indicating that there are multiple CPU trace files then `tprof` will spawn multiple threads, one per trace file. The new syntax is

```
tprof { [-i trace_file] [-n gennames_file] [-C all | list] }
```

If you created multiple CPU trace files with the `trace` command the base name of the file should be entered after the -i option. The -C option could be used to specify the CPUs to run `tprof` against and to generate per CPU `tprof` reports.

Using trcrpt, gennames and tprof With Traces From Multiple CPUs

If you have a trace from a system with two CPUs, then you can execute the following command sequence to analyze these trace files offline with this new `tprof` feature. The method is similar to that documented in the `tprof` man page for analysis of a single trace file. Assume that the trace base file is `trace.log1`, so that the two CPU trace files are `trace.log1-0` and `trace.log1-1`.

- Create the files needed by tprof, with trcrpt and gennames:

  ```
  trcrpt -r trace.log1 > trace.log1.out
  trcrpt -r trace.log1-0 > trace.log1.out-0
  trcrpt -r trace.log1-1 > trace.log1.out-1
  gennames > gennames.out
  ```

- Execute tprof:

  ```
  # tprof -i trace.log1.out -n gennames.out -C all
  Tue Aug  3 15:02:41 1999
  System: AIX 433c Node: 4 Machine: 000416314C00

  0.000 secs in measured interval
  Will process cpu trace file trace.log1.out-0
  Will process cpu trace file trace.log1.out-1
  15.492 secs in measured interval
  17.492 secs in measured interval
   * Samples from __trc_rpt2
   * Samples from __trc_rpt2
   * Reached second section of __trc_rpt2
   * Reached second section of __trc_rpt2
  ```

```
Combining data from __trc_rpt2-0
Combining data from __trc_rpt2-1
Creating Summary Reports
 * Samples from __trc_rpt2
 * Reached second section of __trc_rpt2
```

- The directory where you have the `trace` and `tprof` data should be similar to:

```
# ls
__ldmap                 __tmp.u                 trace.log1
__prof.all              __tmp.u-0               trace.log1-0
__prof.all-0            __tmp.u-1               trace.log1-1
__prof.all-1            __trc_rpt2              trace.log1.out
__tmp.k                 __trc_rpt2-0            trace.log1.out-0
__tmp.k-0               __trc_rpt2-1            trace.log1.out-1
__tmp.k-1               gennames.out            trcrpt.C0-only.out
__tmp.s                 snap                    trcrpt.all-cpus.out
__tmp.s-0               snap-only-g             trcrpt.all-cpus.out-v
__tmp.s-1               trace-ri-cmds
```

- The `trpof` man page describes the reports it generates, but the following output shows how you now have both the summary report for all CPUs, and a report for each CPU in the system that you are using.

  You can review the data for both CPUs with a command similar to:

```
# tail -3 *prof*.all
          =======      ===    =====   ======     ====   ======    =====
            Total        8     1555     1554        0        1        0
```

  You can review the data for CPU 0 with a command similar to:

```
# tail -3 *prof*0
          =======      ===    =====   ======     ====   ======    =====
            Total        4      692      692        0        0        0
```

  You can review the data for CPU 1 with a command similar to:

```
# tail -3 *prof*1
          =======      ===    =====   ======     ====   ======    =====
            Total        5      863      862        0        1        0
```

  You can see that 862 and 692 add up to give 1554.

6.25.4.3 pprof

Previous versions of AIX included a fileset called bos.perf.pmr. This fileset included a shell script, trace reader and `awk` script which together allowed you to record the CPU usage of all processes on a system over a time interval. This fileset is no longer shipped with AIX since this function has been moved into a single C executable, called pprof, which is now shipped as part of the perfagent.tools fileset. The `pprof` man page describes the five types of reports that it can generate. Note that the perfpmr package (the Performance PMR

Data Collection Scripts.) is still available for download from the Internet, for different releases of AIX. This package is still quite useful to help you resolve performance problems.

6.25.5 ipfilter Script

This tool sorts the information provided by the `ipreport` command and presents it in table format. The `ipfilter` command allows you to select which operation headers (NFS, TCP, UDP, IPX and ICMP) to view, displayed in three different reports. The tool is a script which contains both korn shell and `awk` code.

Use ipreport -s

ipfilter reads a report that is created by `ipreport -s`. This command option adds the protocol specification to every line in a packet). If you have a trace with no UDP, NFS, TCP, IPX, or ICMP type protocols, then you will have no output from the ipfilter command. For example, ARP data is currently ignored.

6.25.5.1 Using the new ipfilter Tool

Create your ipreport file as usual with the `iptrace` and `ipreport` commands. Assume that you created an ipreport output file called `ip.rpt.`, and want to create the default output file called `ipfilter.all`. You could then use the new `ipfilter` tool in a manner similar to that shown in the following example:

1. Check the file created by the ipreport command.

```
# head ip.rpt
IPTRACE version: 2.0

Packet Number 1
FDDI: ====( 4372 bytes transmitted on interface fi0 )====
06:18:59.809370880
FDDI: FDDI packet  FDDI: FDDI MAC header:
FDDI: frame control field = 50
FDDI: [ src = 10:00:5a:b8:7a:5f, dst = 10:00:5a:b8:29:40]
....
```

2. Use the ipfilter command:

```
# ipfilter ip.rpt
```

3. Verify that the output file exists with the ls command. A zero byte file would suggest that your trace did not have the protocol information that ipfilter is looking for.

```
# ls -l
total 5656
-rw-r--r--   1 root     sys      2583316 Aug 10 14:55 ip.rpt
-rw-r--r--   1 root     sys       305803 Aug 10 14:56 ipfilter.all
```

4. Finally, look at the output file.

```
# head ipfi*
                                Operation Headers:  ICMP IPX NFS TCP UDP


Ports

------------------------------------
 pkt.     Time          Source          Dest.   Length Seq #     Ack #      Source
Destination Net_Interface Operation
------------------------------------------------------------------------------------
-----------------------------------------------
   1 06:18:59.809370      110.1.1.11      110.1.1.10 4348,
766,  2049(shilp)           fi0 UDP NFS
   2 06:18:59.809387      110.1.1.11      110.1.1.10 4008,
0            0           fi0
   3 06:18:59.809581      110.1.1.11      110.1.1.10 1168,
766,  2049(shilp)           fi0 UDP NFS
   4 06:18:59.819224      110.1.1.11      110.1.1.10 4348,
831,  2049(shilp)           fi0 UDP NFS
#
```

The output in the file ipfilter.all is wide. You can improve the display with a wide screen. For example, use `aixterm -fn Rom8` and then make your aixterm window wide enough to display all the data without wrapping.

6.25.6 CPU Utilization Enhancement

Previous releases of AIX would provide a discrepancy in the measurement of the accurate time in ticks that a given CPU was busy. This error was due to the clock interrupt handler not considering the missed ticks to be counted in measuring the CPU busy time. If any higher priority interrupt was running on this CPU masking the clock interrupt, then the clock interrupt handler may not have had a chance to run every tick to update the sysinfo and cpuinfo counters. For example, a device driver may be using the CPU for a while but have appeared to have only used say one clock tick.

The system counter and the per CPU counter now track these previously mixed ticks so that the accuracy of system and per CPU busy or idle time has been improved.

6.26 Mksysb on CD-R (4.3.3)

CD-recordable devices (CD-R) is supported as a mksysb media on AIX 4.3.3. There are three types of CDs that can be created, discussed in the following sections:

- Personal System Backup
- Generic Backup
- Non-bootable Volume Group Backup

Note: Only 74 minutes CD-R is supported.

6.26.1 Personal System Backup CD

This type of mksysb backup is same as the mksysb backup on a tape media. This backup CD can only boot and install the original machine that the backup has been taken or the machine with same platform and device configuration.

6.26.2 Generic Backup CD

This type of backup CD is used to boot and install any RS/6000 platform (rspc, rs6k or chrp). It contains all three boot images and the device and kernel filesets to enable cloning. The kernel used to build the boot image must be an MP kernel (not necessary to be running) because this kernel supports both UP and MP RS/6000.

6.26.3 Non-Bootable Volume Group Backup

This type of backup CD is non-bootable and contains only a volume group image. If the image in the CD is a rootvg image, then the CD can be used to install AIX after booting from a product CD-ROM. This CD can also be used as a source media for `alt_disk_install` command. CD-R can be used as a backup media for non-rootvg volume group and the volume group can be restored using `restvg` command.

6.26.4 Tested Software and Hardware

Because IBM does not sell or support the RS/6000 software and hardware to create CDs, they must be obtained from third vendors. The following table lists tested software, hardware, and their combinations required for this feature:

Software	Hardware
GNU & Free Software Foundation, Inc. http://www.gnu.org cdrecord version 1.8a5 mkisofs version 1.5	Yamaha CRW4416SX Ricoh MP6201SE 6XR-2X Panasonic Cw-7502-B
Jodian Systems and Software, Inc. http://www.jodian.com CDWrite version 1.3 mkcdimg version 2.0	Yamaha CRW4416SX Ricoh MP6201SE 6XR-2X Panasonic Cw-7502-B
Youngminds, Inc. http://www.ymi.com MakeDisc Version 1.3-Beta2	CD Studio

Note that these software is used in conjunction with `mkcd` command to take backups on CD-Rs.

6.26.5 mkcd Command

Mksysb or savevg images are written to CD-Rs using `mkcd` command. At the time of this writing, the `mkcd` command does not support multiple CDs for same mksysb or savevg images. Multi-volume backup will be supported by applying APAR IY02960 so that the backup size is not limited to 640 MB.

The `mkcd` command requires code supplied by third vendors so that it can create the RockRidge file system and write the backup image to CD-R media. This code must be linked to /usr/sbin/mkrr_fs (for creating the Rock Ridge format image) and /usr/sbin/burn_cd (for writing to the CD-R device). For example, if you are using Jodian software, then you will need to create the following links:

```
ln -s /usr/samples/oem_cdwriters/mkrr_fs_jodian /usr/sbin/mkrr_fs
ln -s /usr/samples/oem_cdwriters/burn_cd_jodian /usr/sbin/burn_cd
```

The process for creating a mksysb CD using `mkcd` command is:

1. If file systems or directories are not specified, they will be created by `mkcd` and removed at the end of the command (unless the -R or -S flags are used). `mkcd` will create following file systems:
   - /mkcd/mksysb_image

     Contains a mksysb image.
   - /mkcd/cd_fs

Contains CD file systems structures. At least 645MB of free space is required.

- /mkcd/cd_image

  Contains final CD image before writing to CD-R. At least 640MB of free space is required.

  These file systems can be NFS mounted.

  The file systems provided by the user will be checked for adequate space and an error will be given if there is not enough space. Write access will also be checked.

2. If a mksysb image is not provided, `mkcd` calls `mksysb`, and stores the image in the directory specified with the -M or in /mkcd/mksysb_image.
3. It calls the mkcdfs function to create the directory structure and copy files based on the cdfs.required.list and the cdfs.optional.list files. It only copies files and commands for the first CD.
4. If a CHRP boot image is required, it generates one and place it in required directory in the CD file system.
5. Device images are copied to ./usr/sys/inst.images if the -G flag is used or the -l flag is given with a list of images to copy.
6. It then copies the mksysb image to the file system. It determines the current size of the CD file system at this point, so it knows how much space is available for the mksysb. If the mksysb image is larger than the remaining space, multiple CDs are required. It only `dd` the specified number of bytes of the image to the CD file system. It then updates the volume ID in a file. A variable is set from a function that determines how many CDs are required to hold the entire mksysb image.
7. The `mkcd` then calls the `mkrr_fs` command to create a Rockridge file system and place the image in the specified directory.
8. If the -G flag is specified, rs6k and rspc boot images are created with the `bosboot` command. If the -G flag is not specified, then the boot image for the platform of the machine is generated. The `cdboot` command is called to create the CD-ROM image. This image replaces the first CD file system image.
9. The OEM command to burn the CD is then called and the first CD is created.
10. If multiple CDs are required, the user is instructed to remove the CD and put the next one in and the process continues until the entire mksysb image is put on the CDs. The second CD does not contain boot images, device support or commands in the file system.

6.27 System RAS improvements (4.3.3)

AIX has numerous new and improved Reliability/Availability/Serviceability (RAS) features which are described in this section. A new log is now used to capture system console messages, since they may contain important information. The file system, system error log, and the system dump process have been enhanced to make it easier to diagnose problems when they occur.

6.27.1 AIX Console Logging

AIX now treats system console messages as critical system information. Previously, these messages were displayed on the current console device. If that screen or window was in use, the messages could be lost. Now, in addition to displaying them on the console, these messages are also logged to a file, along with the originating user and the time of day when the message was written. You can easily retrieve these messages, thus improving your ability to diagnose problems and monitor system status.

6.27.1.1 What is Logged

Only output sent to the console is logged. Any output sent to a device acting as the console is not logged. This means that system informational, error, and intervention-required messages are captured (logged), while other types of output seen at the console are not. For example, getty output, `smitty` output, user interaction (such as command output) at the physical console device, and so forth, are activities that are not logged.

6.27.1.2 How Log Data is Stored

The log file is based on the alog format; this format allows the file to wrap after it attains a predetermined maximum size, which is in multiples of 4096 bytes. The alog command is typically used to view the console log file. The console log file deviates from the normal alog format in that each record of the file contains, in addition to the logged text, the user ID that wrote to the console and the epoch time when it was written. The epoch time is formatted and displayed in the user’s locale date and time when the file is accessed using the alog command, as shown in “Using the Console Log” on page 305.

When the console device is configured or when any modification is made to the console log file, ownership of the file is set to root and permissions are set to 622 to match that of the console device driver special file. The root user can modify the ownership or permissions, but they will not persist across boots.

6.27.1.3 Using the Console Log

You can use one of two commands to enable the console log:

- Execute `swcons -p log_file`

  The swcons command is used to make changes to console logging parameters during system operation. These changes are rescinded at the next console device configuration (typically reboot), and the original console logging parameters are reinstated.

- Execute `chcons -a console_logname=log_file`

  The chcons command is used to make changes to the console logging parameters for the next console device configuration (typically reboot). These changes do not apply to the current running system.

An example of the use of the console log follows:

1. Check the status of the proposed log file; here it does not exist:

   ```
   # ls -l /var/adm/ras/conslog1
   ls: 0653-341 The file /var/adm/ras/conslog1 does not exist.
   ```

2. Enable the log using the swcons command and an absolute path log file name:

   ```
   swcons -p'/var/adm/ras/conslog1' -s'102400' -v'1' -t'1'
   ```

3. Verify that the log file now exists and is the size you specified:

   ```
   # ls -l /var/adm/ras/conslog1
   -rw--w--w-   1 root     system    102400 Aug 30 14:07 /var/adm/ras/conslog1
   # ls -l /dev/console
   crw--w--w-   1 root     system     4,  0 Aug 18 16:29 /dev/console
   ```

   Note that both files have 622 file permissions. The -v flag specified the verbosity level for the console output that is logged. The -t flag specified the verbosity level for console output tagging, which you can use to track the effective user ID of the user that sent the message to the console. Both flags require any value greater than zero.

4. Send some messages to the console device:

   ```
   # echo Data, data everywhere > /dev/console
   # echo Do not ignore me > /dev/console
   # echo Do not forget me > /dev/console
   ```

5. Display the console log (The root user ID is 0) by executing:

   ```
   # alog -f /var/adm/ras/conslog1 -o
            0 Mon Aug 30 14:30:01 CDT 1999 Data, data everywhere
            0 Mon Aug 30 14:30:17 CDT 1999 Do not ignore me
            0 Mon Aug 30 14:30:55 CDT 1999 Do not forget me
   # ls -l /var/adm/ras/conslog1
   -rw--w--w-   1 root     system    102400 Aug 30 14:30 /var/adm/ras/conslog1
   ```

6.27.2 Error Log Enhancements

The error logging service aid is the primary mechanism in AIX for runtime recording of information about hardware and software malfunctions. The purpose of error logging is to collect and record data related to a failure so that it can be subsequently analyzed to determine the cause of the problem, and take corrective action.

6.27.2.1 Expanded Error Log Size

Every error logged to the error logging subsystem can contain detail data at the end of the record entry that is added to the log. This is the section of the error log that you would see at the end of each error when you display the long version of the system error log using the `errpt -a` command. The relevant lines may be similar to the following output:

```
Detail Data
PROBLEM DATA
0144 0000 0000 00C0 C600 9108 1929 2400 1998 0526 0000 0000 0000 0000 0000
0000
```

Previous releases of AIX limited this `PROBLEM DATA` to 230 bytes. For large systems with extra data that needs to be recorded along with register contents, this is not sufficient. Hence, this has been increased to 1024 bytes. Note that even with this change, there are other factors that may limit what is saved. On an old POWER machine the NVRAM is limited to 512 bytes. So if an error is logged here with 1024 bytes of detail data the error logged will be truncated to 512 bytes. This is not be a problem on CHRP machines which have a bigger share of NVRAM devoted to error logging.

6.27.2.2 JFS Error Logging

The error log was originally intended to be for hardware and associated software components, such as for example a device driver logging an error condition from a disk drive. The error log has evolved into a more general purpose mechanism for logging interesting system conditions that need to be reviewed, analyzed and which may require action by you.

Previous releases of AIX did not have any error logging calls made in the file system. File system related errors were not easily identifiable and were often associated with excessive system down time and administration work. Many errors involved system dump analysis for the crashes they caused. Not all of theses errors continue to trigger a system crash, or assert call, but irrespective of a crash, there will now often be useful information in the system error log. There are now 17 new error log types (see Table 30 on page 307), each with a unique label, description, information regarding the causes of the errors, and finally recommended actions. Where appropriate, detail data (For example, file system name, inode number) is also included.

Table 30. New JFS Related Error Labels

JFS_COMP_CORRUPTION	JFS_FS_FRAGMENTED	JFS_FS_FULL
JFS_FS_NOINODES	JFS_FSCK_REQUIRED	JFS_KERNHEAP_DELAY
JFS_KERNHEAP_LOW	JFS_LOG_EXCEPTION	JFS_LOG_WAIT
JFS_LOG_WRAP	JFS_LOG_WRITE_ERR	JFS_META_WRITE_ERR
JFS_META_CORRUPTION	JFS_META_EXCEPTION	JFS_USER_HARDLINK
JFS_USER_WRITEMOUNT	SPECFS_DDINTPRI	

The main error log related JFS serviceability changes can be summarized as follows:

- An error log entry is created when errors require intervention.
- The file system superblock is marked dirty whenever conditions are detected which require a full file system recovery using the `fsck` utility.
- An error code is returned which uniquely identifies a failure as due to file system corruption (for non-fatal errors).
- The error log is not flooded with duplicate entries for the same problem.
- Logging of JFS-specific events is restricted to JFS file systems.

6.27.2.3 Examples of the JFS Error Log Entries

The new entries are listed and described, along with the context of their generation. Not all the details for each entry that you will see from the errpt -a command are shown. Rather, only the new parts of each detailed entry are shown in the following list. For example, the error discussed in “No Free Space in File System” on page 310 will be displayed by the errpt -a command with details similar to the following example:

```
LABEL:          JFS_FS_FULL
IDENTIFIER:     369D049B

Date/Time:       Tue Aug 24 14:53:33
Sequence Number: 36
Machine Id:      000416314C00
Node Id:         rootmaster
Class:           O
Type:            INFO
Resource Name:   SYSPFS
```

```
Description
UNABLE TO ALLOCATE SPACE IN FILE SYSTEM

Probable Causes
FILE SYSTEM FULL

        Recommended Actions
        USE FUSER UTILITY TO LOCATE UNLINKED FILES STILL REFERENCED
        INCREASE THE SIZE OF THE ASSOCIATED FILE SYSTEM
        REMOVE UNNECESSARY DATA FROM FILE SYSTEM

Detail Data
MAJOR/MINOR DEVICE NUMBER
000A 0004
FILE SYSTEM DEVICE AND MOUNT POINT
/dev/hd4, /
```

Questionable User Activities

Certain user actions are inappropriate and run the risk of data loss or corruption, but are not necessarily disallowed. Examples of typical error log entries generated as a consequence of these actions include:

- Directory Hard Links

  The generation of an error log entry anytime a directory hard link is created or destroyed:

```
Label: JFS_USER_HARDLINK
Class:    O
Type:     INFO

Description
INAPPROPRIATE FILE SYSTEM OPERATION

User Causes
DIRECTORY HARD LINK CREATED OR REMOVED

        Recommended Actions
        REVIEW DETAILED DATA

Detail Data
DETECTING MODULE
MAJOR/MINOR DEVICE NUMBER
INODE NUMBER
DIRECTORY
```

- Writing to Mounted File System Device

  An error log entry is emitted when a device containing a mounted JFS file system or JFS log is written to by any process other than a recognized file system utility. (For example, the `fsck` command is a recognized utility.) To preserve binary compatibility, the operation will still be allowed.

-

```
        Label: JFS_USER_WRITEMOUNT
        Class:    O
        Type:     INFO

        Description
        INAPPROPRIATE FILE SYSTEM OPERATION

        User Causes
        PROCESS WROTE TO DEVICE CONTAINING A MOUNTED FILE SYSTEM

                Recommended Actions
                REVIEW DETAILED DATA

        Detail Data
        MAJOR/MINOR DEVICE NUMBER
        PROGRAM NAME
        USER'S PROCESS ID:
        ADDITIONAL INFORMATION
        uid=0 gid=0
```

- Device Driver Returning With Interrupts Disabled

  While not strictly a user error, the device driver has a bug (interrupts must be re-enabled before returning), and using it causes a system crash. The DD_ENT macro was modified to emit an error log entry whenever a driver returns with interrupts disabled. As the errpt entry shows, this is one failure scenario that still results in a system crash where a dump is generated:

-

```
        Label: SPECFS_DDINTPRI
        Class:    S
        Type:     PERM

        Description
        DRIVER RETURNED WITH INTERRUPTS DISABLED

        Failure Causes
        SOFTWARE DEVICE DRIVER
```

```
          SOFTWARE PROGRAM

                  Recommended Actions
                  OBTAIN DUMP
                  CHECK ERROR LOG FOR ADDITIONAL RELATED ENTRIES
                  IF PROBLEM PERSISTS, CONTACT APPROPRIATE SERVICE
REPRESENTATIVE

          Detail Data
          DEVICE MAJOR NUMBER
          RETURN CODE
          ADDITIONAL INFORMATION
          dd_entry=d_*
```

System Limits Related Errors

Error log example entries in this section are intended to provide information necessary to take remedial action. No change was made to the previous behavior (whether or not to crash), unless otherwise specified.

- No Free Space in File System

  An error log entry is generated for disk block allocation failures. This is done once per file system mount, unless your attempt to extend the file system has still not ensured that enough free space is available to meet your needs.

```
Label: JFS_FS_FULL
Class:    O
Type:     INFO

Description
UNABLE TO ALLOCATE SPACE IN FILE SYSTEM

Probable Causes
FILE SYSTEM FULL

        Recommended Actions
      USE FUSER UTILITY TO LOCATE UNLINKED FILES STILL REFERENCED
        INCREASE THE SIZE OF THE ASSOCIATED FILE SYSTEM
        REMOVE UNNECESSARY DATA FROM FILE SYSTEM

Detail Data
MAJOR/MINOR DEVICE NUMBER
FILE SYSTEM DEVICE AND MOUNT POINT
```

In the case of memory mapped files where JFS is bypassed, the error logging is deferred until the next time segments are committed for this file system.

- Excessive File System Fragmentation

  An error log entry is generated for an excessively fragmented disk map, with the suggestion to execute the `defragfs` command. This is done once per file system mount, unless the error condition occurs again after the file system attributes have been changed. A file system is considered excessively fragmented if an allocation fails with ENOSPC, but the number of fragments available is greater than that needed to fulfill the smallest contiguous allocation requirement (there is enough space, but it is not contiguous.)

```
Label: JFS_FS_FRAGMENTED
Class:    O
Type:     INFO

Description
UNABLE TO ALLOCATE SPACE IN FILE SYSTEM

Probable Causes
FILE SYSTEM FREE SPACE FRAGMENTED

        Recommended Actions
        CONSOLIDATE FREE SPACE USING DEFRAGFS UTILITY

Detail Data
MAJOR/MINOR DEVICE NUMBER
FILE SYSTEM DEVICE AND MOUNT POINT
```

- Out of Inodes in File System

  An error log entry is emitted if the file system is out of inodes. This is done once only per file system mount, unless your attempt to extend the file system has still not ensured that enough free inodes are available to meet your needs.

```
Label: JFS_FS_NOINODES
Class:    O
Type:     INFO

Description
UNABLE TO ALLOCATE SPACE IN FILE SYSTEM
```

```
Probable Causes
FILE SYSTEM OUT OF INODES

        Recommended Actions
      USE FUSER UTILITY TO LOCATE UNLINKED FILES STILL REFERENCED
        INCREASE THE SIZE OF THE ASSOCIATED FILE SYSTEM
        REMOVE UNNECESSARY DATA FROM FILE SYSTEM
        REMAKE FILE SYSTEM AND RESTORE USER DATA

Detail Data
MAJOR/MINOR DEVICE NUMBER
FILE SYSTEM DEVICE AND MOUNT POINT
```

- Out of Kernel Heap Space in dlistadd()

  In the dlistadd() function, the system would previously crash if xmalloc() failed due to kernel heap exhaustion. This assert has been eliminated by the implementation of a new algorithm. If the heap allocation still fails, after a reasonable amount of time, then the superblock is marked dirty. This means that you will need to use the `fsck` command before your next attempt to mount the file system. To avoid flooding the error log, these errors should appear only once per boot, or every 24 hours, whichever is the shorter period.

  The following appears in a new error log entry when the JFS kernel heap space is low:

```
-
          Label: JFS_KERNHEAP_LOW
          Class:    O
          Type:     PERM

          Description
          UNABLE TO ALLOCATE SPACE IN KERNEL HEAP

          Failure Causes
          SOFTWARE PROGRAM

                  Recommended Actions
                  IF PROBLEM PERSISTS, CONTACT APPROPRIATE SERVICE REPRESENTATIVE
```

  The new error entry generated when memory becomes available is:

```
          Label: JFS_KERNHEAP_DELAY
          Class:    O
          Type:     TEMP

          Description
          ALLOCATED KERNEL HEAP SPACE AFTER DELAY

          Failure Causes
          SOFTWARE PROGRAM
```

```
        Recommended Actions
        IF PROBLEM PERSISTS, CONTACT APPROPRIATE SERVICE REPRESENTATIVE

Detail Data
ADDITIONAL INFORMATION
Delay=delay_value, approximately (delay*2/HZ) seconds.
```

• JFS Log Wrap

Error log entries are generated to suggest an increase in the size of the JFS log whenever the log wraps or is in danger of wrapping. The entries are:

```
Label: JFS_LOG_WRAP
Class:    O
Type:     INFO

Description
JFS LOG FULL (WRAPPED)

        Recommended Actions
        INCREASE THE SIZE OF THE JFS LOG DEVICE

Detail Data
MAJOR/MINOR DEVICE NUMBER
ADDITIONAL INFORMATION
```

The preceding `JFS_LOG_WRAP` error is a fatal error; that is, it will result in a system crash. To avoid flooding the error log, this entry is emitted only if this error condition occurs more than 10 times within a one hour period.

```
Label: JFS_LOG_WAIT
Class:    O
Type:     INFO

Description
FILE SYSTEM PERFORMANCE IMPAIRED

        Recommended Actions
        INCREASING THE SIZE OF THE JFS LOG DEVICE MAY IMPROVE PERFORMANCE
        UNDERSIZED LOG CAN LEAD TO SYSTEM CRASH

Detail Data
MAJOR/MINOR DEVICE NUMBER
ADDITIONAL INFORMATION
```

File System Meta-data Corruption

The detection of corrupted meta data previously almost always resulted in a fatal system failure. Previously, there was no simple way to determine that such a crash was due to file system corruption. Such crashes would have required effort to try and provide some useful recommended actions.

These crash circumstances have been reviewed, and crashes avoided where possible. Even if the system still crashes, an error log entry identifying which file system was corrupted and recommending checking cables, updating microcode and running `fsck`, helps you make some preliminary problem diagnosis and repair.

For meta data corruption, the `errdemon` daemon logs what information is available at the time of the corruption as an aid to service personnel, marks the superblock dirty as soon as possible, (if possible), and then advises which file system requires the use of the `fsck` command.

- Corrupted File System Control Data

  Upon detection of corrupted file system control data, one of two functions will be called: either v_jfscorruption() from critical sections, or jfscorruption(). These functions provide a central location for dealing with meta data corruption.

  Unfortunately, this error may result in a system crash so the associated error log message reminds you to obtain the dump for analysis. For example, you may want to verify that the dump was related to JFS corruption by looking for the execution of these functions.

  Since calling another function runs the risk of losing data that would normally be retained in registers due to an immediate state-save by the assert() call, data useful for debugging purposes is saved in a global, pinned buffer prior to calling the corruption handling functions.

  The jfserrlog() function stores up to 16 words of detail data in a global pinned buffer (similar to the vmmerrlog). The jfserrlog begins with a four-word header containing a unique tag, a time stamp, the VFS number on which the error occurred, the PDTX of the file system, a location code corresponding to the error, and the number of items stored in the buffer.

  The jfserrsave_metadata() function emits a diagnostic error log entry. The additional data string contains the jfserrlog (similar to hardware sense data). The first new error log entry generated for this error condition is:

```
Label: JFS_META_CORRUPTION
Class:    U
Type:     UNKN
```

```
Description
FILE SYSTEM CORRUPTION

Probable Causes
INVALID FILE SYSTEM CONTROL DATA

        Recommended Actions
        PERFORM FULL FILE SYSTEM RECOVERY USING FSCK UTILITY
        OBTAIN DUMP
        CHECK ERROR LOG FOR ADDITIONAL RELATED ENTRIES

Failure Causes
ADAPTER HARDWARE OR MICROCODE
DISK DRIVE HARDWARE OR MICROCODE
SOFTWARE PROGRAM
STORAGE CABLE LOOSE, DEFECTIVE, OR UNTERMINATED

        Recommended Actions
        CHECK CABLES AND THEIR CONNECTIONS
        INSTALL LATEST ADAPTER AND DRIVE MICROCODE
        INSTALL LATEST STORAGE DEVICE DRIVERS
       IF PROBLEM PERSISTS, CONTACT APPROPRIATE SERVICE REPRESENTATIVE

Detail Data
DETECTING MODULE
MAJOR/MINOR DEVICE NUMBER
ADDITIONAL INFORMATION
```

A general error log entry is emitted whenever the superblock is marked dirty, which calls for running a full fsck on the file system. This informational entry can be tied to the diagnostic entry by comparing the major/minor device numbers with those in the previous error log entry.

```
Label: JFS_FSCK_REQUIRED
Class:    O
Type:     INFO

Description
FILE SYSTEM RECOVERY REQUIRED

        Recommended Actions
        PERFORM FULL FILE SYSTEM RECOVERY USING FSCK UTILITY

Detail Data
```

```
MAJOR/MINOR DEVICE NUMBER
FILE SYSTEM DEVICE AND MOUNT POINT
```

Some specific examples of file system corruption that may result in system crashes and, in most cases, the generation of the above system messages include:

1. Directory Corruption

   Whenever a corrupted directory is observed, an error log entry is emitted, but the file system's superblock is not marked dirty, and the failure is no longer likely to be fatal.

2. Inode Corruption

   An error log entry is generated whenever a corrupted inode is observed. The decision whether or not to halt the system is unchanged. Specific types of inode corruptions include:

   • Inode Allocation

     The dev_ipalloc() function expects a new, unallocated inode. If the inode read from disk has a non-zero link count or non-zero mode, an error log entry is emitted and the system is halted, as was the case for previous releases of AIX.

   • Inode Access

     A valid directory entry should not refer to an inode with a link count of zero. If an inode with a zero link count is detected, an error log entry is emitted and ENOENT returned. The link count is included in the additional information field.

   • Inode Geometry

     In the iallocind() function, if the size indicates indirect geometry but the inode pointer to the indirect block is null, the system crashes. A new error log entry is emitted just before the crash. The additional information field is empty for this entry.

3. Corruption Detected Accessing Allocation Maps

   An error log entry is generated whenever corruption is detected in the diskblock or inode allocation maps. The decision whether or not to halt the system is unchanged. This type of problem is one for which only a JFS_META_CORRUPTION entry can be made before the system is halted. Other types of fatal system asserts dealing with incorrect parameters passed internally do not have any error entries logged. One scenario that will result in this type of fatal assert is an attempt to allocate more fragments than can possibly exist.

4. Corruption Detected Accessing Indirect Block

   Corruption of a file’s indirect blocks can cause a variety of system crashes, including pbitset. An error log entry is generated whenever corrupted indirect blocks are detected. Much of the code dealing with indirect block corruption has been enhanced, and useful information, such as the segment ID, page number, inode address, inode number, and virtual indirect address, is often saved in the jfserrlog buffer.

I/O Errors

This is the fourth general category of JFS errors that should be logged that are distinct from the previous three groups of user errors, system limits, and meta-data corruption.

- Corruption Detected Accessing User Data

  Generally, silent corruption of file data is not detected by the file system. However, data corruption in a compressed file system can be detected during decompression. An error log entry is emitted whenever this data corruption is detected, but it is not fatal so the system does not crash. The error logged each time a problem file system is mounted is:

```
Label: JFS_COMP_CORRUPTION
Class:    U
Type:     UNKN

Description
CORRUPTED FILE IN COMPRESSED FILE SYSTEM

        Recommended Actions
        CHECK ERROR LOG FOR ADDITIONAL RELATED ENTRIES
        IF PROBLEM PERSISTS THEN DO THE FOLLOWING
        REMAKE FILE SYSTEM AND RESTORE USER DATA

Failure Causes
ADAPTER HARDWARE OR MICROCODE
DISK DRIVE HARDWARE OR MICROCODE
SOFTWARE PROGRAM
STORAGE CABLE LOOSE, DEFECTIVE, OR UNTERMINATED

        Recommended Actions
        CHECK CABLES AND THEIR CONNECTIONS
        INSTALL LATEST ADAPTER AND DRIVE MICROCODE
        INSTALL LATEST STORAGE DEVICE DRIVERS
        IF PROBLEM PERSISTS, CONTACT APPROPRIATE SERVICE REPRESENTATIVE

Detail Data
MAJOR/MINOR DEVICE NUMBER
FILE SYSTEM DEVICE AND MOUNT POINT
```

- Exceptions Caught Using longjmpx()

  Any I/O exception on meta data in the top half of JFS emits an error log entry. The jfsexception() function compares the exception address against the segments for any inode being accessed at the time of the exception, and all the special inodes. If the exception is due to an I/O error (EIO) then the following entry is made only once per file system per mount:

```
Label: JFS_META_EXCEPTION
Class:    U
Type:     PERM

Description
STORAGE SUBSYSTEM FAILURE

Probable Causes
I/O ERROR ON FILE SYSTEM CONTROL DATA

        Recommended Actions
        PERFORM FULL FILE SYSTEM RECOVERY USING FSCK UTILITY
        CHECK ERROR LOG FOR ADDITIONAL RELATED ENTRIES

Failure Causes
ADAPTER HARDWARE OR MICROCODE
DISK DRIVE HARDWARE OR MICROCODE
SOFTWARE PROGRAM
STORAGE CABLE LOOSE, DEFECTIVE, OR UNTERMINATED

        Recommended Actions
        CHECK CABLES AND THEIR CONNECTIONS
        INSTALL LATEST ADAPTER AND DRIVE MICROCODE
        INSTALL LATEST STORAGE DEVICE DRIVERS
        IF PROBLEM PERSISTS, CONTACT APPROPRIATE SERVICE
REPRESENTATIVE

Detail Data
DETECTING MODULE
ERROR CODE
INODE NUMBER
MAJOR/MINOR DEVICE NUMBER
ADDITIONAL INFORMATION
iar=0x0 exaddr=0x0
```

Similarly, an exception to a JFS log device may result in the following error log entry:

```
Label: JFS_LOG_EXCEPTION
Class:    U
Type:     PERM
```

```
          Description
          STORAGE SUBSYSTEM FAILURE

          Probable Causes
          I/O ERROR ON FILE SYSTEM LOG DEVICE

                  Recommended Actions
                  PERFORM FULL FILE SYSTEM RECOVERY USING FSCK UTILITY
                  CHECK ERROR LOG FOR ADDITIONAL RELATED ENTRIES

          Failure Causes
          ADAPTER HARDWARE OR MICROCODE
          DISK DRIVE HARDWARE OR MICROCODE
          SOFTWARE PROGRAM
          STORAGE CABLE LOOSE, DEFECTIVE, OR UNTERMINATED

                  Recommended Actions
                  CHECK CABLES AND THEIR CONNECTIONS
                  INSTALL LATEST ADAPTER AND DRIVE MICROCODE
                  INSTALL LATEST STORAGE DEVICE DRIVERS
                  IF PROBLEM PERSISTS, CONTACT APPROPRIATE SERVICE
REPRESENTATIVE

          Detail Data
          DETECTING MODULE
          ERROR CODE
          MAJOR/MINOR DEVICE NUMBER
```

- Meta-data Asynchronous I/O Failures

  If asynchronous write I/O errors to the file system meta-data or log device occur, then the superblock is marked dirty, and the error log entries include (the errors are logged once per mount or log open):

```
          Label: JFS_LOG_WRITE_ERR
          Class:   U
          Type:    PERM

          Description
          FILE SYSTEM LOGGING SUSPENDED

          Probable Causes
          I/O ERROR ON FILE SYSTEM LOG DEVICE

                  Recommended Actions
```

```
                HALT ALL FILESYSTEM ACTIVITY AND REMOUNT
                REFORMAT JFS LOG DEVICE
                CHECK ERROR LOG FOR ADDITIONAL RELATED ENTRIES

        Failure Causes
        ADAPTER HARDWARE OR MICROCODE
        DISK DRIVE HARDWARE OR MICROCODE
        SOFTWARE DEVICE DRIVER
        STORAGE CABLE LOOSE, DEFECTIVE, OR UNTERMINATED

                Recommended Actions
                CHECK CABLES AND THEIR CONNECTIONS
                INSTALL LATEST ADAPTER AND DRIVE MICROCODE
                INSTALL LATEST STORAGE DEVICE DRIVERS
                IF PROBLEM PERSISTS, CONTACT APPROPRIATE SERVICE
REPRESENTATIVE

        Detail Data
        ERROR CODE
        MAJOR/MINOR DEVICE NUMBER
        ADDITIONAL INFORMATION
        pdtx=0 sidx=0

---------------------------------------------------------------------------
        Label: JFS_META_WRITE_ERR
        Class:   U
        Type:    PERM

        Description
        FILE SYSTEM CORRUPTION

        Probable Causes
        I/O ERROR ON FILE SYSTEM CONTROL DATA

                Recommended Actions
                PERFORM FULL FILE SYSTEM RECOVERY USING FSCK UTILITY
                CHECK ERROR LOG FOR ADDITIONAL RELATED ENTRIES

        Failure Causes
        ADAPTER HARDWARE OR MICROCODE
        DISK DRIVE HARDWARE OR MICROCODE
        SOFTWARE DEVICE DRIVER
        STORAGE CABLE LOOSE, DEFECTIVE, OR UNTERMINATED

                Recommended Actions
                CHECK CABLES AND THEIR CONNECTIONS
                INSTALL LATEST ADAPTER AND DRIVE MICROCODE
```

```
                  INSTALL LATEST STORAGE DEVICE DRIVERS
                  IF PROBLEM PERSISTS, CONTACT APPROPRIATE SERVICE
REPRESENTATIVE

          Detail Data
          ERROR CODE
          MAJOR/MINOR DEVICE NUMBER
          ADDITIONAL INFORMATION
          pdtx=0 sidx=0

---------------------------------------------------------------------------
```

6.27.3 System Dump Improvements

There are a number of improvements to the system dump process that are described in this section. Dumps are a snapshot of the state of registers and memory at a critical time such as when the kernel encounters an error condition from which it cannot recover. When a dump occurs, the kernel walks through a table of component dump routines, calling each in turn. Each routine locates the data required by its component and returns a pointer to a component dump table. This table contains entries describing the contiguous regions that need to be dumped for its component. The description includes pointer, length and segment ID of each region. After calling the component dump routine, the kernel checks every page described in the returned component dump table and builds a bitmap indicating each page’s status. Then, a header, the bitmap and those component pages which are currently in memory are written to the dump device. When all of this is done, there exists an image of what was considered important in memory when the crash occurred.

6.27.3.1 JFS Smart Dumps

In previous releases of AIX, disk allocation maps for all file systems were dumped, regardless of whether the system crashed because of file system corruption. A side effect of this is that if you have a large number of file systems mounted, you would get an extremely large dump size. Further, because other file system meta-data (for example, .indirect segment) was not dumped, certain classes of file system problems were difficult or impossible to debug.

Now, file system information, including control information, is only dumped if a file system is involved as part of the problem scenario that triggered the dump generation, such as the case of file system errors. The JFS code has also been reviewed to remove asserts (that is, failure scenarios that are trapped to crash the system) where the risk of further errors are very low.

Examination of file system meta-data is generally only necessary when a crash occurs due to file system corruption. This enhancement causes meta-data to be dumped only in cases of corruption. The inode map and .indirect and .inodes segments are dumped in addition to the disk allocation map. Since directory corruption is not fatal, directory segments are not dumped.

At any point where fatal corruption is detected, the paging device table (PDT) index of the offending file system is recorded in pinned storage so the VMM dump routine can process the special file system segments. The changes introduced in the JFS error logging enhancement, described in the section 6.27.2.2, “JFS Error Logging” on page 306, are also designed to work with this JFS dump change.

6.27.3.2 Support for a Mirrored Dump Device

Previous releases of AIX did not provide complete support for a mirrored dump device, which is a more common limitation since AIX Version 4.1 when the paging device became the default dump device. A mirrored rootvg is a popular configuration, but not all administrators would change the dump device to something other than the mirrored paging device. This dump device change was previously required to obtain a good dump for analysis.

If such a system (that is, previous AIX versions) dumped, the dump data was written to the first mirror copy only without any staleness indications being made. The data was there, but when it was read, LVM simply started to pass back randomly-selected mirror copies for each logical partition, so the data came back in a corrupted state.

This enhancement removes the first two steps in the above list from the problem analysis process, hence allowing for a faster resolution.

With AIX Version 4.3.3, you can use the `readlvcopy` command to read an individual mirror copy of any LV. Hence, `readlvcopy` is now used in the `snap` command to copy the dump data from the first mirror copy only, and thus obtain the original dump data without corruption.

The `readlvcopy` command is available at boot time so one mirrored device can be used for both dump and paging activity. There are still other restrictions that you should be aware of, many of which are described in the `sysdumpdev` man page in the AIX documentation:

- If a paging device is used for dumps, then this paging device must be part of rootvg. This is because rootvg is the only active volume group during the initial boot process, and the dump image must be copied from the dump paging device before its activated as a paging space.

- The primary dump devices must always be in the root volume group for permanent dump devices.
- The secondary dump device may be outside the root volume group unless it is a paging space device.

The `savecore` command has also been changed since it is used to copy a compressed dump from the dump device to a file. Finally, there are also changes to shell scripts behind the SMIT panels you will see from the command `smitty dump`:

Copy a System Dump from a Dump Device to a File.

Copy a System Dump from a Dump Device to a Diskette.

How to Create the Mirrored Dump Device

One scenario you may have is a default dump environment after an overwrite AIX installation. You can create a mirrored dump paging device by the following method:

1. Note that you can verify that you are using the hd6 paging device as the default dump device by the command:

```
# sysdumpdev -l
primary             /dev/hd6
secondary           /dev/sysdumpnull
copy directory      /var/adm/ras
forced copy flag    TRUE
always allow dump   FALSE
dump compression    OFF
```

2. The hd6 logical volume can be mirrored using the `mklvcopy` command:

```
mklvcopy -sn -k hd6 2 hdisk0
```

3. You can use the `lsvg` command to confirm the change. The following example shows only the data relevant for the hd6 logical volume:

```
# lsvg -l rootvg
rootvg:
LV NAME            TYPE       LPs   PPs   PVs  LV STATE      MOUNT POINT
hd6                 paging     16    32    1    open/syncd    N/A
```

Examining a Mirrored Dump

Having created a mirrored dump device, you can verify that it is correctly configured by forcing a dump and then using the `crash` command to examine the dump image. For this test scenario you can do the following:

1. Initiate a system dump. For example, use the `sysdumpstart` command.

   ```
   # sysdumpstart -p
   ```

2. Restart the system after the dump process has completed
3. Logon and verify that you are likely to have a good dump with the following `sysdumpdev` command:

   ```
   # sysdumpdev -L
   Device name:         /dev/hd6
   Major device number: 10
   Minor device number: 2
   Size:                26064384 bytes
   Date/Time:           Thu Jul 29 09:46:47 CDT 1999
   Dump status:         0
   dump completed successfully
   Dump copy filename: /var/adm/ras/vmcore.0
   ```

4. Examine the dump with the `crash` command. In this example, just the `stat` subcommand is used before exiting `crash`. The `crash` command sequence is:

```
# crash /var/adm/ras/vmcore.0
Using /unix as the default namelist file.
> stat
        sysname: AIX
        nodename: 433b
        release: 3
        version: 4
        machine: 006152004C00
        time of crash: Thu Jul 29 09:46:47 CDT 1999
        age of system: 11 min.
        xmalloc debug: disabled
        abend code: 0
        csa: 0x0
> quit
```

Compressed Mirrored Dumps

Extremely large dump devices are required on systems with large amounts of real memory (RAM). Systems with 16 GB of RAM may need 2 GB dump devices. The code change associated with AIX Version 4.3.2 APAR IX86629 provides an option to the `sysdumpdev` command that supports the compression of dumps as they are written to the dump device, thus cutting the required size of a dump device in half.

Using a Compressed Dump Device

Set up, test, and examine the compressed dump environment in a system maintenance window with the following sequence:

1. Check the estimated dump size with the sysdumpdev command.

   ```
   # sysdumpdev -e
   0453-041 Estimated dump size in bytes: 19922944
   ```

2. Enable dump compression with the sysdumpdev command. Note that you can disable dump compression with the sysdumpdev -c command.

   ```
   # sysdumpdev -C
   ```

3. Verify that the new estimated dump device is significantly smaller.

   ```
   # sysdumpdev -e
   0453-041 Estimated dump size in bytes: 9437184
   ```

4. Check the dump environment parameters with the sysdumpdev command. Check that the dump compression option is now ON.

   ```
   # sysdumpdev -l
   primary              /dev/dumps
   secondary            /dev/sysdumpnull
   copy directory       /var/adm/ras
   forced copy flag     TRUE
   always allow dump    TRUE
   dump compression     ON
   ```

5. Initiate a system dump. For example, use the sysdumpstart command.

   ```
   # sysdumpstart -p
   ```

6. When the system reboots, verify that you have a good dump with the sysdumpdev command.

   ```
   # sysdumpdev -L
   0453-039

   Device name:         /dev/dumps
   Major device number: 39
   Minor device number: 2
   Size:                5938688 bytes
   Uncompressed Size:   22132443 bytes
   Date/Time:           Wed Aug 11 17:11:34 CDT 1999
   Dump status:         0
   dump completed successfully
   ```

7. The dump must be written to a file and uncompressed to be read with the crash command. You can use the savecore command to write it to a file, with the -d flag to place it into another directory. Subsequent commands show you how to confirm that savecore produced the desired results.

   ```
   # savecore -d /home
   ```

8. You can check the output from the savecore command and uncompress it for crash with the following commands:

```
# ls -l /home/vmc*
-rw-r-----   1 root     system   5938688 Aug 11 20:07 /home/vmcore.0.Z
# uncompress /home/vmcore.0.Z
# ls -l /home/vmc*
-rw-r-----   1 root     system   22128102 Aug 11 20:07 /home/vmcore.0
```

9. Finally, you can use the crash command to verify that this is a good dump, since you don't get errors.

```
# crash /home/vmcore.0
Using /unix as the default namelist file.
> stat
        sysname: AIX
        nodename: itsosrv2
        release: 3
        version: 4
        machine: 000000131C00
        time of crash: Wed Aug 11 17:11:34 CDT 1999
        age of system: 1 hr., 5 min.
        xmalloc debug: disabled
        abend code: 0
        csa: 0x0
> q
```

6.27.3.3 Dump Progress Indication

As dumps grow larger, it takes longer to write them. Normally, the only indication that a dump is in progress is either a 0c2 or 0c9 value shown on the front panel display. With previous releases of AIX, you may have believed that the dump processing was hung because either 0c2 or 0c9 has been showing for a long period of time, when in reality the dump was still in progress. In these situations, if you press the reset button or the power switch, the dump is lost.

To avoid this, AIX now makes use of the second line of two line front panel displays, in RS6000 models, such as the model F50, that have a two line display, as shown in Figure 39 on page 327. This display shows the number of bytes written so far to the dump device. This serves two purposes:

- It will provide a visual indication that the dump is still progressing (or conversely, show if it has hung).
- It will give you a sense of how much longer the dump will take. (This is possible if you have previously obtained estimates of the dump size using the sysdumpdev -e command, or you have had other dumps previously.)

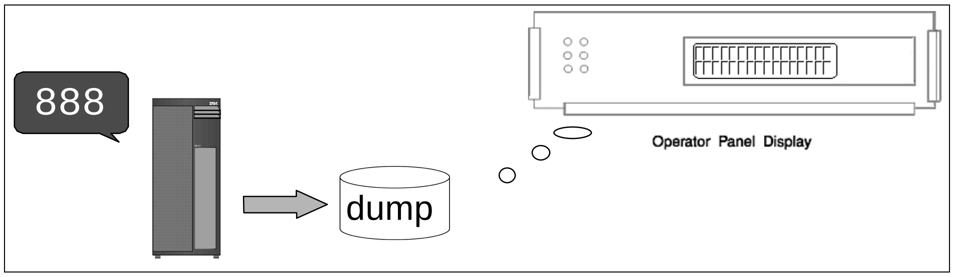


Figure 39. Dump Progress on the Front Panel Display

Many old systems only have three LED digits, most new systems come with a two line by 16 character display which is addressable using RTAS. Other systems that lack a two line display may attempt to display this status indication in some other manner. (For example, SP systems may reflect it back to the control workstation.) Otherwise, attempts to write a status line will return an error value, but there should be no other adverse effects.

6.27.3.4 Dumps Have More Loader Data

When analyzing crash dumps involving the system loader, there is sometimes insufficient data in the dump. In particular, the shared library load lists, which are in global segments maintained by the system loader, are not dumped. A dump routine has been added to dump these segments when a crash occurs. To avoid increasing the size of the dump by an unreasonable amount, an attempt is made to avoid dumping shared library text and data.

New options have been added to both the `crash` and `kdb` dump readers to allow indication that addresses for segments 11 and 13 are to be interpreted as loader data. Previously, these segments were assumed to be VMM segments. Therefore, when data is requested for either segment, for a dump, it was retrieved from the VMM component. You can now select whether the data is to be retrieved from the VMM component or the new loader components. This switch also supports access of the loader segments when running on a live system.

The `crash` and `kdb` dump readers have been updated to allow display of this data from dumps or from live systems. This loader data is displayed in both unformatted and formatted forms.

Table 31. New Dump Loader Data Subcommands

Dump Command	**New or Updated Dump Subcommands**		
crash	cm	od	le

Dump Command	New or Updated Dump Subcommands		
kdb	set	d*	lke

Chapter 7. AIX Workload Manager

AIX Workload Manager (WLM) is an operating system feature introduced in AIX Version 4.3.3. It is provided as an integrated part of the operating system kernel at no additional charge.

7.1 Overview

WLM is designed to give the system administrator greater control over how the scheduler and virtual memory manager (VMM) allocate CPU and physical memory resources to processes. This can be used to prevent different classes of jobs from interfering with each other and to allocate resources based on the requirements of different groups of users.

The major use of WLM is expected to be for large SMP systems, typically used for server consolidation, where workloads from many different server systems, such as print, database, general user, transaction processing systems and so on, are combined. These workloads often compete for resources and have differing goals and service level agreements. At the same time, WLM can be used in uniprocessor workstations to improve responsiveness of interactive work by reserving physical memory. WLM can also be used to manage individual SP nodes.

Another use of WLM is to provide a buffer between user communities with very different system behaviors. WLM can help prevent effective starvation of workloads with certain behaviors, such as interactive or low CPU usage jobs, from workloads with other behaviors, such as batch or high CPU usage.

WLM gives the system administrator the ability to create different classes of service, and specify attributes for those classes. The system administrator has the ability to classify jobs automatically to classes based upon the user, group, or path name of the application.

WLM configuration is performed through a text editor and AIX commands, or through the AIX administration tools SMIT or Web-Based System Manager graphical user interface.

These usages have two different issues that workload management must address. First, goals for amount of resources available to different workloads is required. These goals are not absolute, but the targets should be achieved over the long term to provide a degree of fairness over the long term. The second issue that workload management must address is boundaries on the amount of resources that a workload can receive. These boundaries can be in terms of maximum resources available and minimum resources that must be made available. These boundaries are not intended to be the major means of separating workloads, but are intended to address special situations where goals are not sufficient to provide sufficient isolation.

Like all system administration tools, workload management must be configured wisely. It is quite simple to produce configuration files that, once activated, cause the server to behave in a strange manner or even hung. If you give, for example, many CPU resources to a CPU bound class and leave few resources to the other processes, you may find that logging into the system is almost impossible.

On the other hand, a correct workload management may provide users the confidence of having a good response time even when other heavy operations are in act on the system. For example, you may configure the system to reserve a minimum real memory for data entry processes: the application will normally have its working pages in memory and the user will not have to wait for pages to be swapped in.

7.2 Concepts and Configuration

WLM monitors and regulates the CPU utilization of threads and physical memory consumption of processes active on the system. The manner in which the resources are regulated is dependent on the WLM configuration defined by the system administrator.

There are a number of controlling variables in WLM that facilitate managing classes of jobs to achieve the automatic application of resource entitlement policy you define. The primary concept to remember is that classes are what you manage in WLM, and that there are three job attributes available for process identification: users, groups, and applications/jobs. Class resource shares and class resource limits allow you to define resource entitlements for each class, and tiers allow you to group the classes you are managing.

A view of the basic elements of WLM is shown in Figure 40, and discussed in the sections that follow.

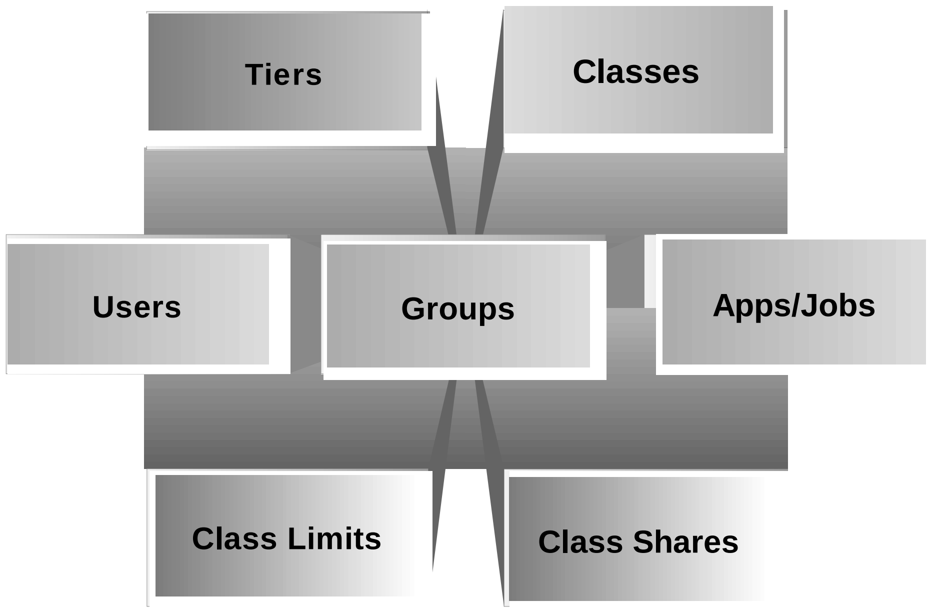


Figure 40. Basic WLM Elements

7.2.1 Job Attributes

The following is a discussion of the attributes for a job.

7.2.1.1 Users

The user name (as specified in the password file) of the user owning a process can be used to determine the class to which the process belongs. Users can be excluded by using an exclamation point (!) prefix. Applications which use the setuid permission to change the effective user ID they run under are still classified according to the user that invoked them.

7.2.1.2 Groups

The group name (as specified in the group file) of a process can be used to determine the class to which the process belongs. Groups can be excluded by using an exclamation point (!) prefix. Applications which use the setgid permission to change the effective group ID they run under are still classified according to the group that invoked them.

7.2.1.3 Applications/Jobs

The full path name of the process being executed can determine the class to which a process belongs. The full path name can include patterns with the same style as in the Korn shell. Applications/jobs can be excluded by using an exclamation point (!) prefix.

Combinations of user, group, and applications/jobs attributes may be used to determine the class.

7.2.2 Categories

The categories within WLM are discussed in the following sections.

7.2.2.1 Classes

Up to 27 classes except Default and System classes can be defined by the system administrator. Two classes and two pseudo-classes are reserved as the following.

There are two predefined classes that cannot be removed.

• Default class

  The default class is named *Default* and is always defined. All non-root processes that are not automatically assigned to a specific class will be assigned to the Default class. You can also assign other processes to the Default class.

• System class

  This class, named *System,* will have all privileged (root) processes assigned to it if not assigned by rules to a specific class, plus the pages of all system memory segments, kernel processes, and kernel threads. You can also assign other processes to the System class. The default is for this class to have a memory minimum limit of 1%.

And, in the first release of WLM, there are two pseudo-classes, named "*Shared"* and *"Unclassified*". No classification rules, resource limits, or resource shares can be specified for these classes. Pseudo-classes are outside WLM control and therefore fall under default AIX resource allocation control. Under default control, if the demands of unclassified processes are great enough, they could potentially starve processes under WLM management.

• Shared pseudo-class

  This pseudo-class receives all memory which is used by processes in more than one class. This includes pages in shared memory regions and pages in files that are used by processes in more than one class. Shared memory and files that are used by multiple processes that all belong to a single class are associated with that class. It is only when a process from a different class accesses the shared memory region or file that the pages are placed in the shared class.

- Unclassified pseudo-class

  This pseudo-class receives all processes which cannot be classified. Wait processes and processes that have called the plock system call will be in Unclassified pseudo-class. Note that in the first release of WLM, processes that are already in existence at the time that WLM is initialized will be in the Unclassified pseudo-class.

The attributes that must be defined for a class are

- Class Name (up to 16 characters in length)
- Tier (zero is the default)
- Resource shares
- Resource limits

When a process is executed, it will be classified into one of the defined classes based on the class assignment rules pertaining to the user, group, or application.

7.2.2.2 Tiers

Tiers are based on the importance of a class relative to other classes in WLM. There are 10 available tiers from 0 through to 9. Tier value 0 is the most important; the value 9 is the least important. As a result, classes belonging to tier 0 will get resource allocation priority over tier 1; tier 1 will have priority over tier 2, and so on. In the first release of WLM, tier resource enforcement is primarily directed at CPU utilization.

7.2.3 Resources

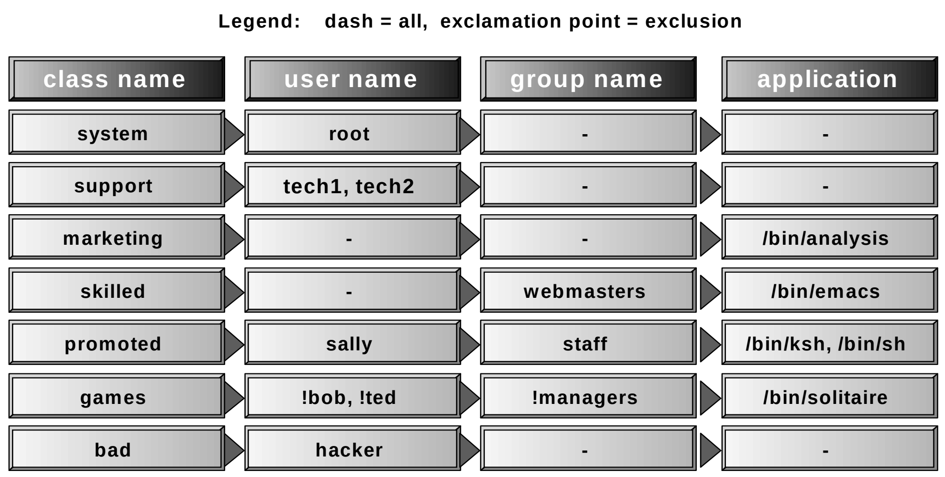
The various resources within WLM are discussed in the following sections.

7.2.3.1 Class Assignment Rules

There are several application behaviors that can lead to difficulties separating an application into different classes and managing them separately.

- The application runs as a single process, possibly with multiple threads. WLM places all threads in a process in the same class.
- The application uses multiple processes, but they all execute the same executable file and always run as the same user and group. WLM can only use the user, group, and application to classify processes, so all these process would be placed in the same class.

After a class has been defined, class assignment rules should be created. The assignment rules are used by WLM to assign a process to a class based on the user, group, application or a combination of these three attributes. The exclamation point (!) as the lead character represents an exclusion. The "!" can be used with all attributes: user, group, and application names. Note: In the first release of WLM, a process that is already running when WLM is initialized will be placed in the Unclassified pseudo-class.



Legend: dash = all, exclamation point = exclusion

class name	user name	group name	application
system	root	-	-
support	tech1, tech2	-	-
marketing	-	-	/bin/analysis
skilled	-	webmasters	/bin/emacs
promoted	sally	staff	/bin/ksh, /bin/sh
games	!bob, !ted	!managers	/bin/solitaire
bad	hacker	-	-

Figure 41. Example of Classes and Class Assignment Rules

7.2.3.2 Class Resource Shares

The number of shares of a resource for a class determine the proportion of a system resource that is allocated to the processes assigned to the class. In simple terms, the resource shares are specified as relative amounts of usage between different classes.

- The default share is 1.
- The allowed share values for one resource for on class are integers from 1 to 65535.
- A class is active if it has at least one process assigned to it.

System resources are only allocated to a class with active processes, so resource percentages are calculated based on the total number of shares requested by all active classes. The reason for using resource shares rather than percentages is that the desired amount is automatically recalculated when new resource shares are added (when a new class is created) or when resource shares are unused (because a class has no processes). The calculation of entitlement is a percentage equal to the class resource shares divided by the total shares of all active classes in the same tier.

However, WLM makes sure that the calculated percentage goal for a resource remains compatible with the minimum and maximum limits for the resource. If the calculated percentage is below the minimum, WLM uses the minimum as the resource shares. If the percentage is above the maximum range, WLM uses the maximum as the resource shares. If the percentage is between the minimum and maximum, WLM uses the calculated value.

Resource shares are specified in the resource shares file. The resource shares are listed by resource type within stanzas for each class.

The following example (Figure 42) displays resource allocation before and after a new class is activated. Initially there are three active classes which have been allocated 5, 7, and 2 resource shares respectively. These resource shares combined are allocated 100 percent of the resource in accordance with their relative share values. When the new class, which has three resource shares, is activated, there are four active classes with resource shares of 5, 7, 2, and 3 with the total active resource shares equal to 17. As a result, when all four classes are active, the class with five resource shares will be allocated 5 of 17, or 29 percent of the system resource (29.4 percent will be rounded down to 29 percent).

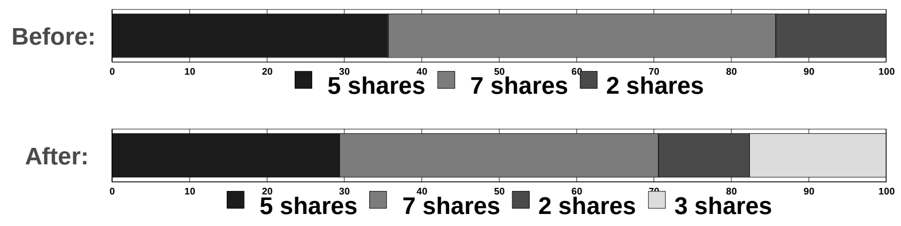


Figure 42. Example of Share Distribution Automatically Adjusting Resources

7.2.3.3 Class Resource Limits

The class resource limits define the minimum and maximum amount of a resource limits (in the first release of WLM, for CPU and memory) that may be allocated to a class as a percentage of the total system resource.

Instead of having the minimum limit *reserve* a percentage of the resource for a given class, a higher resource allocation priority is given to processes in classes that are getting less than their minimum limit of the resource. This makes it more likely that these classes will get the resource if they try to use it. For CPU and memory, if a class is not using the resource at least up to its minimum, the rest is allocated to other classes. WLM cannot guarantee that processes actually reach their minimum limit. This depends on how the processes use their resources and on other limits that may be in effect. For example, a class may not be able to reach its minimum CPU entitlement because it cannot get enough memory. The possible minimum values are integers from 0 to 100. The default value is 0.

A maximum limit restricts the amount of resource that may be utilized by a given class. The possible maximum values are integers from 1 to 100. The default value is 100.

Resource limit values are specified in the resource limit file by resource type within stanzas for each class. The resource limits are specified as a minimum to maximum range, separated by a hyphen (-) with whitespace ignored. Each resource limit value is followed by a percent sign (%).

Class resource limits follow some basic rules.

- Limits take precedence over class share values
- The minimum limit must be less than or equal to the maximum limit.
- The sum of all minimum limits for classes in the same tier cannot exceed 100 percent.
- For CPU, a class use of CPU cycles may exceed the maximum limit.

7.2.4 Resource Handling for the Unclassified Class

The Unclassified class' resources cannot be explicitly controlled. However, WLM does have to make choices about resources that are used by the Unclassified class.

For CPU usage, WLM does not adjust the scheduling priority of Unclassified work. This work has its scheduling priority modified by the scheduler based upon the CPU usage of each thread, exactly as happens when WLM is not invoked. CPU usage by the Unclassified class is not counted toward the total amount of CPU usage available to other classes.

For physical memory usage, the Unclassified class is treated the same as a class that is below its desired resource share value, even though it contributes no shares to the total. Thus, pages will tend to be taken away from classes that are over their desired resource share value and given to classes that are below their desired resource share value and thus also to the unclassified work. Thus, Unclassified work is somewhat favored for memory usage relative to other classes.

7.2.5 Interaction with Other Scheduler Control Mechanisms

The `nice` command causes a process to have its CPU usage selectively favored or penalized with respect to other processes in the system. This effect will also work in a WLM environment. The nice command will cause a process to have its CPU usage selectively favored or penalized with respect to other processes in the same class as the process. The nice command will not affect the CPU utilization of processes in other classes, because WLM will work to have the class' resources meet the requested number of resource shares and resource limits.

The CPU utilization of fixed priority processes is not managed by WLM. This is because setting fixed priority for a process (which can only be done by root) is designed to prevent the scheduler from changing the priority of a process. This is generally done because the process is so important that it must be fixed at a priority higher than the other processes in the system can achieve.

The `schedtune` command can be used to modify the behavior of the scheduler. All options to schedtune continue to work in a WLM environment. The use of schedtune options will not significantly impact the ability of WLM to manage CPU usage.

7.2.6 Interaction with Other Physical Memory Control Mechanisms

The `vmtune` command can be used to modify the behavior of the Virtual Memory Manager (VMM). All options to vmtune continue to work in a WLM environment. Some of the options to vmtune, particularly minperm and maxperm, can hamper WLM's ability to achieve the specified physical memory usage goals.

7.3 WLM Administration

WLM can be administered using three different methods.

- Web-Based System Manager graphical user interface, initiated with the AIX command `wsm wlm`
- System Management Interface Tool (SMIT), initiated with the AIX command `smit wlm`
- Command line and file editing

The workload management is not active at boot time and must be explicitly activated executing the `wlmcntrl` command, which is normally done by an entry in /etc/inittab, when it is not commented.

The classification of processes is made only when they are first executed. This means that processes that were active before workload management is activated are not accounted in the classes defined by the system administrator but are all collected the Unclassified class. All new processes are classified accordingly to the workload management configuration files. In order to have all user processes correctly classified at boot time is then important to have workload management started in early stages in /etc/inittab.

If workload management is stopped, then all previous classification is lost and all processes are put in the Unclassified class.

Reclassification of existing processes is not actually possible so care must be taken when first classification is done. Once `wlmcntrl` is executed, the only possible change in class configuration is the resource metrics, either changing the minimum, the maximum or the share value.

If a class needs to be generated or changed, workload management must be stopped and restarted. This operation, though, causes all running processes to be accounted in the Unclassified class: in order to have those processes reclassified, they need to be restarted.

Reclassification of Processes

A PTF has been scheduled in order to make workload management classify processes running before `wlmcntrl` is issued. It should be available at the end of 1999. If you plan to use workload management we suggest to look for PTF availability.

After PTF is applied, the position of `wlmcntrl` in the /etc/inittab file will not be important since processes executed previously will be classified according to workload management rules.

Only class boundaries may be modified on the fly without stopping and restarting workload management. That is only limits and targets may be changed dynamically.

7.4 Configuration Files

The default workload management configuration files location is /etc/wlm/current directory which is normally a soft link to the directory that actually contains the files. You can maintain the soft link using command line, SMIT, or the Web-Based System Manager thus having multiple configurations that can be activated one at a time. You can also use different locations but you must then use only command line and specify each time the directory containing the files.

The system administrator may prepare different configurations that refer to different workload configurations that make occur on the system. Some cron jobs may be created, for example, to activate a configuration during weekdays and another during weekends.

The configuration files are:

- classes
- limits
- shares
- rules

Except for the rules file, they are all formatted by stanzas whose name is the class name and whose content represent an attribute-value pair. The attribute that can be used depends on the file. Comments may be introduced in each file provided that the comment line begins with an asterisk.

The *classes* file defines the name of the classes and may have as optional attributes the tier value and a short description. You can create up to 29 new classes. An example of the file is the following:

```
System:

Default:

production:

test:
        description = "Low privilege class for testing activities"
        tier = 3
```

If no tier information is provided, the class is supposed belonging to the tier 0, that is the one most important. Tiers value are in the range from 0 to 9.

Two classes are always present and must never be deleted from the classes file:

- Default class: all *non-root* processes not assigned to a specific class
- System class: all *root* processes not assigned to a specific class, pages of system memory segments and kernel threads

Default and System class represent all those processes that are started after workload management and are not explicitly classified by the system administrator.

The *limits* file describes the range of resources that are given to classes, that is the minimum and maximum percentage value of the resource:

```
System:
       memory = 1%-100%

test:
       CPU    = 5%-35%
       memory = 10%-50%
```

In this example, the test class will receive from 5 to 35% of CPU time and from 10 to 50% of memory. If no limits are provided in the file, a default minimum value of 0% and a maximum of 100% is always assumed.

It is possible to provide limits both to System and Default classes. Normally the Default stanza is not present but you can add it like in the following example:

```
Default:
       CPU     = 5%-90%
       memory  = 10%-80%
```

The *shares* file defines the importance of the class in resource allocation. The more shares a class has, the higher resource target the class has. An example of file syntax is:

```
production:
       CPU    = 30
       memory = 60

test:
       CPU    = 10
       memory = 20
```

The single value of a share is not really important, but the relative values of shares among different classes count. In our example, the shares file tells that production test should have three times the CPU and three times the memory of the test classes. Remember, though, that the test class belongs to a lower priority tier of production, so resource allocation is even more complex.

The *rules* file contains all the rules needed to classify each process (with its threads) into the defined classes. Each line of the file is a rule and is composed of five fields:

- class name
- reserved field for future releases
- user name
- group name
- process name

The first field must match an already defined class name, while the other fields may contain values or a dash (-) that means that no requirement is made on that field.

In the user and group name you can use a single name, a list of named divided by a comma, or a name proceeded by a exclamation point that means exclusion. In the application field you can use shell wildcards.

Since processes do not inherit the parent process' class, you have to specify all the process names you want to classify and not only, for example, a shell script that starts all the application servers.

Rule ordering is important as processes are classified based on the first rule that matches their name, user and group. When using SMIT or Web-Based System Manager it is possible to define rules and put them in a specific position inside the rules file.

An example of rules file is:

```
*class       resvd      user       group      application
myclass      -          marc,john  -          -
dataclass    -          -          -          /usr/lpp/bin/oracle
backupclass  -          !root      -          /usr/bin/restore
```

In this case:

- If marc runs anything, including oracle or restore, it will be assigned to class *myclass*
- If root runs oracle, it will be in class *dataclass*
- If root runs restore, it will be in the *Default* class
- If eric runs restore, it will be in class *backupclass*

In the application field full path names must be provided and must reference files that exist when workload management is started. If this is not true, the call of `wlmcntrl` will fail giving the name of the faulty role:

```
# wlmcntrl
A file or directory in the path name does not exist.
1495-036 Cannot stat file /tmp/i_do_not_exist (rule at line 17)
```

This problem may easily arise if the file system where the file is located is not mounted, for example for NFS problems.

The file may be a soft link to another file, but once the workload management is activated, it gets the file vnode so it classifies only the process that is pointed at that moment. If you change the link or delete the link, workload management is not affected.

Set user id (SUID) and set group id (SGID) attributes are not taken into account. In order to classify processes, workload management only considers the user id and the group id of the user that executes the program.

7.5 WLM Commands

There is a set of commands that are used to manipulate the workload management configuration files (`mkclass`, `lsclass`, `chclass`, `rmclass`) and commands to activate, update, stop and monitor the workload management (`wlmcntrl`, `wlmstat`). These commands are only provided for use by the Web-Based System Manager and the SMIT administration tools. The Web-Based System Manager is the preferred method of administrating WLM.

The configuration files may be changed either using an editor or executing the appropriate command but two consideration must be done:

1. The manipulating commands always rewite the modified file reformatting its contents and removing any comment manually added. A copy of original file is put in the workload management directory with extension '.old'.
2. No global coherency checking is made. If syntax errors are present or the following requirement are not met, the activation of workload management fails and error must be resolved manually:
    - minimum resource must be in range from 0 to 100
    - maximum resource must be in range from 1 to 100
    - share value must be in range from 1 and 65535
    - minimum resource must be less or equal to maximum resource

- Default and System classes must exist
- System class must have a non-zero minimum value
- The sum of all minimum values for a resource in the same tier must not be more than 100%

The SMIT and Web-Based System Manager interfaces make use of manipulation command so they can delete from the files your comments. This does not cause data corruption.

Any change to configuration files, either by hand or by command, is not reflected in workload management behavior but will be used only by the next successful execution of `wlmcntrl` command.

7.5.1 chclass

This command changes a class and its limits. Its syntax is

```
chclass    {[-a Attribute=Value]...}        \
           [-c|-m <limit>=<value>]          \
           [-d  Config_dir]                 \
           <class>
```

where:

-a Attribute=Value	is used to change tier and description attribute
-c or -m limit=value	changes limits (min or max keyword) or share (shares keyword) values of CPU (-c flag) or memory (-m flag). The value is an integer.
-d Config_dir	Use this configuration directory instead of default /etc/wlm/current
class	class to be changed

In the following example, we change the tier value and the CPU share value of class test

```
chclass -a tier=1 -c shares=45 test
```

7.5.2 lsclass

This command lists class information. Its syntax is

```
lsclass    [-C|-f] [-d <config_dir>][<class>]
lsclass    -D [-d <config_dir>]
```

where:

--C	Displays the class attributes and limits in colon-separated records
-f	Displays the output in stanzas, with each stanza identified by a class name. Each Attribute=Value pair is listed on a separate line.
-d Config_dir	Use this configuration directory instead of default /etc/wlm/current
class	Class to be shown
-D	Returns the banner and the default values

If no class name is specified, all classes are listed (with attributes if -C or -f is used), otherwise only selected one is displayed.

7.5.3 mkclass

This command creates a new class. Its syntax is:

```
chclass    {[-a Attribute=Value]...}         \
           [-c|-m <limit>=<value>]           \
           [-d  Config_dir]                  \
           <class>
```

where:

-a Attribute=Value	Defines tier and description values
-c or -m limit=value	Defines limits (min or max keyword) or share (shares keyword) values of CPU (-c flag) or memory (-m flag). The value is an integer.
-d Config_dir	Use this configuration directory instead of default /etc/wlm/current
class	class to be created

In the following example, we create the class Development with tier 2, Max CPU 60% and CPU shares 10.

```
mkclass -a tier=2 -c shares=10 -c max=60 Development
```

The values not defined are not placed in the configuration files but have as an effective value the default value you can display with `lsclass -D`.

7.5.4 rmclass

This command removes a class. Its syntax is

```
rmclass    [-d <config_dir>] <class>
```

where:

-d Config_dir	Use this configuration directory instead of default /etc/wlm/current
class	Class to be removed

You cannot delete the Default or System class.

7.5.5 wlmcntrl

This is the command that start, updates and stops the workload management. Its syntax is:

```
wlmcntrl [-d <config_dir>] [-u|-o|-q]
```

where:

-d config_dir	Use this configuration directory instead of default /etc/wlm/current
-u	Updates the existing class definitions
-o	Turns off the workload management
-q	Queries if workload management is active or not. The command returns 0 if it is running, 1 if it is not.

When the command is launched with no option or only with the -d option, it starts the workload management reading the configuration files in the standard directory or in the directory provided with -d option.

If any definition in the configuration file is wrong when starting workload management, an error message is printed on standard error and the `wlmcntrl` command exits.

In AIX 4.3.3 the only allowed updates to workload management configuration are the resource boundaries of classes that is:

- minimum value
- maximum value
- shares value

7.5.6 wlmstat

This command displays statistical data on workload management. Its syntax is

```
wlmstat [-l class_name] [-c|-m] [interval] [count]
```

where:

-l class_name	Restricts the view on class_name only
-c	Display CPU data only
-m	Display memory data only
interval	Display interval in seconds (default 1)
count	Number of reports (default 1).

The statistics shown display the actual usage of CPU and memory of each class defined in the system, including Default, System and Unclassified classes.

7.5.7 Modification of ps command

In order to display for each process in the system to which class it belongs, a new format (*class*) has been added to ps when used with the -F or -o options. An example of the class parameter usage is the following:

```
# ps -o pid,class,args
  PID             CLASS COMMAND
12424      Unclassified -ksh
17552      Unclassified ksh ./cpu_bound
18086         cpu_bound ksh ./cpu_bound3
19370         cpu_bound ksh ./cpu_bound2
20128            System ps -o pid,class,args
```

In the previous example the ps command has been executed by root, so the ps process has been classified as System. The 12424 and 17552 processes are not classified because started before the activation of workload management.

7.6 CPU Management

Three values are used to manage CPU consumption: *Minimum limit, Percentage goal,* and *Maximum limit*.

Minimum limit is the percentage value of CPU usage that should be given to a specific class. It is defined in the limits configuration file. If a class consumes less than minimum limit, it will become more favored by the system.

Percentage goal represents the CPU usage that the class should have in the system. The goal is to keep classes' CPU usage in the range from minimum limit to percentage goal in average. If threads do not need CPU, then real

CPU utilization can be less than minimum limit; if other classes do not need CPU, then a class may consume more than percentage goal.

The per class CPU shares value is used by WLM to compute every second a per class percentage goal. This percentage goal is calculated as the quotient of the class shares divided by the sum of the shares of all the active classes in the same tier. The Unclassified pseudo-class does not participate in the percentage goal computation and always has a percentage goal of 100%. The threads in the Unclassified pseudo-class are never influenced by WLM.

When the CPU consumption of a class exceeds the maximum limit, the threads in this class have their priority value changed to the maximum value of 126, and any other thread becomes more favored. The maximum limit for CPU is not strictly enforced and threads are not stopped: if they ask for more CPU they may have it if there are no more favored threads waiting for CPU availability.

The class CPU usage is a value that is computed at a class level and represents the average over time of the usage of all threads belonging to the class, as a percentage of the total CPU cycles available.

The class CPU usage is computed by averaging over the last five seconds the percentage of the total CPU activity consumed by all the threads in the class. This is the value shown by the `wlmstat -c` command. If you monitor the value and launch a CPU intensive job, the value increases slowly due to the five second averaging and then reaches the effective CPU usage - When stopping the job, the value remains high for some seconds and then decreases.

Once WLM is activated, new processes (except those running at fixed priorities) are classified and the effective CPU usage of the class is compared to the minimum limit, the percentage goal, and the maximum value to determine which threads should be favored and which should be penalized. This adjustment is done by modifying the thread scheduling priority. The thread scheduling priority is calculated as the sum of the following parameters:

- standard thread priority (varies depending on the thread’s CPU usage)
- process nice value
- tier priority adjustment
- class priority adjustment

The standard thread priority and the process nice values are treated exactly as in AIX version 4.3.2.

The tier priority adjustment is a static value derived from the tier value of the class and it is set to be 4 times the tier value.

The class priority adjustment (or DELTA) is computed every second and it is the same value for all the threads belonging to the same class. It is defined using the following values:

- Average CPU consumption in the last 5 seconds (CPU_C)
- Value of CPU_C in the previous second (CPU_LAST)
- DELTA = CPU_C - CPU_LAST

The class priority adjustment depends on the value of the class CPU usage compared to the minimum limit, percentage goal, and maximum limit of the class. Four zones are defined for the class CPU usage as shown in Figure 43 on page 348 and the algorithm for determining the class priority is the following:

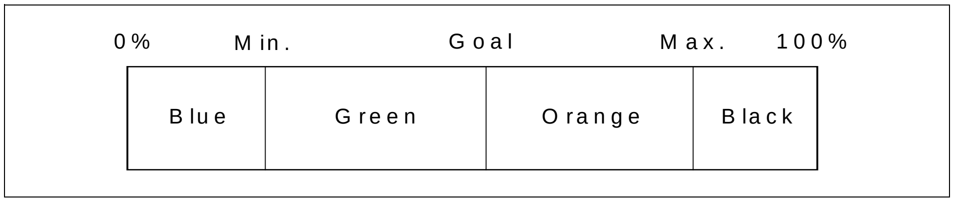


Figure 43. CPU Usage Zones

- In the Black zone, the class priority adjustment is set to 66 to block the thread
- In the Orange Zone:
    - If DELTA<=0, CPU usage is decreasing, so the class priority adjustment is unchanged.
    - if DELTA>0, CPU usage is increasing, so the class priority adjustment is incrementing by the maximum of one and half the DELTA value.
- In Green zone or Blue zone
    - If DELTA>0, CPU usage is increasing, so the class priority adjustment is unchanged.
    - if DELTA<=0, the class is favored by decremented the class priority adjustment by the maximum of one and half the DELTA value.

In any case, the thread’s minimum priority value is 40 and maximum priority value is 126. This limitation is made to avoid interference with fixed priority threads (they generally have priority values less than 40) and to make the thread use the CPU if no other thread is active.

Process swap out in thrashing condition has also been modified to select processes to be swapped according to their class' tier value. Each second, statistics on repaging rate are collected and processes posted for suspension are selected among those in the lowest tier that have a high repaging rate. If not enough processes in this tier have a high repaging rate, then all the processes in the tier can be swapped-out. When all the processes of a tier are swapped-out, the next lower numbered tier is examined.

7.7 Real Memory Management

Real memory consumption is controlled by system administrator providing for each class the *Minimum limit*, *Percentage goal*, and *Maximum limit* thresholds.

Minimum limit is the minimum percentage of real memory a class is guaranteed to have even in case of memory contention.

Maximum limit is the maximum percentage of real memory a class is allowed to consume. A class will not get more.

The percentage goal is a value between minimum limit and maximum limit and it is not strictly enforced. It is defined as the average memory usage the class should have. Its value is computed according to the *shares* value defined for memory in the WLM configuration files. The computation is identical to the computation for the CPU percentage goal explained earlier.

Memory accounting is done at the segment level: all the used physical pages of a segment belong to the same class. This is true for either working or persistent pages. If a segment is shared (referenced) by processes belonging to different classes, its pages are charged to a special pseudo-class named *Shared*.

WLM manages the classes depending on their real memory consumption compared to the minimum limit, percentage goal, and maximum limit values. A color is given to each class according to the following rules (see also Figure 44 on page 350):

- blue if it uses less than the minimum limit
- green if usage is between the minimum limit and the percentage goal
- orange if usage is between the percentage goal and the maximum limit

• red if usage is near the maximum limit

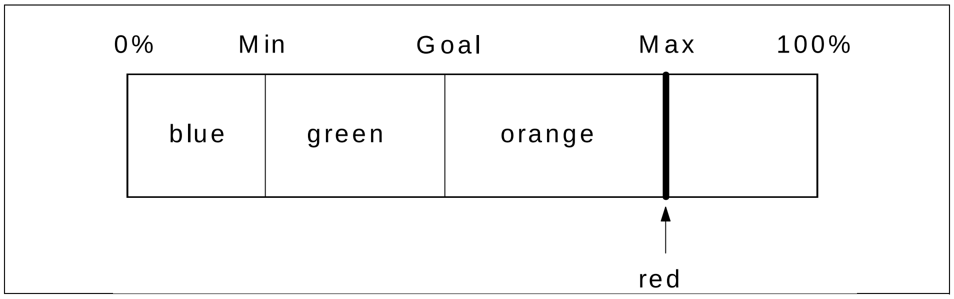


Figure 44. Memory Usage Zones

There are several special cases. Classes with a minimum limit and percentage goal equal to 100% are always blue. Classes with a minimum limit and percentage goal of 0% and a maximum limit of 100% are always orange.

Colors are ordered by temperature: blue is the coolest (most need resources) and red is the hottest (least need resources).

Memory pages are identified by their class' color, so all memory pages that belong to orange classes are called orange pages. The only exception is for working pages that belong to swapped out processes: they are always considered to be orange.

In addition to these colored pages, there are other pages that are not considered by WLM. They are pages that belong to processes that are not classified or pages that are shared among classes. They are always considered to be green pages.

The VMM behaves differently according to the color of pages. Free pages are accessible only to classes that are below their maximum memory limit. If a thread in a class which has reached its maximum memory limit requests a page (page faults), the page will be given only when enough pages have been stolen to the class to bring it back below its maximum limit, even when there are free memory pages available. This means that with WLM on, the page replacement algorithm can run even when there are plenty of free pages available, if classes are at their maximum memory limit. This can degrade the performance of the whole system. This is why memory maximum limits should be used carefully.

When page stealing is in progress, pages are stolen from each color to keep a certain number of pages of each color free. When pages are needed by a process in class, the class can only choose pages of a color that are at least as *warm* as the class is. For example, a process in a class that is orange can only use free pages that were originally orange or red. The free pages that were originally blue or green cannot be used for the process.

7.7.1 Tier Handling

Overcommitment of memory resources may arise through the usage of multiple tiers. The sum of all the minimum limit values for each single tier is enforced during WLM start to be less than 100%, but it may happen that the sum of the minimum limit values of active classes that belong to different tiers is more that 100%. A class is active when it has at least one process assigned to it.

Summing the minimum limit values of active classes starting from the most important tier (tier 0) it is possible to detect from which tier the sum exceeds 100%. The classes in that tier and the higher numbered tiers are considered to be orange regardless of their actual memory consumption. This is done to prevent overcommitment of memory and make sure that the active classes in the lower numbered (higher priority) tiers can have at least their minimum memory requirements satisfied.

7.7.2 Memory Classification

Memory segments are classified at the time of page fault:

- when a segment is first created, it is marked as belonging to the Unclassified pseudo-class.
- when a page fault occurs.
- if the segment was Unclassified, it will be marked as belonging to the class of the faulting process.
- if the segment belongs to same class as the faulting process, it remains in the same class.
- if the segment belongs to a different class than the faulting process, it is marked as belonging to the Shared pseudo-class

File segments are classified on page faults according to the rules described above. The fact that the reclassification occurs on page fault implies that a thread belonging to a different class than the segment can initiate reads and writes and not cause declassification (to Shared) as long as the pages it accesses are in memory.

For I/O operations like asynchronous I/O and NFS, the thread that at page fault time owns the segment is not the one that requested the access. In these cases it must be noted that the segments are charged to the class of the daemon that performs the I/O.

Pinned memory pages are not included in memory management. The color that is assigned to each segment and classes is defined according to the number of non-pinned real memory pages available. The computation is made using the following formula:

$$\text{class ratio} = \frac{\text{class pages}}{\text{physical memory} - \text{pinned pages}} \times 100$$

The color of the class is defined so that

- if class is swapped out: orange
- if class is in a tier that causes overcommitment (as defined earlier): orange
- if class ratio < the minimum limit: blue
- if the minimum limit <= class ratio < percentage goal: green
- if percentage goal <= class ratio < the maximum limit: orange
- if class ratio >= the maximum limit: red

Note: If a process allocates a large number of pinned pages, less memory is available for WLM. The class that pinned memory gets cooler and all other classes become hotter.

7.7.3 Memory Allocation

Page replacement is implemented per color and colored thresholds are defined using the minfree and maxfree system wide values.

The color of the thread that has generated the page fault defines the thresholds to which the actual number of free pages in the system (NUMFRB) is compared when making the decision about page allocation:

- if the number of free pages > color’s minfree, a page is allocated from free list
- if the number of free pages < color’s minfree, page replacement is started and a page is allocated

- if the number of free pages < maxfree of next cooler color, the thread sleeps on the wait list and a page replacement is started

Consider, for example, the situation described in Figure 45 on page 353.

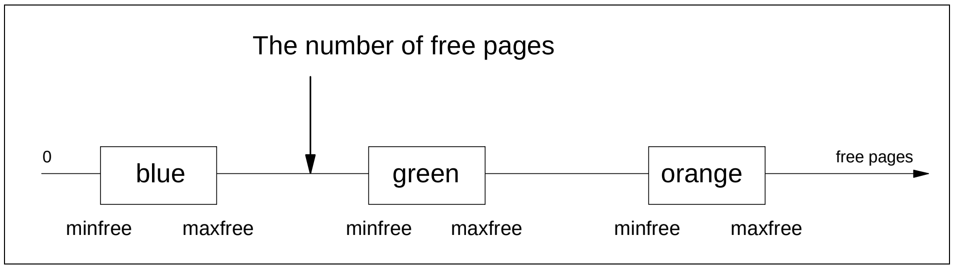


Figure 45. Memory Allocation

In the case of a page fault, a different action is started depending on the color of the thread that caused the page fault. If the thread is blue, the number of free pages is greater than the blue minfree, so the thread gets the page. If the thread is green, there are less free pages than the green minfree value, so the threads gets the page and the VMM starts looking for another page to be put on the free list. If the thread is orange, there are less free pages than the minfree value of the next cooler color (green), so the VMM starts looking for a new page for the free list, but no page is given to the thread that is put in the wait list.

As long as there is no memory starvation (the number of free pages is greater than any colored minfree and maxfree values), a class may grow in memory till it reaches its maximum memory limit value. A thread of a class that has reached its maximum memory limit and causes a page fault becomes red, so it is put to sleep and page replacement is started.

7.7.4 Memory Replacement

When VMM searches for new pages to be added to the free list, it attempts to steal currently used pages. This algorithm has been modified in AIX version 4.3.3 to be color sensitive. The search color defines the type of pages that are scanned to be aged and stolen: only pages with that color or a hotter one will be considered by this algorithm.

The total number of free pages needed is the number of pages currently free plus the number of new pages that need to be freed by the algorithm. The total number of free pages needed gives the initial value of the search color:

The initial search color is determined primarily by of the number of free pages available relative to the minfree values for each color.

The search color is not fixed. The algorithm monitors the number of pages scanned to find a new page and additional requests during the scan and may decide that the search color should be changed to a cooler or a hotter one.

7.8 Very Simple Examples

Workload management configuration may become very complex and system behavior may seem strange at a first look. In this section we provide two extremely simple examples of what may happen when workload management is activated and how the system reacts.

The examples have been studied to represent extreme situations and they make only use CPU and memory limits in order to provide easy to reproduce results. In a normal environment you will normally have many different classes and many processes and workload management will probably be made using shares.

The following cpu_bound script is an infinite loop that makes some computation and makes no I/O. It is a CPU bound process but it does not saturate the CPU.

```
#!/bin/ksh

I=0
while true
do
        (( I = I + 1 ))
        (( I = I - 1 ))
done
```

When executed with workload management turned off it uses almost all CPU and its priority value changes from 90 to 120 due to the fact it consumes much CPU. You can see those values using `ps` and `vmstat` commands:

```
# ps -efo class,pri,ni,args | egrep "cpu_bound|COMMAND"
               CLASS PRI NI COMMAND
        Unclassified 102 20 ksh ./cpu_bound
#
# vmstat 2 3
kthr     memory             page              faults        cpu
----- ----------- ------------------------ ------------ -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa
 0  0 15072  2306   0   0   0   0    2   0 120  460  64  2  1 96  1
 1  0 15072  2305   0   0   0   0    0   0 106   88  26 99  0  0  0
 1  0 15072  2305   0   0   0   0    0   0 106   50  25 99  1  0  0
```

If a new class cpu_bound is created with Max CPU value of 20% and a rule is added to make the cpu_bound script to belong to that class, the activation of workload management changes the system behavior.

The process still uses all the available CPU, thus going over the 20% limit of its class, but its priority value remains fixed to 126 and never changes, as you can see in the following `ps` and `vmstat` outputs:

```
# ps -efo class,pri,ni,args | egrep "cpu_bound|COMMAND"
               CLASS PRI NI COMMAND
           cpu_bound 126 20 ksh ./cpu_bound
#
# vmstat 2 3
kthr     memory             page              faults        cpu
----- ----------- ------------------------ ------------ -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa
 0  0 15074  2280   0   0   0   0    2   0 120  460  63  2  1 96  1
 1  0 15074  2279   0   0   0   0    0   0 105   79  23 99  0  0  0
 1  0 15074  2279   0   0   0   0    0   0 105   21  23 99  0  0  0
```

The process is using too much CPU and its priority value has been increased up to the maximum limit of 126. This does not mean that the process is blocked, but all the other processes in the system (except those with fixed priority) have now a priority value smaller or equal to cpu_bound so they are more favorite in using the CPU.

The following memory_bound script calls itself, causing the creation of a huge number of processes:

```
#!/bin/ksh

/tmp/memory_bound
```

Since each process has its own data area, the presence of all these processes causes the allocation of great amounts of real memory. When the script is executed with workload management turned off, all memory is used and paging activity is started to make memory available to the system. You can show this activity with the `vmstat` command:

```
# vmstat 2 6
kthr     memory             page              faults        cpu
----- ----------- ------------------------ ------------ -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa
 0  0 20949   120   0   0   0   0    2   0 120  459  63  3  1 96  1
 1  0 21925   127   0   0  73 552  998   0 167  886  43  4 55 42  0
 2  0 23606   127   0   1 134 948 3731   0 220 1506  59  6 88  6  0
 2  0 24503    97   0   0  94 504 2528   0 161  799  41  3 56 41  0
 1  0 25111   124   0   1 250 356  765   0 235  564  37  3 54 44  0
 1  0 25912   110   0   2 229 440  882   0 224  700  43  3 52 33 12
```

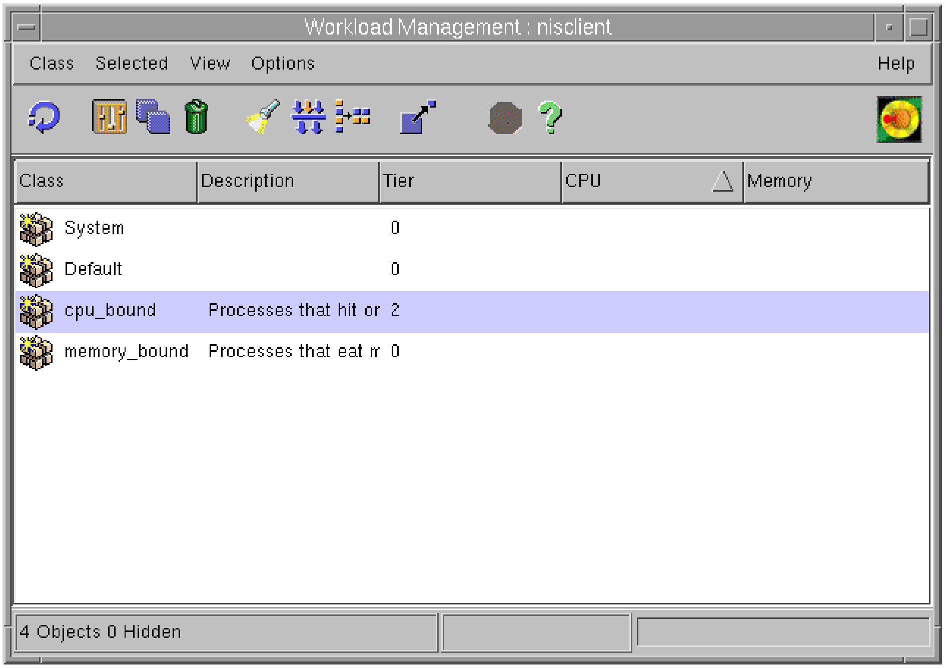
If a new class memory_bound is created with Max memory value of 70% and a rule is added to make the memory_bound script to belong to that class, the activation of workload management changes the system behavior.

When the sum of real memory used by the set of memory_bound processes reaches the 70% of real memory, the VMM no longer allocates new real memory but starts paging, stealing pages from the memory_bound classes pages. The free pages no longer goes near the 127 limit and there is a big paging activity that involves only the memory_bound processes.

```
# vmstat 2 10
kthr     memory             page              faults        cpu
----- ----------- ------------------------ ------------ -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa
 0  0 15682 11594   0   0   0   0  200   0 120  463  63  3  1 95  1
 1  0 17434  9617   0   0   0   0    0   0 111 1607  59  7 33 60  0
 3  0 21627  4815   0   0   0   0    0   0 105 3737 109 16 84  0  0
 3  0 25617   271   0   0  14  77 11947   0 113 3578 104 15 85  0  0
 0  1 26397   309   0   0 414 450  873   0 268  742  77  2 44 36 18
 1  0 27177   299   0   0 391 459  583   0 236  719  43  4 54 42  0
 1  0 27899   376   0   0 405 404  468   0 201  646  39  2 56 42  0
 1  0 28356   376   0   0 230 262  310   0 143  414  30  2 53 44  0
 2  0 29131   376   0   0 395 445  486   0 169  704  37  3 97  0  0
 3  0 29763   377   0   0 321 363  401   0 173  587  39  2 56 42  0
```

7.9 Web-Based System Manager Interface

A `wsm` interface is available to manage the workload management configuration files and to start, stop and monitor the status of classes. The basic panel is shown in Figure 46: it provides the current configuration.



Class	Description	Tier	CPU	Memory
System		0		
Default		0		
cpu_bound	Processes that hit or	2		
memory_bound	Processes that eat m	0		

4 Objects 0 Hidden

Figure 46. Workload Management Web-Based System Manager Main Panel

Like all the other Web-Based System Manager interfaces, you can double click on the existing classes to change their configuration, or you can select on the upper left corner on **Class** to use the main menu, shown in Figure 47.

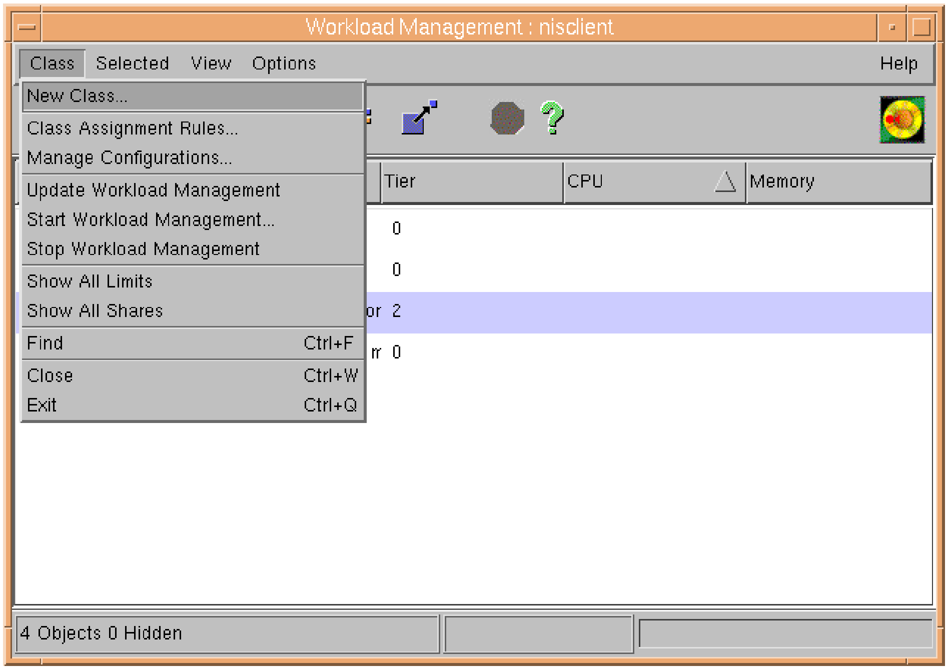
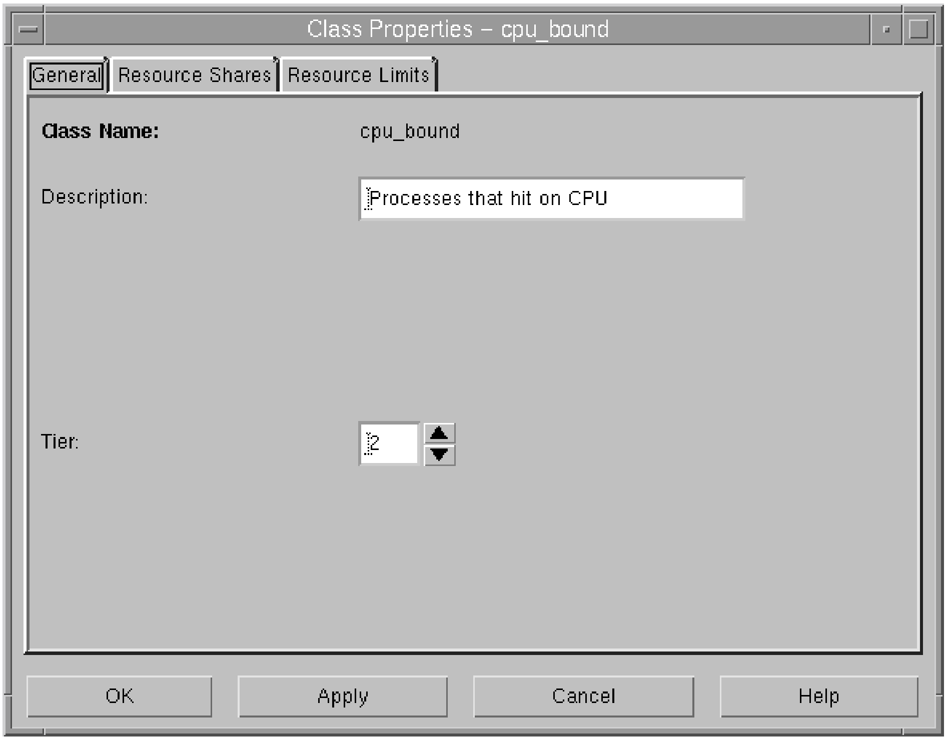


Figure 47. Workload Management Web-Based System Manager Menu

Using the menu you can add, delete and modify classes. A panel is provided where the class name and characteristics can be easily entered, as shown in Figure 48.



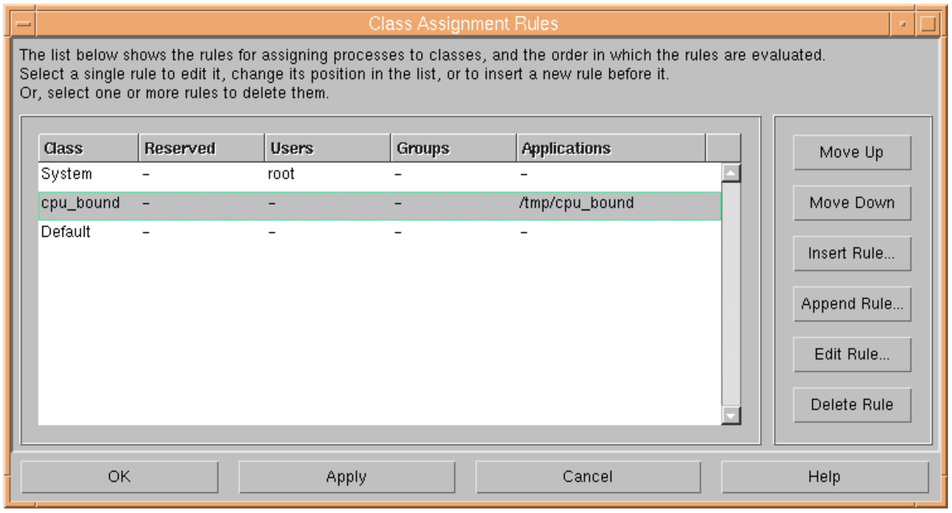
Class Name: cpu_bound

Description: Processes that hit on CPU

Tier: 2

Figure 48. Adding a Class using Web-Based System Manager

In a similar easy way it is possible to define rules and to alter the order in which the rules are applied during process classification. An example is shown in Figure 49.

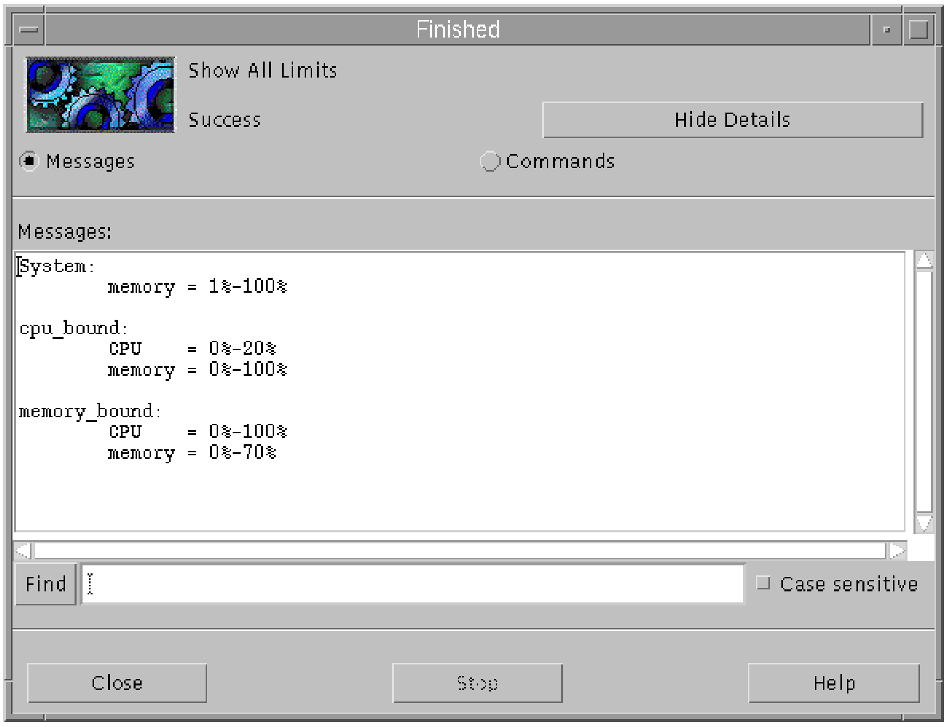


The list below shows the rules for assigning processes to classes, and the order in which the rules are evaluated.
Select a single rule to edit it, change its position in the list, or to insert a new rule before it.
Or, select one or more rules to delete them.

Class	Reserved	Users	Groups	Applications
System	-	root	-	-
cpu_bound	-	-	-	/tmp/cpu_bound
Default	-	-	-	-

Figure 49. Rules Management Using Web-Based System Manager

Other useful features available in Web-Based System Manager interface are the activation, update and deactivation of workload management, as well as the queries on the status of resources. An example is shown in Figure 50, where system limits are displayed.

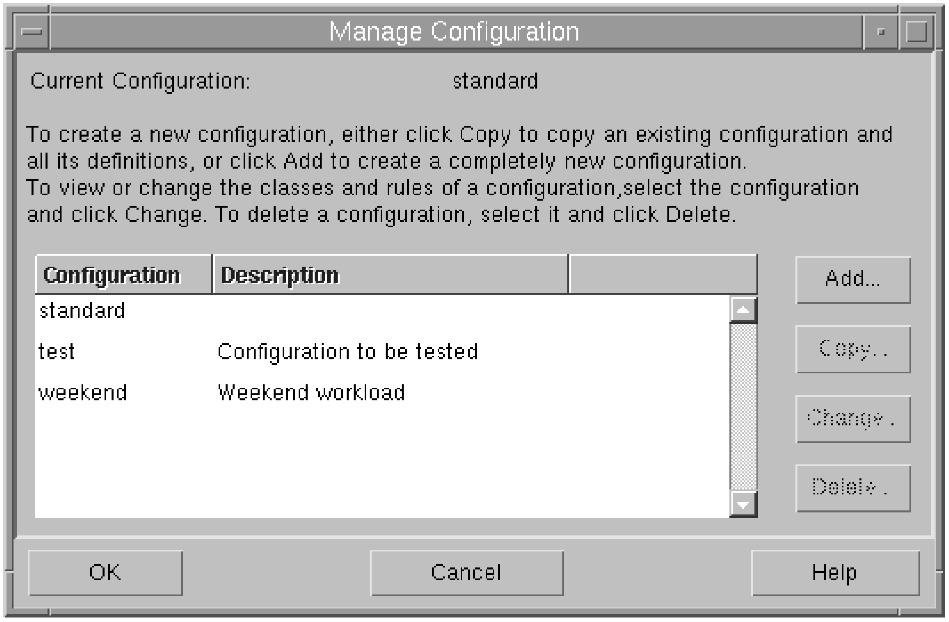


```
System:
        memory = 1%-100%

cpu_bound:
        CPU    = 0%-20%
        memory = 0%-100%

memory_bound:
        CPU    = 0%-100%
        memory = 0%-70%
```

Figure 50. Show Defined Limits with Web-Based System Manager

A useful and unique feature of Web-Based System Manager is configuration management that allows you to have multiple workload management configuration you may decide to activate in different moments. You do not have this option in SMIT, but you can do the same job at command line. Web-Based System Manager maintains different directories under the /etc/wlm directory that contain the configuration files and changes the /etc/wlm/current soft link pointing to the active configuration. The name of the directory represents the name of the configuration.

At the main workload management panel you can click on **Class** and then **Manage Configurations...** to bring up the panel that shows all defined configurations. An example is in Figure 51.


Manage Configuration

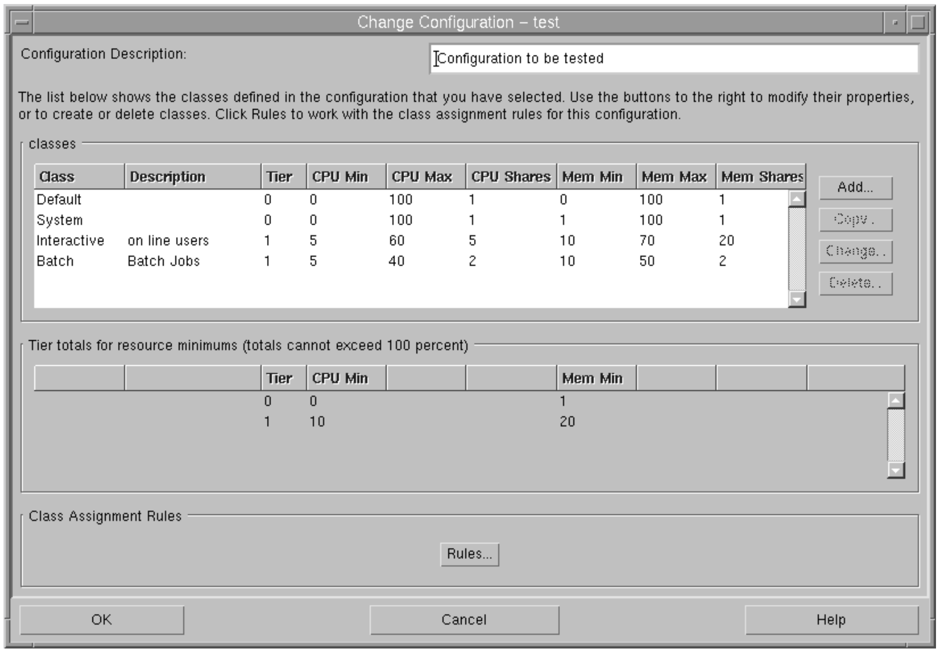
Current Configuration: standard

To create a new configuration, either click Copy to copy an existing configuration and all its definitions, or click Add to create a completely new configuration.
To view or change the classes and rules of a configuration,select the configuration and click Change. To delete a configuration, select it and click Delete.

Configuration	Description
standard	
test	Configuration to be tested
weekend	Weekend workload

Figure 51. Multiple Workload Management Configurations

From the Manage Configuration panel you can alter the structure of any configuration, even if not active. The panel that is provided when you click on the **Change...** button is even more effective of the other Web-Based System Manager panels, as you can see in Figure 52.



Configuration Description: Configuration to be tested

The list below shows the classes defined in the configuration that you have selected. Use the buttons to the right to modify their properties, or to create or delete classes. Click Rules to work with the class assignment rules for this configuration.

classes

Class	Description	Tier	CPU Min	CPU Max	CPU Shares	Mem Min	Mem Max	Mem Shares
Default		0	0	100	1	0	100	1
System		0	0	100	1	1	100	1
Interactive	on line users	1	5	60	5	10	70	20
Batch	Batch Jobs	1	5	40	2	10	50	2

Tier totals for resource minimums (totals cannot exceed 100 percent)

Tier	CPU Min	Mem Min
0	0	1
1	10	20

Class Assignment Rules

Figure 52. Change Configuration Panel

You have a full description of all classes with tiers, Min, Max and shares values as well as the sum of all minimum values provided to each tier, that must never exceed the 100% value.

Chapter 8. Networking Enhancements

Internet Protocol Version 6 (IPv6) was first introduced in AIX version 4.3. In AIX version 4.3.2, IPV6 routing is supported. Some important network improvements for Web servers are also supported in AIX version 4.3.2.

In these sections, the networking enhancements in AIX version 4.3, 4.3.1, 4.3.2 and 4.3.3 are discussed.

8.1 Internet Protocol Version 6

Internet Protocol Version 6 (IPv6) is the next generation Internet Engineering Task Force (IETF) networking protocol that will become the industry standard network protocol for the internet of the future.

IPv6 extends the maximum number of Internet addresses to handle the ever-increasing Internet user population. IPv6 is an evolutionary change from IPv4 that has the advantage of allowing a mixture of the new and the old to coexist on the same network. This coexistence enables an orderly migration from IPv4 (32-bit addressing) to IPv6 (128-bit addressing) on an operational network.

8.1.1 IPv6 Introduction

This initial release of IPv6 in AIX 4.3.0 is a migration platform to enable user migration to IPv6. The AIX IPv6 migration platform in AIX 4.3.0 supports IPv6 host function only. This means that no gateway support is included (it is available in later updates), so IPv6 packets cannot be forwarded from one interface to another on the same RS/6000.

The following function is included in this update:

- IPv6 128-bit addressing
- Neighbor Discovery/Stateless Address Auto configuration
- Internet Control Message Protocol (ICMPv6)
- Tunneling over IPv4
- IP Security (IPSec)
- Resolver support for /etc/hosts
- Commands/applications enabled for IPv6
- IPv6 Socket Library Support
- System management changes

The following network media is supported in the AIX IPv6 migration platform:

- Ethernet

- Token-Ring
- FDDI

8.1.2 IPv6 128-Bit Addressing

A brief introduction to IPv6 addressing is presented here. For more detail, refer to the appropriate RFC documents. See section 8.1.11, “IPv6 and IPSec-Related RFCs Implementation” on page 397 for a listing of RFCs supported by the AIX IPv6 migration platform in AIX 4.3.0.

8.1.2.1 Text Representation of Addresses

As shown in the following example, an IPv6 address is represented by hexadecimal digits separated by colons, where IPv4 addresses are represented by decimal digits separated by dots or full-stops. IPv6 is, therefore, also known as colon-hex addressing, compared to IPv4's dotted-decimal notation.

IPv6 addresses are 128-bit identifiers for interfaces and sets of interfaces. Note that IPv6 refers to *interfaces* and not to *hosts*, common to IPv4.

There are three conventional forms for representing IPv6 addresses as text strings:

1. The preferred form is `x:x:x:x:x:x:x:x`, where the `x`s are the hexadecimal values of the eight 16-bit pieces of the address, each separated by a colon.

   Examples are:

   ```
   FEDC:BA98:7654:3210:FEDC:BA98:7654:3210
   1080:0:0:0:8:800:200C:417A
   ```

   Note that it is not necessary to write the leading zeros in an individual field, but there must be at least one numeral in every field (except for the case described in 2).

2. Due to the method used for allocating certain styles of IPv6 addresses, it will be common for addresses to contain long strings of zero bits. To make writing addresses containing zero bits easier, a special syntax is available to compress the zeros. The use of `::` (two colons) indicates multiple groups of 16-bits of zeros. Note that the `::` can only appear once in an address. The `::` can also be used to compress the leading or trailing zeros in an address.

   For example, the following addresses:

```
1080:0:0:0:8:800:200C:417A  a unicast address
FF01:0:0:0:0:0:0:43         a multicast address
0:0:0:0:0:0:0:1             the loopback address
0:0:0:0:0:0:0:0             the unspecified addresses
```

may be represented as:

```
1080::8:800:200C:417A       a unicast address
FF01::43                    a multicast address
::1                         the loopback address
::                          the unspecified addresses
```

3. An alternative form that is sometimes more convenient when dealing with a mixed environment of IPv4 and IPv6 nodes is `x:x:x:x:x:x:d.d.d.d`, where `x` is the hexadecimal values of the six high-order 16-bit pieces of the address, and `d` is the decimal values of the four low-order 8-bit pieces of the address (standard IPv4 representation).

   Examples:

```
0:0:0:0:0:0:13.1.68.3
0:0:0:0:0:FFFF:129.144.52.38
```

   or in compressed form:

```
::13.1.68.3
::FFFF:129.144.52.38
```

   Note: `FFFF` is used to represent addresses of IPv4-only nodes (those that do not support IPv6).

8.1.2.2 Types of IPv6 Address

In IPv6, there are three types of addresses:

Unicast

An identifier for a single interface. A packet sent to a unicast address is delivered to the interface identified by that address. A unicast address has a particular scope as shown in the following lists:

- Link-local
  - Valid only on the local link (that is, only one hop away).
  - Prefix is `fe80::/16`.

- Site-local
  - Valid only at the local site (for example, inside IBM Austin).
  - Prefix is `fec0::/16`.
- Global
  - Valid anywhere in the Internet.
  - Prefix may be allocated from unassigned unicast space.

There are also two special unicast addresses:

- `::/128` (unspecified address).
- `::1/128` (loopback address - note that in IPv6 this is only one address not an entire network).

Multicast

An identifier for a set of interfaces (typically belonging to different nodes). A packet sent to a multicast address is delivered to all interfaces identified by that address. A multicast address is identified by the prefix `ff::/8`. As with unicast addresses, multicast addresses have a similar scope. This is shown in the following lists:

- Node-local
  - Valid only on the source node (for example, multiple processes listening on a port).
  - Prefix is `ff01::/16` or `ff11::/16`.
- Link-local
  - Valid only on hosts sharing a link with the source node (for example, Neighbor Discovery Protocol data).
  - Prefix is `ff02::/16` or `ff12::/16`.
- Site-local
  - Valid only on hosts sharing a site with the source node (for example, multicasts within IBM Austin).
  - Prefix is `ff05::/16` or `ff15::/16`.
- Organization-local.
  - Valid only on hosts sharing organization with the source node (for example, multicasts to all of IBM).
  - Prefix is `ff08::/16` or `ff18::/16`.
- The `0` or `1` part in these prefixes indicates whether the address is permanently assigned (`1`) or temporarily assigned (`0`).

Anycast

An identifier for a set of interfaces (typically belonging to different nodes). An anycast address is an address that has a single sender, multiple listeners, and only one responder (normally the nearest one, according to the routing protocols' measure of distance). An example may be several Web servers listening on an anycast address. When a request is sent to the anycast address, only one responds.

Anycast addresses are indistinguishable from unicast addresses. A unicast address becomes an anycast address when more than one interface is configured with that address.

Note: There are no broadcast addresses in IPv6, their function being superseded by multicast addresses.

8.1.3 Neighbor Discovery/Stateless Address Autoconfiguration

Neighbor Discovery (ND) protocol for IPv6 is used by nodes (hosts and routers) to determine the link-layer addresses for neighbors known to reside on attached links and maintain per-destination routing tables for active connections. Hosts also use Neighbor Discovery to find neighboring routers that forward packets on their behalf and detect changed link-layer addresses. Neighbor Discovery protocol (NDP) uses the ICMPv6 protocol with a unique message types to achieve the above function. In general terms, the IPv6 Neighbor Discovery protocol corresponds to a combination of the IPv4 protocols Address Resolution Protocol (ARP), ICMP Router Discovery (RDISC), and ICMP Redirect (ICMPv4), but with many improvements over these IPv4 protocols.

IPv6 defines both a stateful and stateless address autoconfiguration mechanism. Stateless autoconfiguration requires no manual configuration of hosts, minimal (if any) configuration of routers, and no additional servers. The stateless mechanism allows a host to generate its own addresses using a combination of locally available information and information advertised by routers. Routers advertise prefixes that identify the subnet(s) associated with a link, while hosts generate an interface-token that uniquely identifies an interface on a subnet. An address is formed by combining the two. In the absence of routers, a host can only generate link-local addresses. However, link-local addresses are sufficient for allowing communication among nodes attached to the same link.

8.1.3.1 NDP Application Kernel Support

For kernel function, a new version of the netinet kernel extension has been provided that contains code to handle both IPv4 and IPv6. New special processing handled by the kernel (or NDP applications) includes:

- Interface initialization
  - Autoconfiguration

    - Duplicate address detection (DAD)
    - Joining the all nodes multicast group
    - Sending router solicitations
    - Responding to router advertisements
- Interface maintenance
    - Managing prefixes (changing, if necessary, when router advertisements are received)
    - Timing out prefixes when advertised lifetime expires
    - Forming an appropriate address for received prefixes
    - DAD as necessary
- NDP support
    - Destination routing (NDP table)
    - Path MTU
    - Determination of neighbor unreachability (NUD)
- Source address selection
    - For multihomed hosts (problems with link-local)
- Routing
    - Uses version 4 routes for compatible addresses (through the Simple Internet Transition [SIT] interface)
    - Maintains multiple default routes, with fast switching based on NUD
- Tunneling
    - See section 8.1.5, “Tunneling over IPv4” on page 372 for more details.
- ICMPv6 - control messages
    - See section 8.1.4, “Internet Control Message Protocol (ICMPv6)” on page 370 for more details.
- Address mapping
    - Incoming v4 packets passed up to v6-aware applications through mapping the address to v6

Note: For more detailed information on this topic, refer to RFCs 1970 and 1971.

8.1.4 Internet Control Message Protocol (ICMPv6)

ICMPv6 is used by IPv6 nodes to report errors encountered in processing packets and to perform other internet-layer functions, such as diagnostics (ICMPv6 ping) and multicast membership reporting.

8.1.4.1 ICMPv6 Message Types

ICMPv6 messages are grouped into two classes:

- Error messages
- Informational messages

Error messages are identified by having a zero in the high-order bit of their message Type field values. Thus, error messages have message types from 0 to 127. Table 32 lists the different types and their meanings.

Table 32. ICMPv6 Error Messages

Type	Description	Code	Cause
1	Destination unreachable	0	No route to destination
		1	Communication with destination administratively prohibited
		2	Not a neighbor
		3	Address unreachable
		4	Port unreachable
2	Packet too big	0	Packet too big
3	Time exceeded	0	Hop limit exceeded in transit
		1	Fragment reassembly time exceeded
4	Parameter problem	0	Erroneous header field encountered
		1	Unrecognized Next Header type encountered
		2	Unrecognized IPv6 option encountered

Informational messages have message type values from 128 to 255. Table 32 lists the information types and their meanings.

Table 33. ICMPv6 Informational Messages

Type	Description
128	Echo request
129	Echo reply
130	Group membership query
131	Group membership report
132	Group membership reduction

Note: For more detailed information relating to message types and formats, refer to RFC 1885.

8.1.5 Tunneling over IPv4

The key to a successful IPv6 transition is compatibility with the existing installed base of IPv4 hosts and routers. Maintaining compatibility with IPv4, while deploying IPv6, streamlines the task of transitioning the Internet to IPv6.

In most deployment scenarios, the IPv6 routing infrastructure will be built-up over time. While the IPv6 infrastructure is being deployed, the existing IPv4 routing infrastructure can remain functional and will be used to carry IPv6 traffic. Tunneling provides a way to use an existing IPv4 routing infrastructure to carry IPv6 traffic.

IPv6/IPv4 hosts and routers can tunnel IPv6 datagrams over regions of IPv4 routing topology by encapsulating them within IPv4 packets. Tunneling can be used in a variety of ways:

Router-to-Router
IPv6/IPv4 routers interconnected by an IPv4 infrastructure can tunnel IPv6 packets between themselves. In this case, the tunnel spans one segment of the end-to-end path that the IPv6 packet takes.

Host-to-Router
IPv6/IPv4 hosts can tunnel IPv6 packets to an intermediary IPv6/IPv4 router that is reachable through an IPv4 infrastructure. This type of tunnel spans the first segment of the packet's end-to-end path.

Host-to-Host
IPv6/IPv4 hosts that are interconnected by an IPv4 infrastructure can tunnel IPv6 packets between themselves. In this case, the tunnel spans the entire end-to-end path that the packet takes.

Router-to-Host
IPv6/IPv4 routers can tunnel IPv6 packets to their final destination IPv6/IPv4 host. This tunnel spans only the last segment of the end-to-end path.

Tunneling techniques are usually classified according to the mechanism by which the encapsulating node determines the address of the node at the end of the tunnel. In the first two tunneling methods listed above, router-to-router and host-to-router, the IPv6 packet is being tunneled to a router. In the last two tunneling methods, host-to-host and router-to-host, the IPv6 packet is tunneled all the way to its final destination.

8.1.5.1 Tunneling Mechanisms

The following is the path taken when tunneling:

- The entry node of the tunnel (the encapsulating node) creates an encapsulating IPv4 header and transmits the encapsulated packet.
- The exit node of the tunnel (the decapsulating node) receives the encapsulated packet, removes the IPv4 header, updates the IPv6 header, and processes the received IPv6 packet.
- The encapsulating node needs to maintain soft state information for each tunnel, such as the MTU of the tunnel, to process IPv6 packets forwarded into the tunnel.

Note: The information presented above is a brief summary of tunneling from RFC 1933. Refer to the RFC for more detailed information on this topic.

8.1.6 IP Security

In this section, we provide a brief IP Security (IPSec) description and a simple configuration example.

8.1.6.1 What is IP Security?

In the simplest meaning, IPSec provides ways to authenticate peers and encrypt IP packets along with filtering capabilities. IP protocol did not have any security mechanisms before IPSec was introduced. Typically security mechanisms were provided through use of other transport/application level protocols such as SSL and Kerberos, or use of firewalls for packet filtering. By adding robust security mechanisms on IP protocol layer, most applications can benefit from it without any costs because IP is the most widely used and only network layer protocol in TCP/IP world.

Basic security services provided by IPSec are similar to those that are provided by other security protocols. They are:

- Authentication

  IPSec provides a way to authenticate peers. This means that the data originator is really the one the receiver believes communicating with.
- Data integrity

  By using hash functions, IPSec assures that the data in a packet has not been modified.
- Encryption

By encrypting data in a packet, the message in a packet is protected from malicious eavesdroppers.

- Robustness against replay attacks

  IPSec implements robustness against replay attacks by having sequence number filed in its headers.

- Access control

  IPSec provides a packet filter mechanism based on various criteria.

Also, IPSec can be used as a building block for virtual private network (VPN). By definition, a VPN is an extension of an enterprise's private intranet across a cost-effective public network. By using IPSec, the enterprise can connect two remote locations through Internet as though they were connected through a secure leased line. Figure 53 shows a simple VPN through an IPSec tunnel.

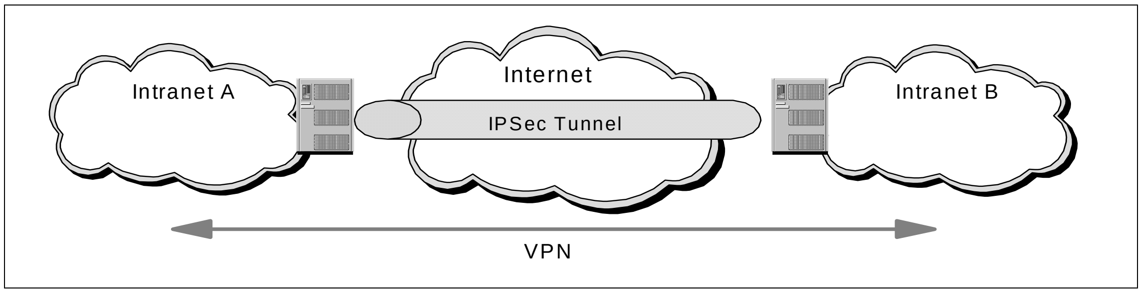


Figure 53. Virtual Private Network

The IP packets that flow through the IPSec tunnel are authenticated and typically encrypted so that any two entities in intranet A and intranet B can communicate securely.

Another important feature of IPSec is that it is independent from any algorithms although it defines a standard set of algorithms initially. IPSec is designed to work with different kinds of encryption and authentication algorithms. By allowing the selection and the replacement of the algorithms, it is possible to change the authentication and encryption methods in the future when better and stronger methods are developed.

8.1.6.2 Basic Concepts of IP Security

In this section, basic concepts of IPSec are described.

Security Association

A Security Association (SA) is a security agreement between two entities (hosts, routers). It is fundamental to IPSec. An SA is identified by security parameter index (SPI), destination IP address and security protocol. SPIs are carried over inside IPSec headers so that each entity can identify characteristics of the SA from security association database (SAD). Followings are important features of SA.

- SA is simplex

  An SA defines security characteristics only for a unidirectional flow. When two entities communicate using IPSec and if only one SA exists, security operations are applied to either of flows. If bidirectional security operations are necessary, at least two SAs must be defined.

- SAs must be predefined

  SAs must be predefined before any actual data flow takes place. An SA can be defined either by manual configuration or by dynamic mechanisms. IPSec standards defines Internet key exchange (IKE) mechanism that can also be used for dynamically defining SAs.

- Security protocols

  Authentication header (AH) and Encapsulating Security Payload (ESP) are currently defined as security protocols for IPSec. When defining the security protocol for an SA, underlying security algorithms are also defined.

- Security Parameter Index

  An SPI is 32-bit value and is used to identify an SA from other SAs of same destination IP address and protocol.

- Security Association Database

  A security association database (SAD) contains parameter information about each SA such as security protocol (AH or ESP), algorithms the protocol uses , keys, sequence number verification capability, protocol mode (tunnel or transport) and SA lifetime.

- Security Policy Database

  A security policy database (SPD) specifies what security services are to be offered to the IP traffic, depending on factors such as source, destination, whether it is inbound, outbound, or the like.

On AIX 4.3, SAD and SPD roughly correspond to tunnel database and filter rules.

Authentication Header

Authentication header (AH) is one of two security protocol defined for IPSec. The purpose of AH is to provide authentication and integrity to IP packets.

Optionally, it can provide anti-replay attack capability. AH does not have any encryption capabilities.

AH authenticates and assures the integrity of whole IP packet including IP header except some mutable fields (in transport mode). The mutable fields contain a value such as TTL that changes when the packet flows through networks.

Because AH only has authentication and integrity check capabilities, it is often used when no encryption requirements are necessary or where encryption is prohibited.

AH is defined as an optional headers for both IP v4 and v6. It has following header format:

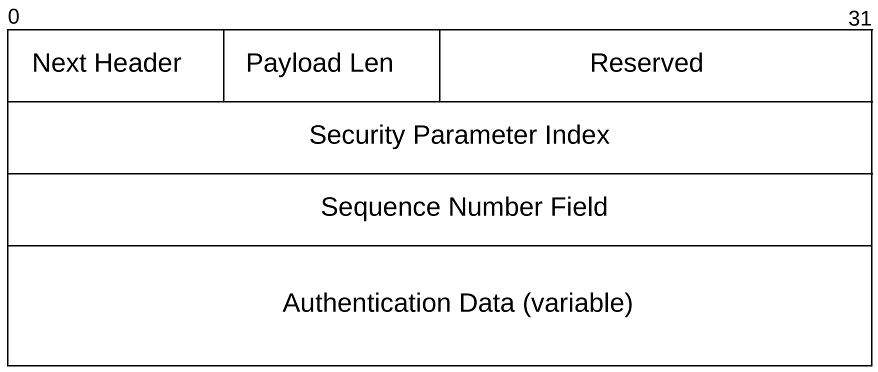


Figure 54. Authentication Header

- Sequence Number Field

  This field is used by receiver for anti-replay attack purpose. The use of this filed is enabled when defining the SA.
- Authentication Data

  This field contains integrity check value (ICV). The ICV is calculated by authentication algorithm defined in the SA when the packet is sent. Receivers can check the integrity of the packet by matching the ICV with the output value of its own calculation.

AH is located soon after IP header in IP v4 and appropriate position defined in RFC 2460 in IP v6.

The following two algorithms are mandatory to AH:

- HMAC with MD5

- HMAC with SHA-1

When an IP packet with AH received (inbound), the packet is processed as follows:

1. The receiver assembles packets if necessary.
2. The receiver looks for the appropriate SA consulting SAD and SPD using destination IP address, security protocol (AH) and SPI. The SA may contain sequence number verification capability, authentication algorithm and one or more keys required for the algorithm.
3. Sequence number field is verified if necessary. If the packet fails the verification, it is discarded.
4. The ICV in the authentication data field is verified using the authentication algorithm. If the computed value matches ICV, the packet is authenticated and passed to higher layer.

At this point, the integrity of the packet is assured because the ICV matches, and the packet is authenticated because the keys (typically shared keys) used for computing the ICV are proved to be valid.

When sending an IP packet with AH (outbound), the sender processes the packet as follows:

1. Looks up SAD and SPD to check if AH processing is necessary.
2. Increments sequence number filed.
3. Calculates ICV. The sender calculates ICV without mutable fields. The value of these fields are set to zero when calculating. Padding to authentication data filed is done if necessary when storing ICV.
4. Fragments the packet if necessary.

Encapsulating Security Payload

Encapsulating Security Payload (ESP) is another security protocol defined for IPSec. ESP provides authentication, integrity, encryption and anti-replay attack capability to IP packets.

Unlike AH, ESP does not authenticate IP header and the encryption does not cover IP header and ESP header. In tunnel mode, the IP header is also authenticated and encrypted because the IP header is inside the payload for the new encapsulating IP packet (see 8.1.6.3 on page 380).

Figure 55 illustrates ESP packet format. Because ESP consists of ESP header, ESP trailer and ESP auth data, a whole IPv4 packet is shown for explanation.

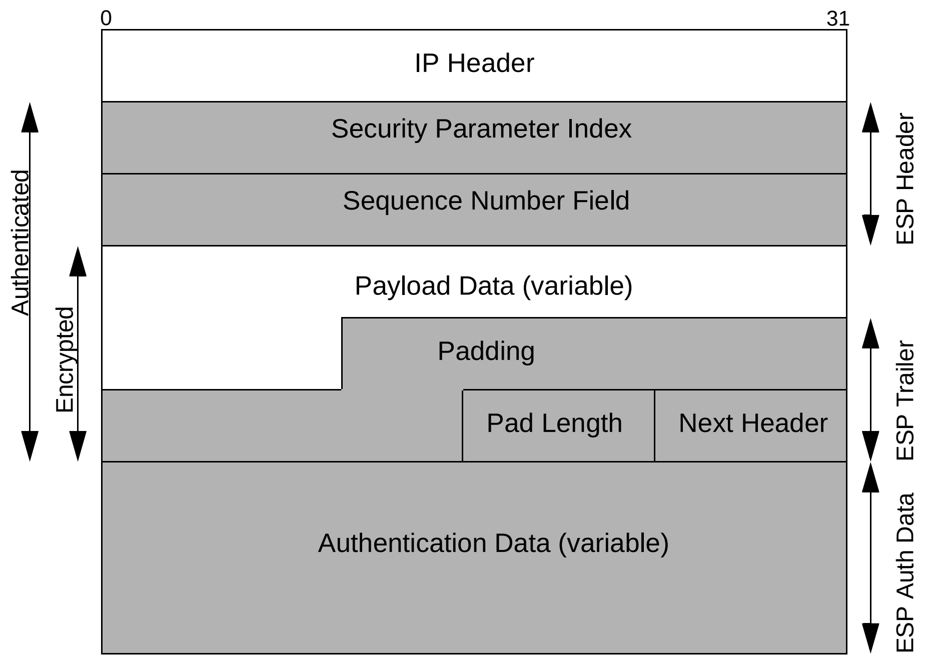


Figure 55. Encapsulating Security Payload

- ESP Header

  ESP Header contains SPI and sequence number. The purpose of these fields is same as that of AH.

- Payload Data

  This field contains headers for upper layer protocol such as TCP and application data.

- ESP Trailer

  Because most encryption algorithms require that the input data must be an integral number of blocks, padding is often necessary to set the length of encrypted data on a certain boundary.

- ESP Auth Data

  This field contains the ICV calculated by the authentication algorithm designated when defining the SA.

The following encryption and authentication algorithms are mandatory to be implemented to ESP:

- DES in CRC mode

- HMAC with MD5
- HMAC with SHA-1
- NULL Authentication algorithm
- NULL Encryption algorithm

ESP is designed for use with symmetric encryption algorithms (shared key encryptions).

When an IP packet with ESP received (inbound), the packet is processed as follows:

1. The receiver assembles packets if necessary.
2. The receiver looks up SAD and SPD for appropriate SA for the packet.
3. Optionally, the sequence number filed is verified for anti-replay attack purpose.
4. The ICV in authentication field is checked if the value matches the computed value on receiver’s side. If the two values match, the packet is authenticated.
5. The payload data, padding, padding length and next header fields are decrypted using keys, encryption algorithm and other necessary data from the SA.

When sending and IP packet with ESP (outbound), the sender processes the packet as follows:

1. The sender looks up SAD and SPD for appropriate SA for the packet.
2. The sender encrypts the payload data, padding, padding length and next header filed.
3. Sequence number is generated for anti-replay attack purpose.
4. The ICV is calculated and stored in authentication data filed.
5. The packet is fragmented if necessary.

Note

AH and ESP can be used at the same time.

Key Management

Both AH and ESP use keys to authenticate or encrypt/decrypt the contents of IP packets. The keys used are typically symmetric (shared) keys to avoid high CPU usage of public key calculation. No matter if the keys are symmetric or asymmetric, they must be changed periodically to keep the robustness. The keys can be changed manually in agreement with each entity. But when the number of hosts and routers participating increases, automatic key exchange mechanism is essential. Also automatic negotiation of SA is necessary to cope with the increased entities.

RFC 2409 *The Internet Key Exchange (IKE)* defines ISAKMP/Oakley as a key exchange protocol. Internet security association and key management protocol (ISAKMP) is a framework for key exchange defined in RFC 2408. Oakley is a key determination protocol defined in RFC2412. To avoid a chicken and egg type problem, ISAKMP/Oakley protocol is typically implemented as an independent subsystem from IPSec and communicates through UDP port 500 without using IPSec.

ISAKMP/Oakley

By setting up ISAKMP/Oakley, administrators can benefit from its automated generation and refresh of cryptographic keys function as well as automated negotiation of SA. The defined SA can be inserted into SAD and referred afterwards when actual IPSec data exchange takes place.

ISAKMP/Oakley protocol uses two-phase approach to establish SAs and keys between two entities.

- Phase 1

  In phase 1, ISAKMP establishes an SA for itself. By establishing the SA, the subsequent message flows between two ISAKMP entities are protected. It generally uses public key cryptography to define shared keys that will be used in phase 2 message exchanges. Once phase 1 has been activated, multiple phase 2 message exchanges can be done. Because phase 2 uses shared key, the computational overhead for encrypting and decrypting messages are low.

- Phase 2

  In phase 2, ISAKMP establishes SAs and defines keys that will be used to protect IP packets exchanged between two IPSec users. Phase 2 negotiation can also be used to refresh the cryptographic keys for specific SAs only. The actual negotiations are done by sending proposal payloads that contain preferred security protocols (AH, ESP) or cryptography algorithms.

8.1.6.3 Transport Mode SA vs Tunnel Mode SA

An SA has two modes, transport mode and tunnel mode. The mode of an SA is determined when defining the SA. A transport mode SA is an SA between two hosts that do not forward IPSec packets to other hosts or routers. On the other hand, a tunnel mode SA is established between a host and a router, or a router and a router. Also, the tunnel mode SA can be established between hosts. The tunneling is a common concept for encapsulating a packet inside another packet of same network protocol layer. The IPSec tunneling is actually modeled after RFC 2003 *IP Encapsulation within IP*. The difference between the ordinary IP encapsulation and IPSec tunneling is that in IPSec tunneling, the packet inside is authenticated or encrypted.

When transport SAs are established between two entities, only packets that are exchanged between them are treated securely. The relayed packets are not authenticated or encrypted when they act as routers. To encrypt/authenticate the relayed packets, tunnel mode SAs and appropriate filters must be defined.

The primary difference between transport mode SA and tunnel mode SA is the header location of each security protocols (AH and ESP).

The following figures illustrate the location of security headers of each protocol in each mode (IPv4 only).

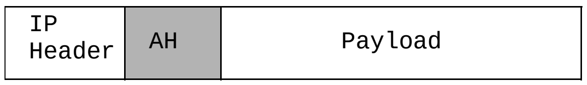


Figure 56. AH in Transport Mode

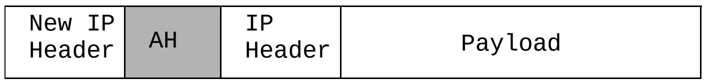


Figure 57. AH in Tunnel Mode

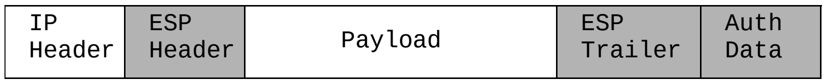


Figure 58. ESP in Transport Mode

New IP Header	ESP Header	IP Header	Payload	ESP Trailer	Auth Data

Figure 59. ESP in Tunnel Mode

In Figure 57 and Figure 59, the old IP packets including old IP header are encapsulated within new IP packets. This represents two important features for implementing a VPN. First, the IP address in old IP header is hidden when the packet traverses Internet so that two entities in different intranets can communicate each other using their private IP addresses. Second, all contents of old IP packets can be authenticated or encrypted or both so that they can be protected from security attacks or exposures.

Note that the meaning of the word *tunnel* used when implementing IPSec features on AIX 4.3 is different from the meaning used in this section. When you configure IPSec on AIX, the word tunnel means roughly SA. For example, you create a tunnel even when you define an transport mode SA. Also, a tunnel may produce a set of SAs related to the tunnel.

8.1.6.4 IP Security Configuration Example

In this section, a simple VPN configuration example is shown. The configuration assumes a network environment in Figure 60.

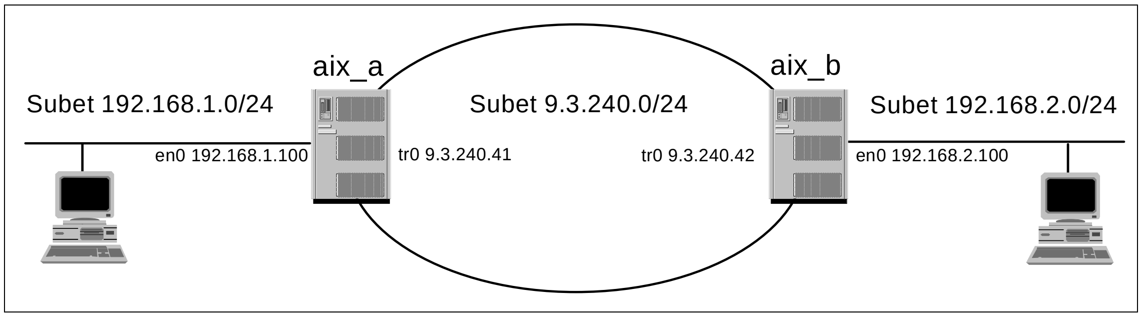


Figure 60. VPN Configuration Example

The configuration scenario is as follows:

- Host aix_a and aix_b establish an IPSec tunnel (tunnel mode SA).
- The tunnel is established as a manual tunnel.
- All the traffics between subnet 192.168.1.0 and 192.168.2.0 are encrypted and authenticated using both ESP and AH.

Establishing a Tunnel Between aix_a and aix_b

Use following command to create a tunnel on aix_a:

```
# /usr/sbin/gentun -v 4 -t manual -p ae -P ae -s ’9.3.240.41’ \
-d ’9.3.240.42’ -m ’TUNNEL’ -p ’ea’ -a ’HMAC_MD5’ -e ’DES_CBC_4’ \
-N ’300’ -l ’0’
Tunnel 1 for IPv4 has been added successfully.
Filter rule 6 automatically generated for this tunnel.
Filter rule 7 automatically generated for this tunnel.
```

At this point, a tunnel definition and default filter definitions are generated on aix_a. The keys for the ESP and AH security protocols used by their underlying algorithms can be specified to `gentun` command. In this example, they are generated automatically.

User following command to export the tunnel definition to a file:

```
# exptun -v 4 -f /tmp
No IBM tunnel found for IPv4
Manual tunnel(s) has been exported to /tmp/ipsec_tun_manu.exp successfully.
```

/tmp/ipsec_tun_manu.exp is generated. Move the file to aix_b’s /tmp directory and issue following command to import the tunnel definition:

```
# imptun -v 4 -f /tmp
IBM tunnels not found.
IPv4 tunnel 1 imported as 1.
Filter rule 5 automatically generated for tunnel 1.
Filter rule 6 automatically generated for tunnel 1.
```

Start IPSec on both aix_a and aix_b if not started yet.

```
# mkdev -l ipsec_v4
```

Activate the tunnel on both aix_a and aix_b:

```
# mktun -v 4 -t 1
Filter rules for IPv4 has been updated.
Tunnel 1 for IPv4 activated.
```

At this point, only packets that have 9.3.240.41 and 9.3.240.42 as their source and destination address are authenticated and encrypted. The `ipreport` command output shows that AH and ESP headers are included in the packets.

```
====( 130 bytes transmitted on interface tr0 )==== 14:20:01.064098532
802.5 packet

802.5 MAC header:
access control field = 0, frame control field = 40
```

```
[ src = 00:04:ac:61:74:00, dst = 00:20:35:7c:5f:d8]
802.2 LLC header:
dsap aa, ssap aa, ctrl 3, proto 0:0:0, type 800 (IP)
IP header breakdown:
        < SRC =      9.3.240.41 >  (aix_a.itsc.austin.ibm.com)
        < DST =      9.3.240.42 >  (aix_b.itsc.austin.ibm.com)
        ip_v=4, ip_hl=20, ip_tos=0, ip_len=108, ip_id=35259, ip_off=0
        ip_ttl=60, ip_sum=24a, ip_p = 51 (AH)
Authentication header breakdown:
        spi=300, payload length=16 bytes, seq_num=0x1
        next header=50 (ESP)
Encapsulating Security Payload breakdown:
        spi=300, seq_num=0xfabe0647
```

Setting up Filters for Subnets

For two entities in subnets 192.168.1.0 and 192.168.2.0 to communicate each other using the IPSec tunnel defined in the last step, filter definitions that match the subnets and the tunnel definition must be created.

Create the filter definition on aix_a using following command. Use -s flag to specify the source subnet address, -d for the destination subnet address and -t for the tunnel ID:

```
# /usr/sbin/genfilt -v 4  -a ’P’ -s ’192.168.1.0’ -m ’255.255.255.0’ \
-d ’192.168.2.0’ -M ’255.255.255.0’ -g ’y’ -c ’all’ -o ’any’ -p ’0’ \
-O ’any’ -P ’0’ -r ’B’ -w ’B’ -l ’N’ -f ’y’ -t ’1’ -i ’tr0’
```

Create the filter definition on aix_b using following command:

```
# /usr/sbin/genfilt -v 4  -a ’P’ -s ’192.168.2.0’ -m ’255.255.255.0’ \
-d ’192.168.1.0’ -M ’255.255.255.0’ -g ’y’ -c ’all’ -o ’any’ -p ’0’ \
-O ’any’ -P ’0’ -r ’B’ -w ’B’ -l ’N’ -f ’y’ -t ’1’ -i ’tr0’
```

Activate the filters on both aix_a and aix_b using following command:

```
# /usr/sbin/mkfilt -v 4 -u
```

Then all the traffics between subnet 192.168.1.0 and 192.168.2.0 are authenticated and encrypted.

For ISAKMP configuration examples, see chapter 3 of redbook *A Comprehensive Guide to Virtual Private Networks, Volume III: Cross-Platform Key and Policy Management*, SG24-5309.

8.1.7 Resolver Support for /etc/hosts

IPv6 hosts entries in the /etc/hosts database follow the same standards used for IPv4-only. The hosts file may contain IPv4 and IPv6 addresses as shown in the following example:

An example of the entries in a typical /etc/hosts file:

```
127.0.0.1                loopback   localhost
129.144.248.127          percy
129.144.1.45             jumbuck
::8190:3584              redback
129.144.53.132           redback
::129.144.248.127        percy
fec0:1::11               ethspook
::ffff:129.144.248.127  taipan
fec0:1::                 ethpercy
```

8.1.8 Commands and Applications Enabled for IPv6

The AIX IPv6 migration platform in AIX 4.3.0 has enabled a significant subset of TCP applications to support the IPv6 protocol. It is, however, important to note here that applications not supporting IPv6 in this first release will still function normally for IPv4 networks. Lack of IPv6 support simply means that the application has not been made IPv6 aware.

Some of the major applications that have not been made IPv6 aware are:

- DCE/DFS
- NFS/NIS
- HACMP
- AnyNet

The applications shown in Table 34 are able to run on either the IPv4 or IPv6 stacks. The code automatically determines the appropriate IP version to use.

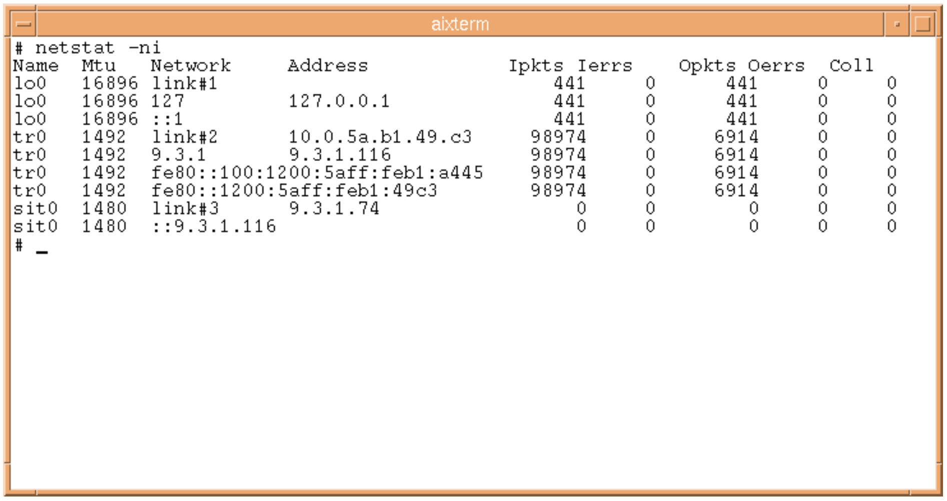
Table 34. Applications Ported to IPv6

Command	Description
autoconf6	Automatically configures IPv6 addresses
crash/ndb	Network debugger updated for IPv6
ftp/ftpd	File transfer
ifconfig	Network interface configuration
inetd	Inetd server

Command	Description
iptrace/ipreport	Trace and dump network packets including IPv6
ndb	Display/update ND cache
ndpd-host	Support for receiving Router Advertisements
netstat	Network statistics
ping	ICMP echo/reply diagnostic
rcp	Remote copy
rexec/rexecd	Remote command execution
rlogin/rlogind	Remote login
route	Add/delete IPv4 and IPv6 routes
rsh/rshd	Remote shell access
tcpdump	Dumps Network packets, including IPv6
telnet/telnetd	Remote login
tftp/tftpd	File transfer

8.1.8.1 netstat

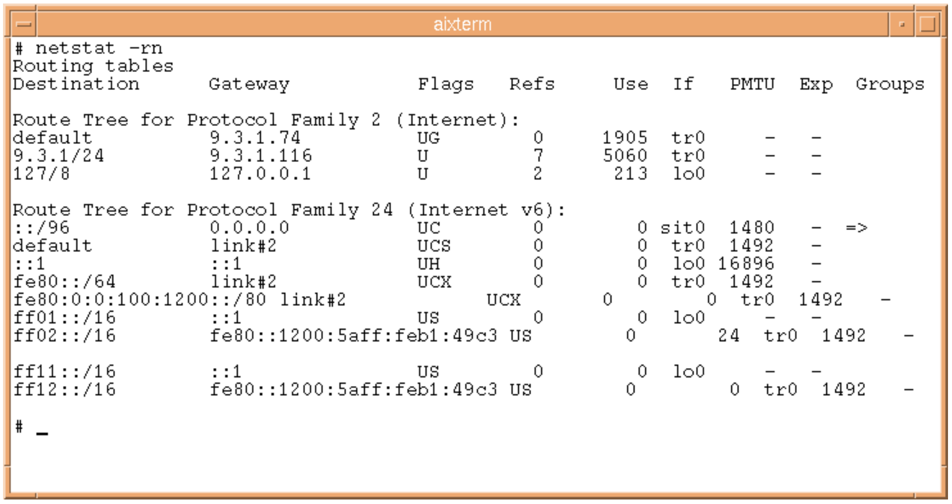
`netstat` displays the contents of various network-related data structures. There are the same number of output formats, but they now show additional IPv6 information. Typical output from the `netstat -ni` command is shown in Figure 61. Note the display of IPv6 addresses.



```
# netstat -ni
Name  Mtu   Network     Address            Ipkts Ierrs    Opkts Oerrs  Coll
lo0   16896 link#1                           441     0      441     0     0
lo0   16896 127         127.0.0.1            441     0      441     0     0
lo0   16896 ::1                              441     0      441     0     0
tr0   1492  link#2      10.0.5a.b1.49.c3   98974     0     6914     0     0
tr0   1492  9.3.1       9.3.1.116          98974     0     6914     0     0
tr0   1492  fe80::100:1200:5aff:feb1:a445  98974     0     6914     0     0
tr0   1492  fe80::1200:5aff:feb1:49c3      98974     0     6914     0     0
sit0  1480  link#3      9.3.1.74               0     0        0     0     0
sit0  1480  ::9.3.1.116                        0     0        0     0     0
# _
```

Figure 61. Typical Output from the netstat -ni Command

Typical output from the `netstat -rn` command is shown in Figure 62.

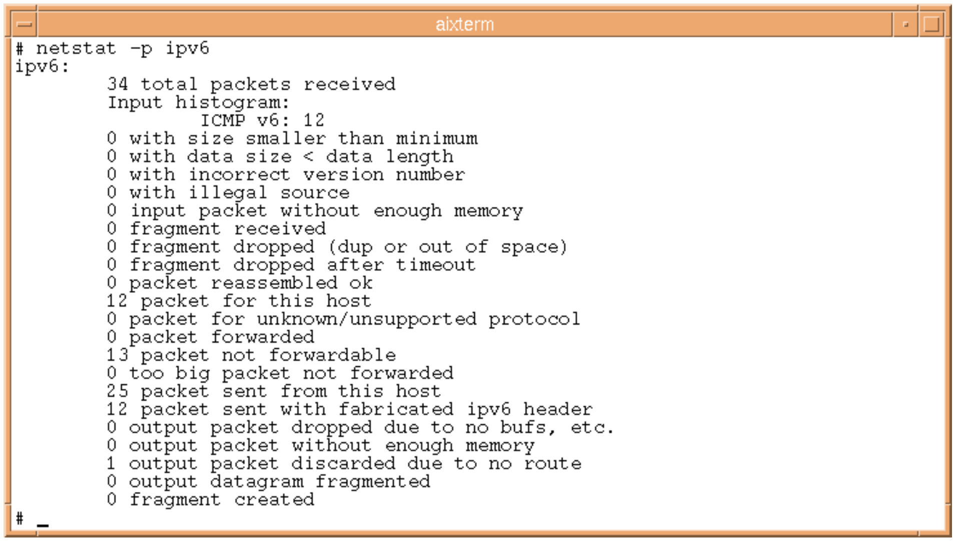


```
# netstat -rn
Routing tables
Destination      Gateway            Flags   Refs     Use  If    PMTU  Exp  Groups

Route Tree for Protocol Family 2 (Internet):
default          9.3.1.74           UG         0     1905  tr0     -   -
9.3.1/24         9.3.1.116          U          7     5060  tr0     -   -
127/8            127.0.0.1          U          2      213  lo0     -   -

Route Tree for Protocol Family 24 (Internet v6):
::/96            0.0.0.0            UC         0        0 sit0  1480   -  =>
default          link#2             UCS        0        0  tr0  1492   -
::1              ::1                UH         0        0  lo0 16896   -
fe80::/64        link#2             UCX        0        0  tr0  1492   -
fe80:0:0:100:1200::/80 link#2             UCX       0        0  tr0  1492    -
ff01::/16        ::1                US         0        0  lo0     -   -
ff02::/16        fe80::1200:5aff:feb1:49c3 US        0       24  tr0  1492   -

ff11::/16        ::1                US         0        0  lo0     -   -
ff12::/16        fe80::1200:5aff:feb1:49c3 US        0        0  tr0  1492   -

# _
```

Figure 62. Routing Tables Shown by netstat -rn Command

Again, note the IPv6 style addresses and the use of the `::` (double colon).

For displaying IPv6 statistics, a new option has been added to `netstat`. By using the -p ipv6 option, `netstat` will display all the statistics associated with IPv6. Typical output from this command is shown in Figure 63.

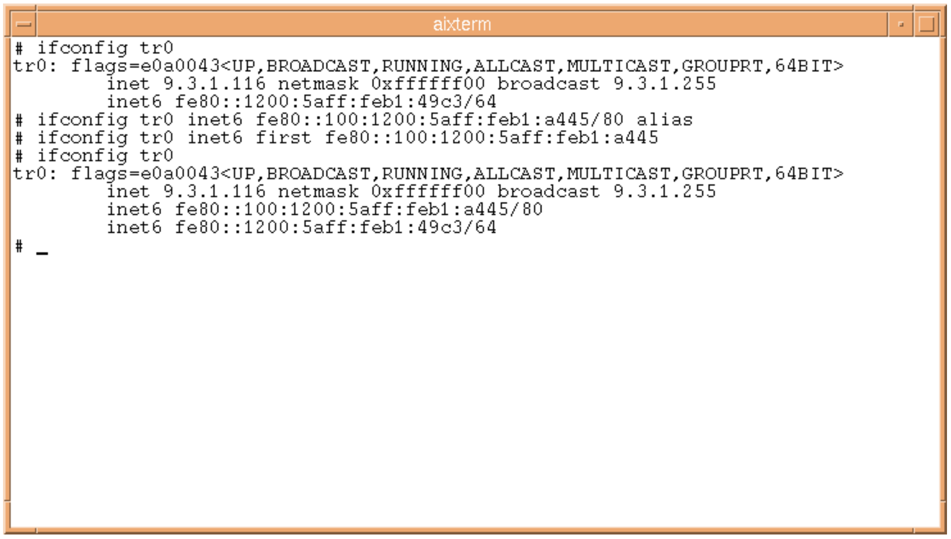


```
# netstat -p ipv6
ipv6:
        34 total packets received
        Input histogram:
                ICMP v6: 12
        0 with size smaller than minimum
        0 with data size < data length
        0 with incorrect version number
        0 with illegal source
        0 input packet without enough memory
        0 fragment received
        0 fragment dropped (dup or out of space)
        0 fragment dropped after timeout
        0 packet reassembled ok
        12 packet for this host
        0 packet for unknown/unsupported protocol
        0 packet forwarded
        13 packet not forwardable
        0 too big packet not forwarded
        25 packet sent from this host
        12 packet sent with fabricated ipv6 header
        0 output packet dropped due to no bufs, etc.
        0 output packet without enough memory
        1 output packet discarded due to no route
        0 output datagram fragmented
        0 fragment created
# _
```

Figure 63. IPv6 Statistics from netstat -p ipv6 Command

8.1.8.2 ifconfig

`ifconfig` is used to assign an address to a network interface or configure network interface parameters. Address family `inet6` has been added to the list of `ifconfig` address family arguments. A new parameter called first can be set. This puts an IPv6 address at the first place on an interface to select it as the source for unbound sockets. Examples of the use of `ifconfig` are shown in Figure 64. This example shows the interface status, binding a site-local address to `tr0`, then making the site-local address the primary IPv6 address. Finally, the results of the actions are displayed.



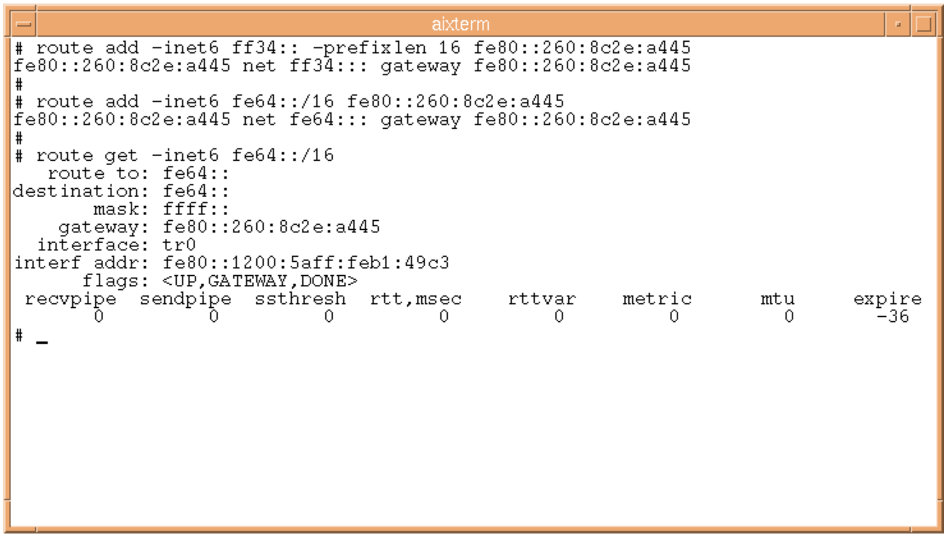
```
# ifconfig tr0
tr0: flags=e0a0043<UP,BROADCAST,RUNNING,ALLCAST,MULTICAST,GROUPRT,64BIT>
        inet 9.3.1.116 netmask 0xffffff00 broadcast 9.3.1.255
        inet6 fe80::1200:5aff:feb1:49c3/64
# ifconfig tr0 inet6 fe80::100:1200:5aff:feb1:a445/80 alias
# ifconfig tr0 inet6 first fe80::100:1200:5aff:feb1:a445
# ifconfig tr0
tr0: flags=e0a0043<UP,BROADCAST,RUNNING,ALLCAST,MULTICAST,GROUPRT,64BIT>
        inet 9.3.1.116 netmask 0xffffff00 broadcast 9.3.1.255
        inet6 fe80::100:1200:5aff:feb1:a445/80
        inet6 fe80::1200:5aff:feb1:49c3/64
# _
```

Figure 64. Example of ifconfig Command Usage

8.1.8.3 route

`route` was modified to process IPv6 addresses and structures. For the IPv6 address family, -inet6 must be used. The length of a destination prefix can be specified with the option -prefixlen or with the syntax prefix/length. There is also a new command to route called `get` that looks up and displays the route for a destination. See Figure 65 for `route` command usage examples.

Note: The `get` command works for all protocol families, not just IPv6.



```
# route add -inet6 ff34:: -prefixlen 16 fe80::260:8c2e:a445
fe80::260:8c2e:a445 net ff34::: gateway fe80::260:8c2e:a445
#
# route add -inet6 fe64::/16 fe80::260:8c2e:a445
fe80::260:8c2e:a445 net fe64::: gateway fe80::260:8c2e:a445
#
# route get -inet6 fe64::/16
   route to: fe64::
destination: fe64::
       mask: ffff::
    gateway: fe80::260:8c2e:a445
  interface: tr0
interf addr: fe80::1200:5aff:feb1:49c3
      flags: <UP,GATEWAY,DONE>
 recvpipe  sendpipe  ssthresh  rtt,msec    rttvar    metric      mtu     expire
       0         0         0         0         0         0        0       -36
# _
```

Figure 65. Example of route Command Usage

8.1.8.4 autoconf6 Command

autoconf6 is a new command with IPv6. The autoconf6 command is used to assign addresses to network interfaces and add some needed routes at boot time. It configures the link-local address, adds the appropriate interface route, and adds a compatibility address if an IPv4 address exists. autoconf6 should be used after configuration of IPv4 and before any IPv6 action. It can take, as an argument, the name of the main IEEE LAN interface.

Table 35. auotconf6 Options

Flag	Description
-v	Verbose output.
-6	Do not install the SIT tunnel interface.

The following shows an example of the typical output that is received when running the autoconf6 -v command:

```
got interface lo0
got interface en0
got interface sit0
default IEEE interface is en0
interface en0 is booting
setup IEEE interface en0
config en0 fe80::260:8c2e:a445/80
add route flags=801
        dest=ff02::
        gateway=fe80::260:8c2e:a445
        mask=ffff::
```

```
add route flags=801
        dest=ff12::
        gateway=fe80::260:8c2e:a445
        mask=ffff::
Sending ND sol for fe80::260:8c2e:a445 on en0
en0 setup good
interface en0 is booted
setup SIT interface sit0
config sit0 9.3.145.129
config sit0 ::903:9181/96
add route flags=901
        dest=::
        gateway=::903:9181
        mask=ffff:ffff:ffff:ffff:ffff:ffff::
setup loopback interface lo0
config lo0 ::1/128
add route flags=801
        dest=ff01::
        gateway=::1
        mask=ffff::
add route flags=801
        dest=ff11::
        gateway=::1
        mask=ffff::
add route flags=901
        dest=::
        gateway=fe80::260:8c2e:a445
        mask=::
setup2 IEEE interface en0
config en0 ::903:9181 alias
         with mask ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffc0
setup2 loopback interface lo0
config lo0 ::7f00:1/128 alias
```

8.1.8.5 TCP/IP Tracing Commands

The TCP/IP tracing commands `tcpdump`, `iptrace`, and `ipreport` have been modified to understand IPv6 addresses and data structures. Tracing IPv6 data flow is therefore as easy as IPv4.

8.1.8.6 traceroute Command

The `traceroute` command now takes advantage of the IPv6 protocol hop limit field and attempts to elicit an ICMPv6 TIME_EXCEEDED response from each gateway along the path to some host. The only mandatory parameter is the destination host name or IPv6 address. It infers the interface MTU at the beginning or uses the packet size argument. It also tries to get the path MTU using ICMPv6 PACKET_TOO_BIG messages. There are three new options with `traceroute`: -d, -f, and -g. These are described in Table 36.

Table 36. traceroute Options

Flag	Description
-d	Turn on socket-level debugging.
-f flow	Set the flow identifier in probe packets (default zero).

Flag	Description
-g gateway_addr	Add gateway_addr to the list of addresses in the IPv6 Source Record Route header. If no gateways are specified, the SRR extra header is omitted.

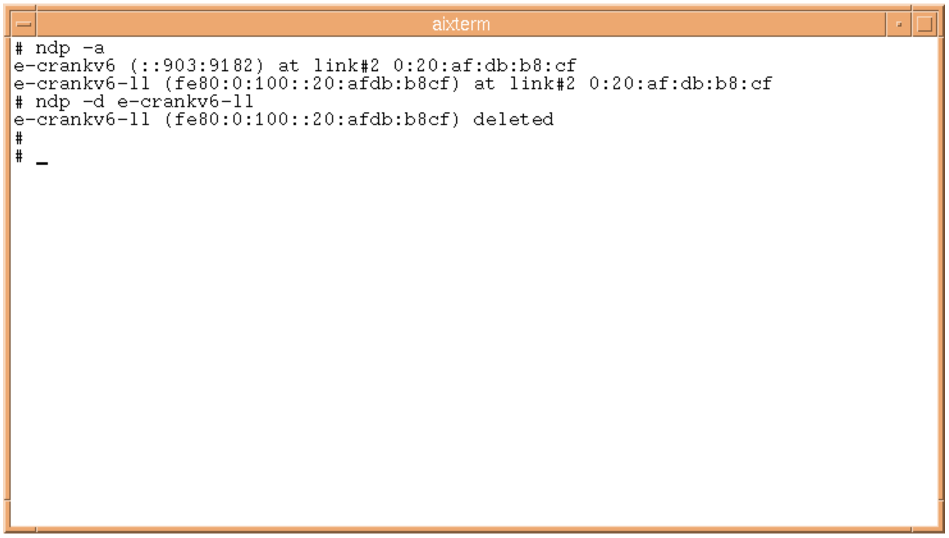
8.1.8.7 ndp Command

The `ndp` command displays and modifies the IPv6-to-Ethernet address translation tables used by the IPv6 Neighbor Discovery Protocol. With no flags, the command displays the current NDP entry for hostname. The host may be specified by name or by number using IPv6 textual notation. Table 37 shows these options.

Table 37. ndp Options

Flag	Description
-a	The program displays all of the current NDP entries.
-d	A superuser may delete an entry for the host called hostname with the -d flag.
-i interface_index	Specify the index of the interface to use when a NDP entry is added with the -s flag (useful with the local-link interface).
-n	Show network addresses as numbers (normally ndp attempts to display addresses symbolically).
-s <media_type> hostname hardware_addr	Hostname with the hardware address hardware_addr for the specified media type. The hardware address is given as six hex bytes separated by colons. The valid media types are: ether, 802.3, fddi, and 802.5. For 802.5 (token-ring), an optional src route can be appended to the hardware_addr. The src route is specified as a list of 16-bit hex numbers separated by colons. The entry is made permanent unless the word temp is given in the command.
-t <if type>	Invoke an interface-specific ndp command. The remaining syntax for the command is based on the interface-specific command.

Figure 66 shows use of the `ndp` command to first show all the current NDP entries and then delete the entry associated with `e-crankv6-11`.



```
# ndp -a
e-crankv6 (::903:9182) at link#2 0:20:af:db:b8:cf
e-crankv6-ll (fe80:0:100::20:afdb:b8cf) at link#2 0:20:af:db:b8:cf
# ndp -d e-crankv6-ll
e-crankv6-ll (fe80:0:100::20:afdb:b8cf) deleted
#
# _
```

Figure 66. Example of ndp Command Usage

8.1.8.8 ndpd-host Command

The `ndpd-host` command manages the NDP (Neighbor Discovery Protocol) for non-kernel activities, such as router discovery, prefix discovery, parameter discovery, and redirects. `ndpd-host` deals with the default route, including default router, default interface, and default interface address. Table 38 provides the flags for the `ndpd-host` command.

Table 38. ndpd-host Options

Flag	Description
-d	Enables debugging (exceptional conditions and dump)
-v	Logs all interesting events (daemon.info and console)

8.1.8.9 inetd Daemon

The `inetd` daemon creates AF_INET sockets for services that are tcp4 and udp4 and AF_INET6 sockets for services that are tcp6 and udp6. If the -6 option is used, `inetd` creates AF_INET6 sockets for services that are TCP and UDP. Otherwise, it creates AF_INET sockets for these services. The new members of the servtab structure are se_family (address family) and se_ctrladdr_size (for the differences in size of sockaddr_in and sockaddr_in6). Also, the se_un member was modified to include the sockaddr_in6 structure. The config() function has an added AF_INET6 case.

8.1.9 IPv6 Socket Library Support

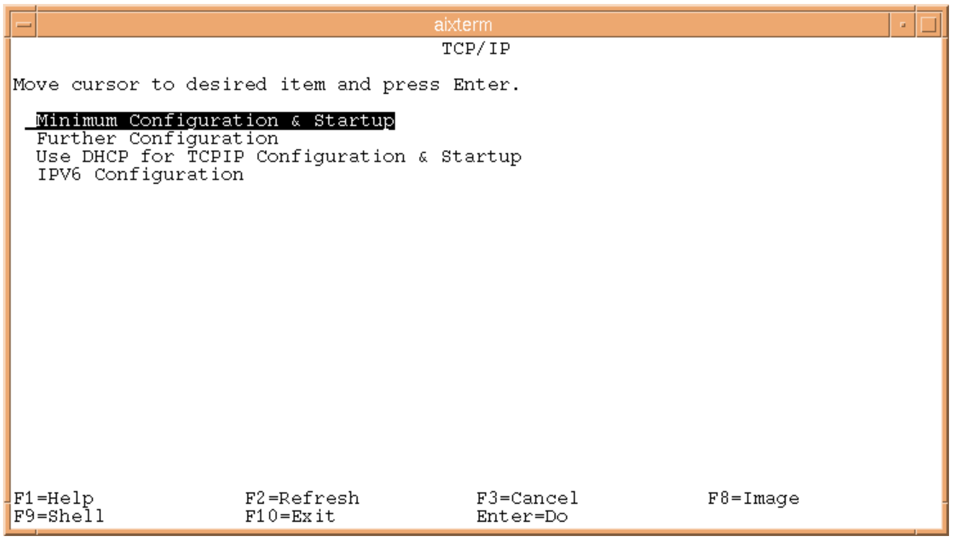
Socket library support has been modified to allow for application developers to begin porting to IPv6:

- IPv6 data structures (include files define these).
- socket(), bind() and connect() support the AF_INET6 address family and the sockaddr_in6 address structure.
- ASCII print function is provided with inet_pton() and inet_ntop().
- Address/name translations with IPv6 addresses.
- Interface/index information retrieval.
- Address manipulation/query (in6_ifXXX(), XXX_ADDR6()).
- IPv6-specific getsockopt() and setsockopt() calls.

8.1.10 System Management Changes and Additions

The recommended method of configuring IPv6 is through SMIT. New SMIT panels have been added and existing ones changed to achieve this. Existing system configuration commands, such as `mkdev` and `chdev`, can also be used but will not be described here.

The IPv6 configuration panels are reached through a new option added to the standard TCP/IP panel, as shown in Figure 67.

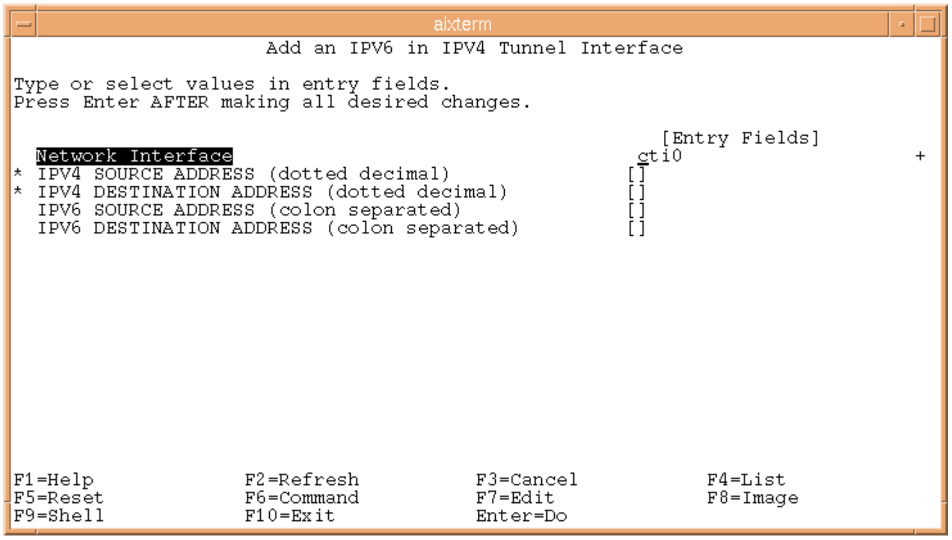


```
                                   TCP/IP

Move cursor to desired item and press Enter.

  Minimum Configuration & Startup
  Further Configuration
  Use DHCP for TCPIP Configuration & Startup
  IPV6 Configuration















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

Figure 67. New SMIT TCP/IP Configuration Panel Entries

Many of the IPv6 SMIT panels are virtually identical, in both layout and function, to their IPv4 equivalents, so we will not give examples in this publication. One panel that does not exist in non-IPv6-enabled releases is the panel for configuration of tunnel interfaces. This is shown in Figure 68.

When adding an interface, both the IPv6 and IPv4 source and destination addresses must be specified.



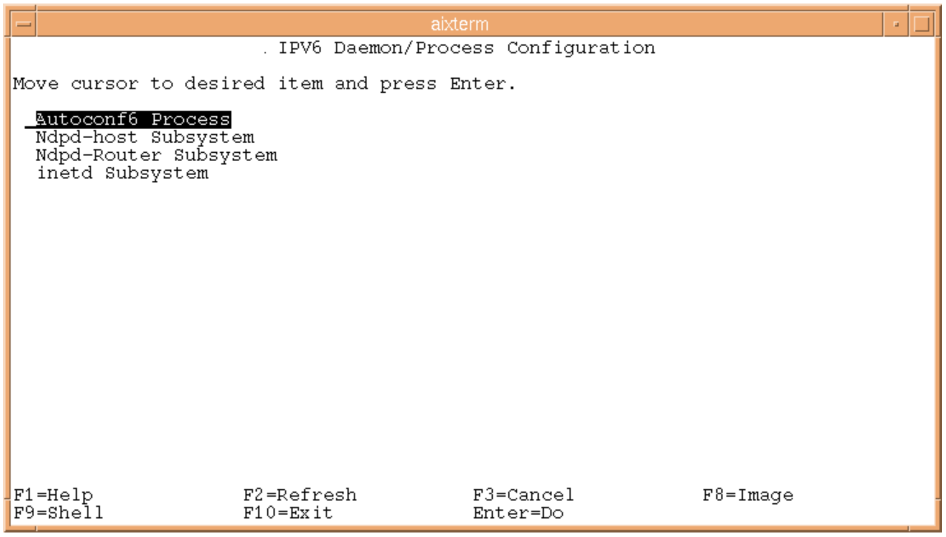
```
aixterm
                     Add an IPV6 in IPV4 Tunnel Interface

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

                                                        [Entry Fields]
  Network Interface                                    cti0                  +
* IPV4 SOURCE ADDRESS (dotted decimal)                []
* IPV4 DESTINATION ADDRESS (dotted decimal)           []
  IPV6 SOURCE ADDRESS (colon separated)               []
  IPV6 DESTINATION ADDRESS (colon separated)          []













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

Figure 68. Configuring IPv6 Tunnel Interfaces with SMIT

This panel executes the `chdev` command that calls `ifconfig` and adds an entry to the ODM.

There is also a SMIT panel that allows configuration of the IPv6 daemons, shown in Figure 69. It is reached through the SMIT fastpath, `smit daemon6`.

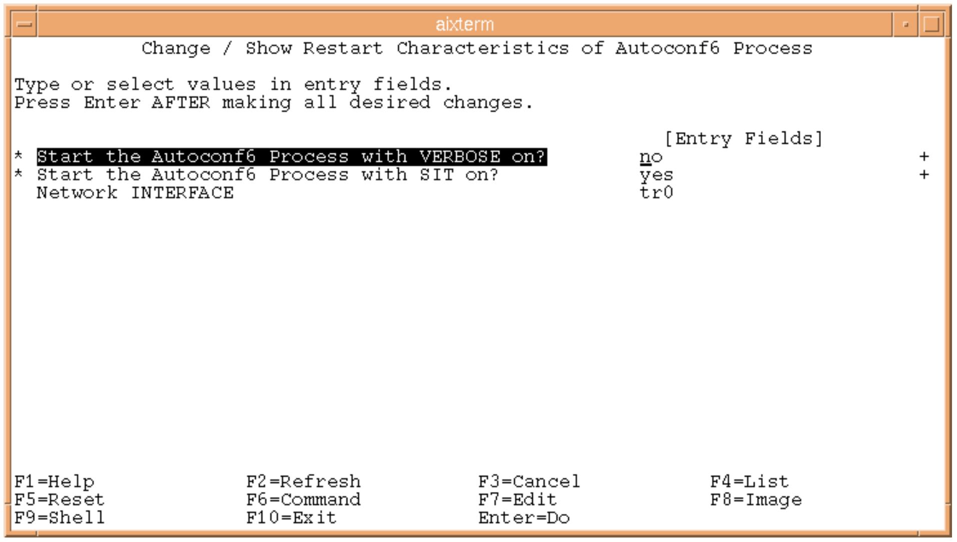
This panel presents the user with options to configure the autoconf6 process, the ndpd-host subsystem, and also provides a link to the inetd subsystem configuration.



```
                    IPV6 Daemon/Process Configuration

Move cursor to desired item and press Enter.

  Autoconf6 Process
  Ndpd-host Subsystem
  Ndpd-Router Subsystem
  inetd Subsystem

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

Figure 69. IPv6 Daemon Configuration SMIT Panel

If the **Change / Show** option is selected from the Autoconf6 SMIT configuration panel, the following panel is displayed. The VERBOSE parameter instructs the autoconf6 process to display what it is doing and also show any errors it is encountering. The SIT option allows the autoconf6 process to be started without the SIT interface and IPv4-compatible parameters being installed. Once you have made any changes you want to the default parameters and pressed the **Enter** key, SMIT uncomments the autoconf6 entry in the /etc/rc.tcpip file and starts the autoconf6 process.



```
           Change / Show Restart Characteristics of Autoconf6 Process

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

                                                       [Entry Fields]
* Start the Autoconf6 Process with VERBOSE on?        no                    +
* Start the Autoconf6 Process with SIT on?            yes                   +
  Network INTERFACE                                   tr0

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

Figure 70. IPv6 autoconf6 SMIT Configuration Panel

8.1.11 IPv6 and IPSec-Related RFCs Implementation

Table 39 shows the RFCs that have been implemented in the IP Version 6 support included with AIX Version 4.3.0. Consult these RFCs, or their successors, for the latest information on IP Version 6 protocols.

Table 39. RFCs Implemented in AIX Version 4.3.0

RFC number	RFC Title
1825	Security Architecture for the Internet Protocol
1826	IP Authentification Header (AH)
1827	IP Encapsulating Security Payload (ESP)
1828	IP Authentification Using Keyed MD5
1829	The ESP DES-CBC Transform
1883	Internet Protocol, Version 6 (IPv6) Specification
1884	IP Version 6 Addressing Architecture
1885	Internet Control Message Protocol (ICMPv6) for IPv6
1933	Transition Mechanisms for IPv6 Hosts and Routers
1970	IPv6 Stateless Address Autoconfiguration

RFC number	RFC Title
1971	Neighbor Discovery for IP Version 6 (IPv6)

8.2 IP Security Enhancements (4.3.1)

AIX 4.3.1 has added new functions to IP Security:

- Triple-DES is an additional choice for U.S. and Canada customers.
- Performance improvements have been made to the CDMF and DES encryption.
- The filter table now supports unlimited number of rules.
- Any combination of installed authentication and encryption algorithms can be used for ESP with Authentication.

In addition, new parameters have been added to the following IPSec commands:

- `exptun`
- `imptun`
- `mktun`
- `rmfilt`

8.3 IP Security Enhancements (4.3.3)

AIX 4.3.3 provides following IPSec enhancements, which will be discussed the sections that follow:

- Improved RAS
- Perfect Forward Secrecy Support
- eNetwork Firewall Support
- Web-Based System Manager Panel Enhancements
- IP Address Ranges Support
- On Demand Tunnel Support

8.3.1 Improved RAS

Several RAS enhancements are added to IPSec. They are:

- Better tracing of IKE traffic

On AIX 4.3.3, logging for IKE messages are enabled. The contents of the IKE messages which are sent to the remote communication peer is parsed and displayed in the log file specified in the configuration file /etc/isakmpd.conf. If the ISAKMP packet is encrypted, only the ISAKMP header contents is printed in the log file. Also the logging capability can be limited to messages sent to a specific host by specifying the IP address. The syntax of /etc/isakmp.conf is:

```
Log_Level Log_File IP_Address_Remote_Host
```

IP_Address_Remote_Host is optional.

- `iptrace` and `ipreport` support

  The `iptrace` and `ipreport` command supports logging and displaying ISAKMP headers and AH and ESP information.

- SYSLOG support

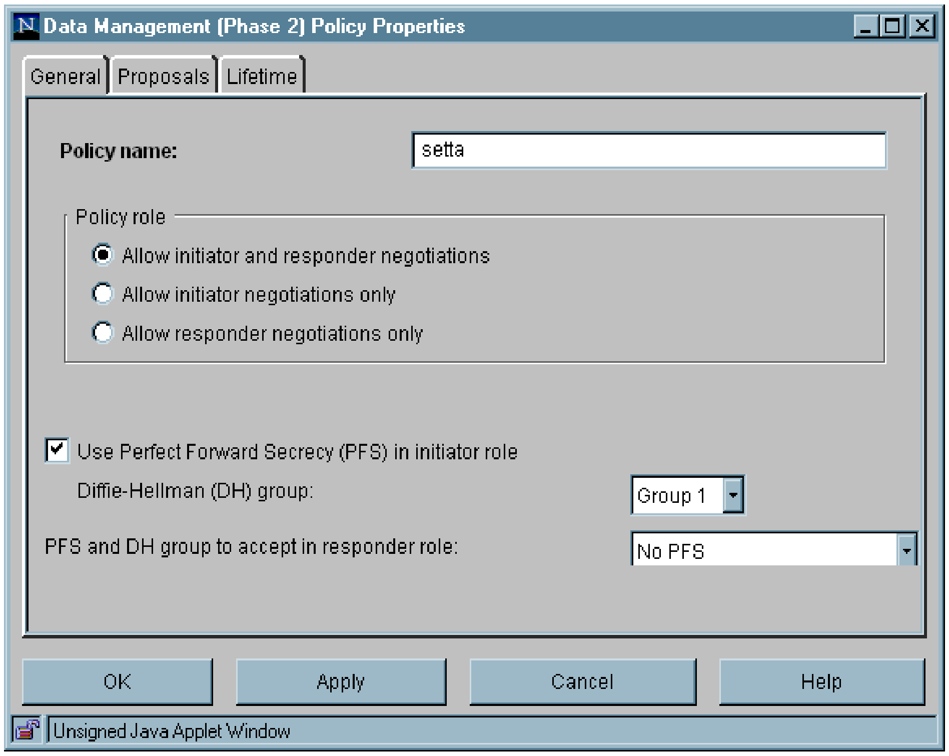
  There are four levels of logging in the ISAKMP daemon. They are None, Events, Error, and Information. Out of the four levels of logging the Events level of logging messages are logged to SYSLOG instead of logged to the file specified in /etc/isakmpd.conf.

- AIX auditing support

  IPSec supports AIX auditing subsystem for better security auditing.

8.3.2 Perfect Forward Secrecy Support

The Perfect Forward Secrecy (PFS) means that each key uses fresh keys and that information on new keys cannot be derived from knowing the values of the old keys. The PFS is supported when refreshing keys in IKE phase 2 tunnels. Figure 71 shows **Data Management (Phase 2) Policy Properties** panel and the values that you can be specify. This panel can be launched by selecting **Data Policy->New Data Management Policy...** on **Data Management (Phase 2) Protection Policies** panel.


Data Management (Phase 2) Policy Properties

General | Proposals | Lifetime

Policy name: setta

Policy role

- (●) Allow initiator and responder negotiations
- ( ) Allow initiator negotiations only
- ( ) Allow responder negotiations only

[✓] Use Perfect Forward Secrecy (PFS) in initiator role

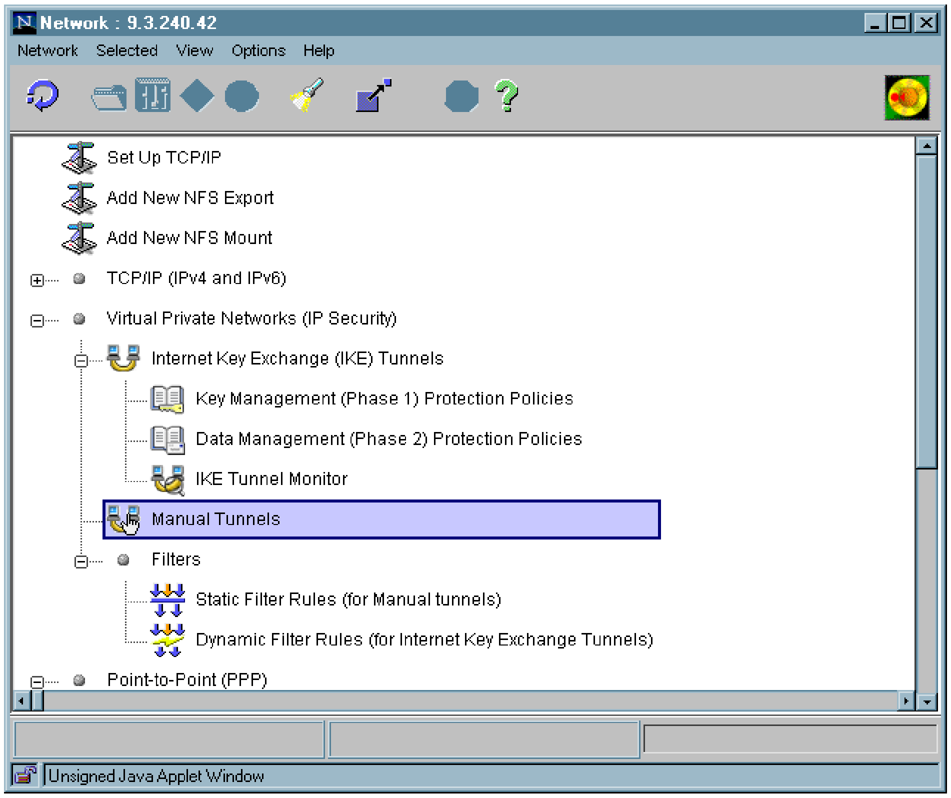
Diffie-Hellman (DH) group: Group 1

PFS and DH group to accept in responder role: No PFS

Figure 71. Perfect Forward Secrecy Support WSM Configuration Panel

8.3.3 Web-Based System Manager Panel Enhancements

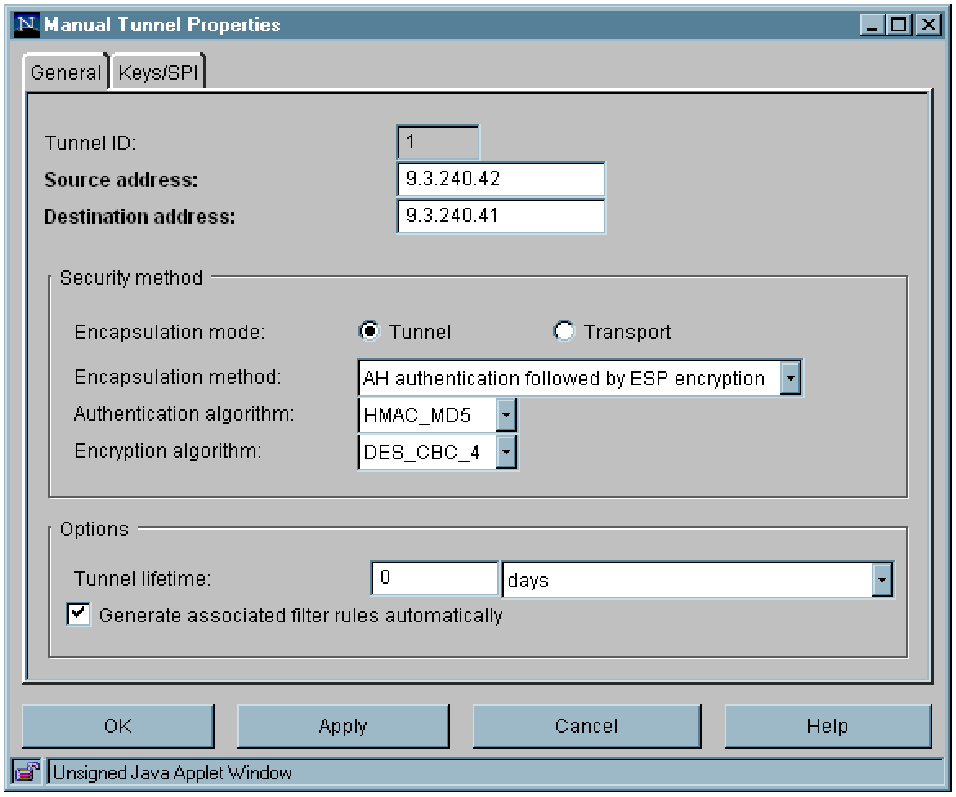
On AIX 4.3.3, you can create, delete and change manual tunnels and static filters using Web-Based System Manager panels. Figure 72 shows the new **Network** application panel.



Set Up TCP/IP
Add New NFS Export
Add New NFS Mount
TCP/IP (IPv4 and IPv6)
Virtual Private Networks (IP Security)
Internet Key Exchange (IKE) Tunnels
Key Management (Phase 1) Protection Policies
Data Management (Phase 2) Protection Policies
IKE Tunnel Monitor
Manual Tunnels
Filters
Static Filter Rules (for Manual tunnels)
Dynamic Filter Rules (for Internet Key Exchange Tunnels)
Point-to-Point (PPP)
Unsigned Java Applet Window

Figure 72. New WSM Network Application Panel

Figure 73 shows the manual tunnel configuration panel. This panel can be launched by selecting **Tunnel->New Manual Tunnel...** menu from **Manual Tunnel** panel.



Tunnel ID: 1
Source address: 9.3.240.42
Destination address: 9.3.240.41

Security method

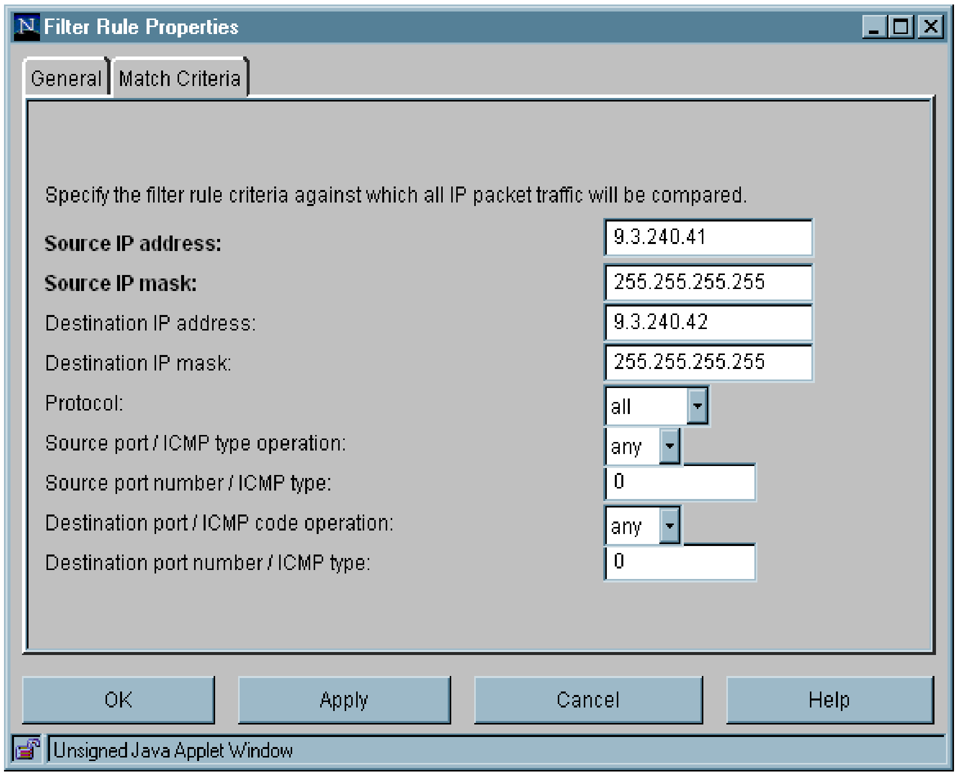
Encapsulation mode: [✓ Tunnel] Transport
Encapsulation method: AH authentication followed by ESP encryption
Authentication algorithm: HMAC_MD5
Encryption algorithm: DES_CBC_4

Options

Tunnel lifetime: 0 days
✓ Generate associated filter rules automatically

Figure 73. Manual Tunnel Support WSM Configuration Panel

Figure 74 shows filter rule properties launched by selecting **Filter Rule->New Filter Rules...** menu on **Static FIlter Rules (for Manual tunnels)** panel.



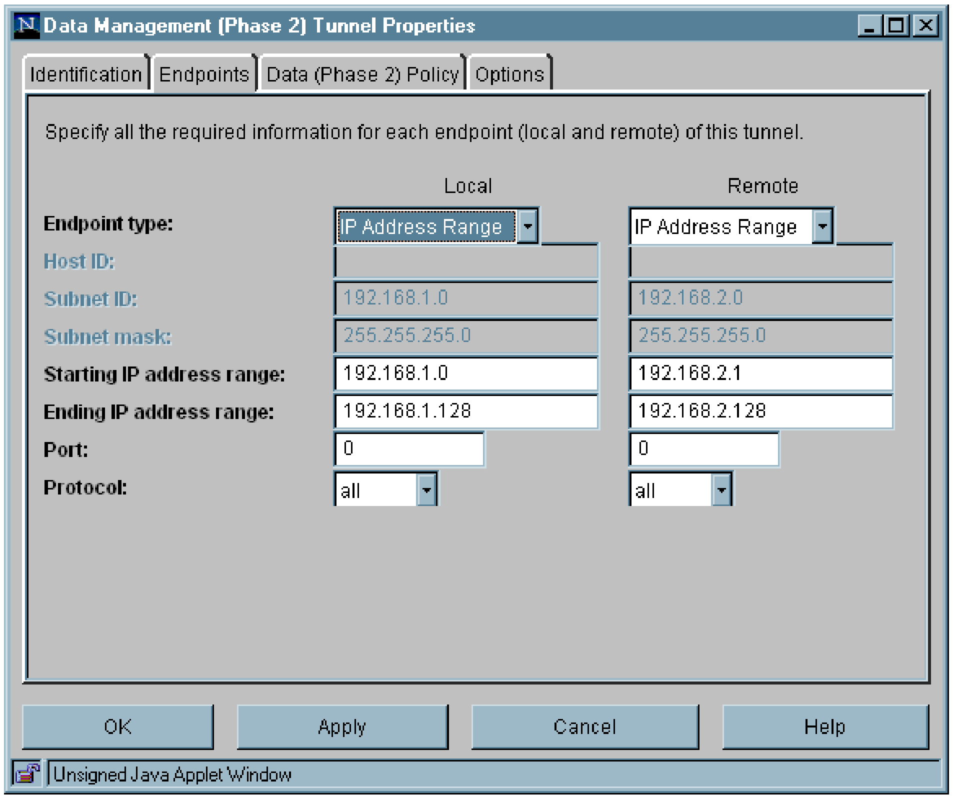
Specify the filter rule criteria against which all IP packet traffic will be compared.

Source IP address:	9.3.240.41
Source IP mask:	255.255.255.255
Destination IP address:	9.3.240.42
Destination IP mask:	255.255.255.255
Protocol:	all
Source port / ICMP type operation:	any
Source port number / ICMP type:	0
Destination port / ICMP code operation:	any
Destination port number / ICMP type:	0

Figure 74. Static Filter Support WSM Configuration Support

8.3.4 IP Address Ranges Support

On AIX 4.3.3, users of ISAKMP phase 2 tunnel (the traffic between the uses are authenticated/encrypted using protocols defined in phase 1) can be specified by IP address ranges as well as hosts and subnets. This capability gives administrators a finer granularity for security administration. Figure 75 shows the configuration panel in that panel IP address ranges can be specified. This panel is launched by selecting **Tunnel->New Tunnel->New Data Management Tunnel...** in **Internet Key Exchange (IKE) Tunnels** panel.


Data Management (Phase 2) Tunnel Properties
Identification
Endpoints
Data (Phase 2) Policy
Options
OK
Apply
Cancel
Help
Unsigned Java Applet Window

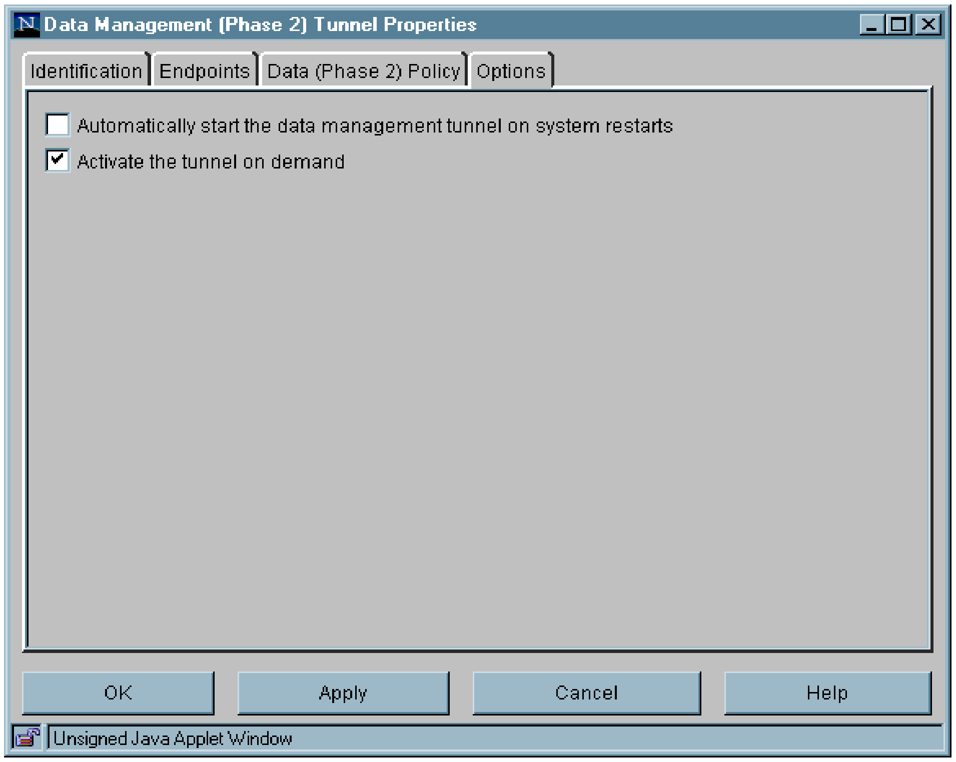
Specify all the required information for each endpoint (local and remote) of this tunnel.

	Local	Remote
Endpoint type:	IP Address Range	IP Address Range
Host ID:		
Subnet ID:	192.168.1.0	192.168.2.0
Subnet mask:	255.255.255.0	255.255.255.0
Starting IP address range:	192.168.1.0	192.168.2.1
Ending IP address range:	192.168.1.128	192.168.2.128
Port:	0	0
Protocol:	all	all

Figure 75. IP Address Ranges Support WSM Configuration Panel

8.3.5 On Demand Tunnel Support

On previous versions of AIX, when tunnels are created, they can be activated now or, now and on reboot. On AIX 4.3.3, they can be activated on demand without manual intervention of administrators. The tunnels that have this capability is called on demand tunnels. On demand tunnels can be created when defining properties for IKE phase 2 tunnels. Figure 76 shows the panel with **Activate the tunnel on demand** check box. This panel can be displayed by selecting **Options** tab on the same panel displayed in Figure 75.


Data Management (Phase 2) Tunnel Properties

Identification | Endpoints | Data (Phase 2) Policy | Options

[ ] Automatically start the data management tunnel on system restarts
[✓] Activate the tunnel on demand

Figure 76. On Demand Tunnel Support WSM Configuration Panel

8.4 TCP/IP Command Security Enhancement (4.3.1)

AIX Version 4.3.1 offers secure remote TCP/IP commands: `rsh`, `rcp`, `rlogin`, `telnet`, and `ftp`. With this capability, Kerberos 5 authentication is used between these commands and server daemons, avoiding the need for user passwords to pass in the clear on the network. Instead, Kerberos 5 credentials are used to authenticate users. User credentials can be forwarded to the server.

8.5 Dynamic Host Configuration Protocol Enhancements (4.3.1)

A new command `dadmin` has been added to assist the DHCP administrator.

The `dadmin` command lets the DHCP administrator query and modify the state of his DHCP servers' databases. It gives the administrator the ability to locally or remotely query the DHCP server for the status of an IP address, query for a pool of IP addresses, query for a client, delete an IP address mapping, refresh the server, and change the server's tracing level. The `dadmin` command is backwards compatible with previous AIX release DHCP servers to list their IP address status and refresh.

When querying for IP address information, the `dadmin` command returns the IP address's status, and depending on this, may return the lease duration, start lease time, last leased time, whether the server supports DNS A record updates for this IP address, and the client identifier that is mapped to this IP address.

When querying for client information, the `dadmin` command returns the client's IP address and IP address status, whether the server supports DNS A record updates for this IP address, the last time the client was given any IP address, and the hostname and domain name used by the client.

When modifying the server tracing level, the `dadmin` command sets and returns the server tracing level in the form of a tracing mask. This mask represents a bit string, where each bit represents whether a specific log item is being traced by the server (see DHCP Server Configuration in the online documentation). From least significant to most significant order, these log items are LOG_NONE, LOG_SYSERR, LOG_OBJERR, LOG_PROTOCOL and LOG_PROTERR (same value), LOG_WARN and LOG_CONFIG (same value), LOG_EVENT and LOG_PARSEERR (same value), LOG_ACTION, LOG_INFM, LOG_ACNTING, LOG_STAT, LOG_TRACE, LOG_START, and LOG_RTRACE.

8.6 TFTP Block Size Option (4.3.1)

The implementation of TFTP in AIX Version 4.3.1 has been enhanced to include the Block Size Option proposed in RFC 1783. Older implementations of TFTP relied upon a fixed block size of 512 octets that although easy to code and implement in the limited ROM space available in some clients, proves to be very inefficient on LANs whose MTU size may be 1500 octets or more.

This implementation of TFTP has modified the TFTP Read Request and TFTP Write Request packets to include a block size option. When a client sends a read or write request, it can now include an option to request that the server uses a block size other than 512 octets. If the server is willing to accept the blocksize option, it responds with an Option Acknowledgment (OACK) containing the blocksize that must be equal to, or smaller than, that requested by the client. If the client is unable to accept the blocksize returned by the server, it must reply with an error packet and terminate the transfer.

Tests on the performance of TFTP with different blocksizes have revealed that file transfer time can be reduced by as much as 80 percent by using larger block sizes.

Benefits are only gained when a client is able to perform block size negotiation. At the time of writing, the IBM Network Station TFTP client has this function. This change does not affect the current AIX TFTP client application.

Note

If the blocksize exceeds the MTU of the path, then fragmenting will occur, and performance improvements will be reduced by the need to perform reassembly of packets.

8.7 IPv6 Routing Support (4.3.2)

The first release of IPv6 support in AIX 4.3 was a host implementation. It did not support routing between interfaces on the same server. Also, the routing applications were not well defined and not available in the INRIA code base to interpret IPv6 routing protocols.

AIX Version 4.3.2 supports IPv6 gateway capability in addition to IPv6 host supported in AIX Version 4.3. Version 6.0 of gated from Institut National de Recherche en Informatique et en Automatique (INRIA) is ported from INRIA to AIX 4.3.2 with IPV6 support so that gated and some of its routing protocols can manipulate IPv6 addresses. Some changes have been made to the AIX kernel, netinet kernel extension, and non-gated applications to support IPv6 routing.

8.7.1 Gated Version 6.0

The gated daemon provides gateway routing functions for the following protocols:

- Routing Information Protocol (RIP)
- Routing Information Protocol Next Generation (RIPng)
- Exterior Gateway Protocol (EGP)
- Border Gateway Protocol (BGP) and BGP4+

- Defense Communications Network Local-Network Protocol (HELLO)
- Open Shortest Path First (OSPF)
- Intermediate System to Intermediate System (IS-IS)
- Internet Control Message Protocol (ICMP) / Router Discovery routing protocols
- Simple Network Management Protocol (SNMP)

The gated process can be configured to perform all of these protocols or any combination of them.

Dynamic routing tables change according to the network conditions. They are built from information passed to gated by the routing protocols.

Each protocol performs the same basic functions, in that they determine the best route for a destination and then distribute the routing information to other systems on the network. Each protocol performs these tasks based on their own particular algorithms.

Gated collects the routing information from each protocol and then selects the best route per destination among them. Each routing protocol has its own way of calculating the cost of a route, that is, the metric value of a route. For example, RIP uses distance (hop count) and HELLO uses time (delay in milliseconds) to determine the cost of a route. Gated uses its own preference values to determine who has the best route. Preference values range from 0-255 with lower numbers indicating more preferred routes. Once gated has selected its best route to a destination, it puts that route in its routing table and later installs it into the kernel's routing table.

There are three new commands delivered along with the new gated supporting IPv6 routing. They are `gdc`, `ospf_monitor`, and `ripquery`.

Following are some brief descriptions of these new introduced commands:

8.7.1.1 gdc Command

The `gdc` command provides an operational user interface for gated. The interface is user-oriented for the operation of the gated routing daemon. It provides supports for:

- Starting and stopping gated daemon
- Delivering signals to manipulate the gated daemon
- Maintenance and syntax checking of configuration files
- Removal of state dumps and core dumps

The `gdc` command can reliably determine the running state of gated and produces a reliable exit status when errors occur, making it advantageous for use in shell scripts that manipulate gated.

8.7.1.2 ospf_monitor Command

The `ospf_monitor` command queries OSPF routers to provide the detailed statistics. It monitors the OSPF gateways.

8.7.1.3 ripquery Command

The `ripquery` command sends a RIP request or POLL command, to a RIP gateway to request all the routes known by the gateway. It queries the RIP gateways. This command is used as a tool for debugging gateways.

Gated has the following enhancements:

- SRC support is added to `gated`. You can start gated using SRC or `gdc`.
- Message catalog is added for `gated`, `ripquery`, `ospf_monitor`, and `gdc`.

Following IPv6 protocols are added in gated:

- IPv6 Routing Information Protocol (RIPNG)
- IPv6 Internet Control Message Protocol (ICMPv6)
- IPv6 Border Gateway Protocol (BGP4+)

8.7.2 IPv6 Routing Functions

The changes for IPv6 routing in INRIA are merged into the AIX netinet and kernel. Also, some of the routing applications are ported to AIX, specifically, ndpd-router and updates to ndpd-host and ndp.

AIX 4.3.2 adds the following IPv6 functions:

- IPv6 unicast routing support
- IPv6 multicast routing support (no mrouted6)
- IPv6 anycast address support - mostly receive processing
- IPv6 multi-homed link local and site local support. The reason for adding multi-homed host support is because most routers are multi-homed and need the ability to reference different hosts on different links at the link local and site local levels.

8.7.2.1 IPv6 Unicast Routing

A packet sent to a unicast address is delivered to the interface identified by that address.

There are two main aspects added in AIX 4.3.2: `ndpd-router` and the some netinet changes.

The first major function is `ndpd-router`. The `ndpd-router` command provides a protocol engine for NDP and RIPng. It provides a simple method to distribute routes between gateways. This function is redundantly provided by `gated`. The routing specific mechanisms, such as router advertisements and answers to router solicitations are handled by ndpd-router. RIPng is the follow-on for RIP with regards to IPv6.

The other major function is to provide extensions to the icmpv6 support so that all the NDP messages are supported. The new messages are for handling redirects.

The modifications to the IP input path allows the forwarding of IPv6 packets between interfaces on the same machine. This function is identical to the IPv4 low-level routing function. The ip6forwarding function handles redirecting incoming packets.

8.7.2.2 IPv6 Multicast Routing

The use of Internet Protocol (IP) multicasting enables a message to be transmitted to a group of hosts, instead of having to address and send the message to each group member individually. The Class D Internet addressing is used for multicasting.

In general, there are two pieces to multicast routing. One piece is a method to determine where the multicast message should be sent, and the other is a method to dynamically change the place the packet goes based on network conditions. In IPv4, the `mrouted` daemon handles both functions with the help of IGMP. In IPv6, multicasting, and its related membership function, is all handled in icmpv6. Usually, this is done by maintaining a table similar to a routing table that is used to direct messages. In IPv4, this table is maintained in netinet and updated by `mrouted`. In IPv6, INRIA adds it as an additional routing table. This way the route command could be used to manipulate static multicast routes.

8.7.2.3 IPv6 Anycast Address Support

In AIX 4.3, the anycast is similar to INRIA. This method involved creating anycast addresses and associating them with a specific interface. This is acceptable because, on incoming packet reception, AIX would talk to all the addresses on the address lists. But to modify these addresses, a user would have to remember the associated interface. INRIA has moved anycast addresses on to a private list. The management is separate from the normal addresses. In fact, a command called any6 is added to manipulate these addresses.

8.7.2.4 IPv6 Multi-Homed Support

AIX 4.3.0 is supported for IPv6 and multi-homed global addresses. This falls out naturally from the IPv6 routing table. The problem is with site local and link local addresses. The problem with site local addresses is the definition of network boundaries. This is not handled by the kernel. The problem with link local addresses is which interface to go out of for doing the initial NDP neighbor solicitations. The code defines a #define, MULTI_HOMED, that can have four values. Each value has different actions and operations.

The AIX IPv6 will support the host requirements from the following RFCs:

- RFC 1883 - Internet Protocol, Version 6 (IPv6) Specification
- RFC 1884 - IP Version 6 Addressing Architecture
- RFC 1885 - Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6)
- RFC 1886 - DNS Extensions to support IP Version 6
- RFC 1887 - An Architecture for IPv6 Unicast Address Allocation
- RFC 1970 - Neighbor Discovery for IP Version 6 (IPv6)
- RFC 1971 - IPv6 Stateless Address Autoconfiguration
- RFC 1972 - A Method for the Transmission of IPv6 Packets over Ethernet networks
- RFC 1981 - Path MTU Discovery for IP Version 6
- RFC 2019 - A Method for the Transmission of IPv6 Packets over FDDI networks

For a complete list of all RFCs and Internet drafts pertaining to IPv6, refer to `http://www.ietf.org/html.charters/ipngwg-charter.html`. This is an evolving list, and AIX conforms to a subset of these.

8.7.3 Commands Changed

The following AIX commands have been changed to support IPv6 routing:

- `ndpd-host`
- `ndp`
- `route`
- `autoconf6`

- `hostnew`
- `nslookup`
- `ifconfig`

There are new no options added:

- ip6forwarding

  This option works like the existing ipforwarding only for IPv6 packets.

  The IP packets forwarding function will be enabled by the following command:

  ```
  #no -o ip6forwarding=1
  ```

  The value of ipforwarding specifies whether the kernel should forward packets. The default value of 0 prevents forwarding of IP packets. A value of 1 enables forwarding.
- multi_homed
- main_if6
- main_site6
- site6_index

8.8 Gratuitous ARP (4.3.3)

Gratuitous ARP, available starting with AIX Version 4.3.3, provides the following two features:

- If the hardware address for an IP address changes, then all the other hosts on the cable that has an ARP entry with the old hardware address update their entry with the new hardware address.
- When assigning an IP address to an interface it is possible to detect if another host is already configured with the same IP address.

This behavior is achieved by sending an ARP request looking for the machine’s own IP address when an IP address is assigned to an interface, either during the system startup time or when the system is already up (on both occasions usually using `ifconfig`). All interfaces that support ARP, except for ATM, have in AIX 4.3.3 this features: Ethernet, token ring, FDDI, to name a few.

According to the ARP protocol, the receiver of an ARP packet updates his ARP entry with the sender's hardware address if he already has an entry for the sender's IP address.

When machine B receives an ARP request from machine A and detects that the sender’s IP address is equal to its own but the hardware address is different, it logs this misconfiguration on his system so that the system administrator is warned about it. Machine B then responds to machine A with his hardware address and machine A would detect the same misconfiguration because the machine B’s IP address in the ARP response packet would be same as its but the hardware address would be different. Then machine A detects the error too.

The `ifconfig` command does not fail when duplicate addresses are in use in the network, but the system administrator can detect the problem looking at the error log on both machines finding an entry similar to the following:

```
LABEL:          AIXIF_ARP_DUP_ADDR
IDENTIFIER:     FE2DEE00

Date/Time:       Thu Jul 29 10:16:58
Sequence Number: 12
Machine Id:      006152004C00
Node Id:         433b
Class:           S
Type:            PERM
Resource Name:   SYSXAIXIF

Description
DUPLICATE IP ADDRESS DETECTED IN THE NET

Failure Causes
ARP RESPONSE RECEIVED FOR MY IP ADDRESS

        Recommended Actions
        CONTACT NETWORK ADMINISTRATOR

Detail Data
DUPLICATE IP ADDRESS
0903 F028
MAC ADDRESS
0004 AC61 73FE
```

The error log entry provides the MAC address of the other machine and the value if the duplicated IP address. The IP address is provided in hexadecimal format, so in the previous example it is 09.03.F0.28 that is 9.3.240.40.

8.9 Enhancement to the ifconfig Command (4.3.2)

The new flags introduced to `ifconfig` command are:

```
ifconfig [ -m ] Interface [ ProtocolFamily ] Interface ProtocolFamily
ifconfig -a [ -m ] [ -d ] [ -u ] [ ProtocolFamily ]
```

Table 40 provides the descriptions of the new ifconfig flags:

Table 40. ifconfig New Flags for Display Interface Information

Flags	Descriptions
-a	The -a flag can be used instead of an interface name. This flag instructs `ifconfig` to display information about all interfaces in the system.
-d	The -d flag displays interfaces that are down.
-m	If the -m flag is passed before an interface name, `ifconfig` will display all of the supported media for the specified interface.
-u	The -u flag displays interfaces that are up.

8.10 Latest BIND DNS (NameD) Support (4.3.2)

The Domain Naming System (DNS) is a method to distribute a large database of IP addresses, hostnames, and other record data across administrative areas. The end result is a distributed database maintained in sections by authorized administrators per domain.

AIX, similar to other UNIX platforms, offers the BIND DNS server. BIND, or the Berkeley Internet Name Daemon, is a DNS server implementation provided by the Internet Software Consortium. It has become the standard for DNS server implementations and a benchmark for DNS server intercompatibility.

AIX 4.3.2 incorporates IBM DNS value-added functions to the latest level of BIND, Version 8.1.2. This involves adding IBM secure dynamic DNS update protocol and incremental zone transfers to BIND 8.1.2, as well as extending this BIND’s NOTIFY ability and parameter configuration.

The following are the new functions provided Bind Version 8.1.2:

• Secure dynamic DNS updates

  Currently, BIND offers only the unsecured RFC 2136 update protocol. This is an insufficient offering to customers desiring to implement a dynamic DNS environment in their networks. The secure update protocol is added as the solution for RFC 2136's insecure shortcomings and to provide backward compatibility with current AIX dynamic DNS customers.

• Incremental zone transfer

Implement the RFC 1995 Incremental Zone Transfer protocol. This protocol defines a method through which secondary DNS servers can update their existing zone data to incorporate all the cumulative changes to the primary zone since the last transfer. This protocol supersedes the performance of ordinary zone transfers by limiting the amount of network traffic between primary and secondary DNS servers and the subsequent computation time in incorporating an entirely new zone. The protocol ensures that incremental zone transfers can be sent to indicate changes from both dynamic updates and from zones changed on disk (those reincorporated through a refresh signal or server restart).

• Notify

  Implementing the RFC 1996 Notify process. This is a method by which the primary DNS server can indicate to its secondary nameservers that zone data has been updated. This decreases the time periods in which a secondary DNS server will have data out of synchronization with its primary DNS server.

• File conversion utility

  Extending the configuration file conversion utility to support IBM functional additions to previous BIND releases. This involves mapping the dynamic keywords of previous named.boot files to a functional equivalent in the named.conf configuration file.

• Proprietary protocol for secure updates of dynamic notify dynamic zones.

  Bind 4.9.3 is available using named4. Bind 8.1.2 uses named8.

8.11 PMTU Active by Default (4.3.3)

When one IP host has a large amount of data to send to another host, the data is transmitted as a series of IP datagrams. It is preferable that these datagrams be of the largest size that does not require fragmentation anywhere along the path from the source to the destination. This datagram size is referred to as the Path MTU (PMTU), and it is equal to the minimum of the MTU of each hop in the path.

RFC1191 provides the mechanism to discover the best PMTU and is supported by AIX, but before version 4.3.3 if was not active by default.

The routing table grows as new PMTU are discovered, with very little performance impact on the system. The new route expires after a while of inactivity.

Three parameters of no command have changed default value to make PMTU active by default. Their new value is:

```
udp_pmtu_discover = 1
tcp_pmtu_discover = 1
route_expire = 1
```

The first two activate PMTU either for UDP and TCP packets, while the third one makes routes expire.

8.12 Interface Specific Network Options (4.3.3)

AIX implements a set of system-wide network parameters and some of them can impact the performance of TCP/IP. AIX supports a wide range of physical media on a single system (SLIP, PPP, ISDN, 10 Mbps Ethernet, Gigabit Ethernet, and others) and using the same network parameter for each interface may not be optimal since some of them may have unique characteristics that require special values.

In AIX 4.3.3, the concept of interface specific network options (ISNO) has been introduced. The user has the ability to activate or deactivate ISNO using the `no` command. A new global option *use_isno* is available and its default value is 1 (enabled). The intended use of this option is for service related issues, to allow for a diagnostic tool to eliminate potential tuning errors.

This design allows to define adapter specific defaults that are shipped in the predefined attribute ODM database PdAt. The global network options are still maintained using the `no` command, but the value of the ISNO takes precedence over the global values, assuming that the TCP/IP traffic flows over an interface that has ISNO set.

The ISNO values are persistent across system boots when placed in the customized attribute ODM database CuAt. The interface configuration method (`/usr/lib/methods/cfgif`) retrieves of ISNO values in CuAt or PdAt and passes them to the `ifconfig` command.

The `ifconfig` command has been changed to modify the following ISNO values:

- tcp_recvspace
- tcp_sendspace
- tcp_mssdflt
- tcp_nodelay

- rfc1323

Suppose, for example, that you want to activate the rfc1323 network option only for the token ring adapter tr0. You can check that use_isno is enable and that the global rfc1323 value is 0 executing:

```
# no -o use_isno
use_isno = 1
#no -o rfc1323
rfc1323 = 0
```

Then you can show the current configuration of the interface and then change the rfc1323 value using the `ifconfig` command.

```
# ifconfig tr0
tr0: flags=e0a0043<UP,BROADCAST,RUNNING,ALLCAST,MULTICAST,GROUPRT,64BIT>
        inet 9.3.240.40 netmask 0xffffff80 broadcast 9.3.240.127
# ifconfig tr0 rfc1323 1
```

Issuing again the ifconfig tr0 command you can see that the interface has been changed according your actions:

```
# ifconfig tr0
tr0: flags=e0a0043<UP,BROADCAST,RUNNING,ALLCAST,MULTICAST,GROUPRT,64BIT>
        inet 9.3.240.40 netmask 0xffffff80 broadcast 9.3.240.127
         rfc1323 1
```

In order to delete the interface specific network option you can use the following syntax:

```
# ifconfig tr0 -rfc1323
# ifconfig tr0
tr0: flags=e0a0043<UP,BROADCAST,RUNNING,ALLCAST,MULTICAST,GROUPRT,64BIT>
        inet 9.3.240.40 netmask 0xffffff80 broadcast 9.3.240.127
```

Disabling the use_isno parameter deactivates all values defined on each interface, applying the global definitions. No local values are then allowed and `ifconfig` fails if applied, as shown in the following example:

```
# no -o use_isno=0
# ifconfig tr0 rfc1323 1
ioctl (set interface options): The socket does not allow the requested operation.
```

8.13 Inheriting TCP Socket Options (4.3.3)

Before AIX 4.3.3, TCP options had to be reset after each new connection was made after calling accept(). This requires more setsockopt() calls to set options and the instruction path length was longer.

For example, Web servers that wanted to turn on TCP_NODELAY had to issue a separate system call for each new connection. The setsockopt() call costs more than 1200 instructions on AIX 4.3.2.

In AIX 4.3.3, sockets created by use of the accept() system call inherit TCP socket options from the listening socket. These options are:

- TCP_NODELAY
- TCP_STDURG
- TCP_RFC1323

TCP_RFC1323 was already inherited from the listening socket in previous versions.

8.14 Web Server Performance Improvements

Web server workload has a large number of small file operations. In smaller file operations, the cost of processing connection set-up/tear-down far exceeds the data touching (such as copy or checksum) operations. By reducing the number of packets that are sent or received for connection set-up/tear down, performance is improved. IBM has developed a set of TCP changes to reduce the number of packets from 9 to 7 or 6.

For Web transaction oriented environments using HTTP protocols, AIX Version 4.3.2 reduces the number of network packets that are exchanged between the workstation and the server. This lowers the CPU packet processing overhead, increasing server performance and capacity.

Additional Web server throughput is achieved in AIX 4.3.3 introducing a new kernel extension that is able to satisfy HTTP GET requests on behalf of the Web server looking into Network Buffer Cache objects. No data is copied between kernel and user space and no user context switch is necessary. Tests made with this technology have doubled the SPECweb96 throughput.

8.14.1 Reducing the Number of TCP Packages (4.3.2)

There are some modifications to the TCP protocol implementation in AIX 4.3.2 that improves the performances of HTTP-like transactions on Web servers and clients.

A performance improvement for Web servers and clients can be achieved by reducing the total number of TCP packets that are exchanged, without violating the TCP protocol. For Web servers and clients, this can be done by delaying certain ACK messages and piggybacking them with the next packet that will be sent and also by sending the FIN message along with the last data packet.

A typical HTTP transaction requires the exchange of nine TCP packets:

1. The client sends a SYN TCP packet to connect to the server.
2. The server acknowledges the client's SYN and sends a SYN to accept the connection.
3. The client acknowledges the server's SYN.
4. The client sends its HTTP request in a data packet.
5. The server sends a data packet containing the acknowledgment and the answer to the client's request.
6. The server shuts down its side of the connection by sending a FIN packet.
7. The client acknowledges the server's FIN.
8. The client shuts down its side of the connection by sending a FIN packet.
9. The server acknowledges the client's FIN.

In AIX 4.3.2, the TCP exchange packets are reduced from nine to seven by using `no` command and to six by using send_file() system call with the SF_CLOSE flag.

8.14.1.1 Commands Affected

Reducing the TCP package number can be set by the user from the command line or by an application on a per process basis.

Two new options, delayack and delayackports, have been added into the `no` command.

delayack

The delayack options delays ACKs for certain TCP packets and attempts to piggyback them with the next packet sent instead. It enables the user to specify whether the delay will be performed for the SYN ACKs, the FIN ACKs, or both. This will only be performed for connections whose destination port is specified in the list of the delayackports attribute.

The delayack option may have one of the following four values:

0	No delay, the default value
1	Delay the ACK for the SYN only
2	Delay the ACK for the FIN only

3 Delay the ACK for both

Setting delayack to any other value will result in a failure with EINVAL error code. The following are some examples of using delayack:

```
# no -o delayack=3
```

sets delay the ACK for both SYN and FIN.

```
# no -d delayack
```

sets delayack to the default value 0.

delayackports

The delayackports option allows a user to specify a list of up to ten ports for which the ACKs for SYNs or FINs defined by the delayack port option will be performed.

This allows a system administrator to configure this operation for specific applications that use a specific port. Although this will improve the performance for Web servers, this may not be good for other applications.

The delayackports option takes a list of ports numbers separated by commas and enclosed in curly braces.

For example: `no -o delayackports={80,1080,1180}`

An empty list, `no -o delayackports={}`, will clear the ports.

`# no -d delayackports`, sets delayackports to the default. Braces {} indicate no ports.

The send_file() system call with the SF_CLOSE flag will send the FIN packet with the last data.

8.14.1.2 Reducing the Contention of INIFADDR and Route Lock

There are some changes in kernel to the route lock and inifaddr lock in order to improve performance. These changes decrease the processing time for each incoming and outgoing packet. These changes are internal and do not affect external user interfaces.

The route lock and inifaddr are used by the TCP/IP protocol packets:

INIFADDR lock	examined by every incoming packet
Route lock	examined by every outgoing packet

Reducing the Contention of INIFADDR Lock

Currently, for every incoming packet, the inifaddr list must be searched with the lock held. This causes lock contention.

The INIFADDR lock contention is critical for SMP machines. This lock is used for all input packets, causing unacceptable lock contention on machines with multiple adapters. Therefore, it is an important lock on 12-way RS/6000 S70 machine with multiple adapters.

Since this inifaddr list is changed very rarely, but searched so often, the lock contention is reduced by:

- Allowing concurrent searches
- New macros to allow for simple read-write locking, and the INIFADDR lock macros are changed to use them.

Reducing the Contention of Route Lock

Currently, all outgoing IP packets must take the single route lock. On an SMP machine with multiple adapters, a large number of packets must take this lock, so it is the hot (critical) lock on output. Reducing the contention on this lock can increase the throughput on SMP systems with multiple adapters.

There is a global route lock that must be taken any time any route is changed or used. Each connection keeps a pointer to a cached route, but it still must take the route lock to check the validity of its cached route. So, the route lock must be taken for every outgoing packet.

The new design eliminates the requirement to take the global route lock for outgoing packets in the case where the cached route is valid. This should improve performance for TCP, where the cached route will usually be valid so the global lock will not be needed on packet output. For UDP, the cached route often is not valid, and so the global lock will still be needed for output. But since the contention on this lock from other code will be reduced, there is also a performance improvement for UDP.

The global route lock that protects the structure of the routing table has also been changed to use the same read/write lock in places where the global lock is still needed.

8.14.2 HTTP GET Kernel Extension (4.3.3)

A new AIX loadable kernel HTTP GET extension has been added to AIX 4.3.3 to improve Web response time. It is based on technology developed in IBM Watson Research and IBM Raleigh originally known as AFPA (Adaptive Fast

Path Architecture) and renamed FRCA (Fast Response Cache Architecture). Tests made with this technology have doubled the SPECweb96 throughput.

HTTP GET requests are intercepted and the response is sent directly from the AIX Network Buffer Cache (NBC) on the input interrupt. No data is copied between kernel and user space and no user context switch is necessary. If the HTTP GET request can be serviced by the engine, the user space Web server is not contacted, and never sees the request. GET requests which cannot be serviced by the kernel engine are passed up to the user space Web server.

The Web client is not aware of the difference between Web server and FRCA responses. At the moment, only HTTP 1.0 protocol is supported. If the Web client uses HTTP 1.1 protocol, FRCA does not provide cached data and the Web server is always contacted.

The Web server can add objects to the NBC in order to make them available to the kernel extension using a specific API. Web servers that are not modified to use the API can function normally, even if another Web server is using FRCA on the same machine.

For this initial release, the API is not published. The only Web server which currently uses the FRCA API is the IBM HTTP Server, based on Apache. The API will be published in a later release.

For the IBM HTTP Server, the system administrator must only insure that the FRCA kernel extension is loaded, before starting the Web server, and that FRCA is enabled in the Apache config files.

FRCA may be manually configured to operate with other Web servers. The system administrator must use the `frcactrl` command to activate the kernel extension for the specific Web server, and manually load the file cache. See 8.14.2.2, “frcactrl Command” on page 425 for more information. Note that this function is not supported in the initial AIX 4.3.3 release. It will be shipped in one of the first PTFs.

The basic data flow for this model is provided in Figure 77.

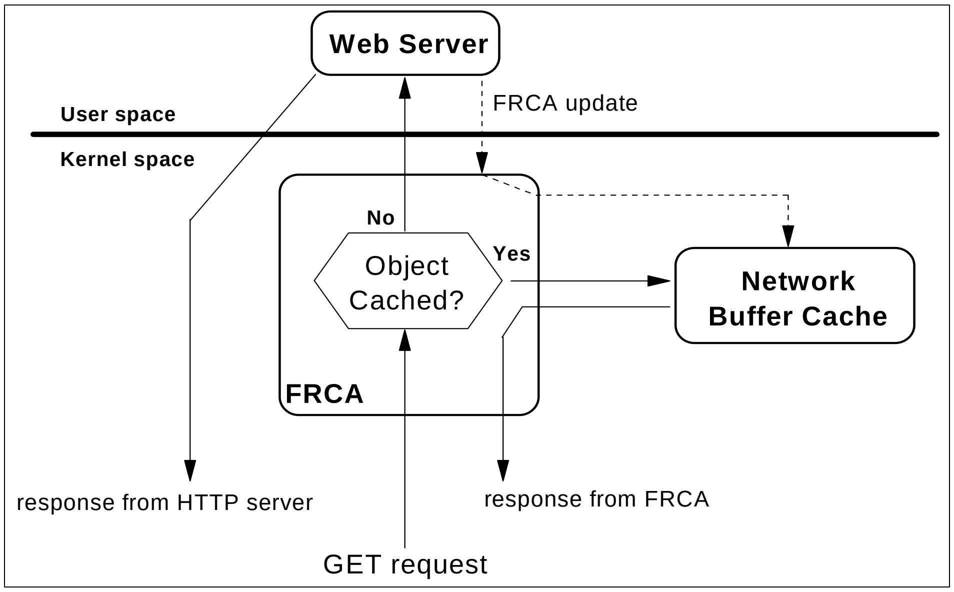


Figure 77. FRCA GET Data Flow

A time stamp is kept on each object. When an object is requested on the interrupt and its time stamp is older than a given time period, the request is passed up to user space as though the item was not in the cache. This insures that objects are periodically refreshed. The system administrator may set this time period with the `frcactrl revaltimeout` subcommand. The default is 10 seconds. A Web site whose static content seldom changes may choose to set this timer to a week or longer. Note that pages that are manually loaded with the `frcactrl` command cannot be aged, as there is no way to automatically reload them. If an administrator uses manually loaded pages, it is his responsibility to reload the pages if they have been updated.

FRCA includes a logging function. Since no disk I/O may be done on the interrupt, the log information is stored on per-CPU queues. A kernel thread is started to write log data in a file. For the IBM HTTP Server, the location of the log file is specified in the Apache config file. For other Web servers this must be done with the `frcactrl` command. The first log file’s name is the name specified concatenated with ".0". Log files are written up to gigabytes in size. After that a new file is opened with an incremented suffix. It is the responsibility of the system administrator to periodically erase or archive old log files.

Servicing the HTTP requests directly on the interrupt greatly increases the performance of Web servers. However, it should be noted that these gains can only be achieved when the system has enough physical memory to cache a significant part of the Web traffic footprint. This feature should not be used on machines without sufficient physical memory. It is also recommended that the nbc_limit option of the `no` command should not be set to higher than half the value of the wall.

Since all the processing for a cache hit occurs on the interrupt, there is a danger that the system could spend too much time servicing interrupts. This could happen if the network load was enough to saturate the CPUs of the machine. This would mean that important user level processes would be starved of CPU time. A typical symptom of this problem, is that the console appears to freeze up. To prevent this from occurring, FRCA has a configurable parameter to specify how much of the time that the requests are allowed to be serviced on the interrupt. If a request comes out of this percentage limit, it will automatically be sent to user space, thus allowing the CPU to be shared among the user level Web server and other user processes. The default for this value is 75% of the requests will be allowed on interrupt. If a system does not have a heavy network load, this value can be set to 100%, with the `frcactrl pctonintr` subcommand, which will enable optimal processing.

The FRCA cache supports the Virtual Host capability of Apache. This allows a Web server to service requests for different host names in the HTTP request protocol over one IP address. This means that the cache must also have a name space scoping capability, which adds a little overhead. An administrator may choose to disable this function with the `frcactrl nsisupport` subcommand. However, this is not recommended.

As an example, we configured an IBM HTTP Server (version 1.3.6 build July 28 1999) and enabled the FRCA component using the provided sample httpd.conf.sample.afpa configuration file. We then used a browser to ask repeatedly the same files. Using `netstat -c` and `frcactrl stats` commands we could see that FRCA did provide most of the pages on behalf of the Web server:

```
# frcactrl stats
      Total    Deferred       Cache       Cache    Resource
   Requests    Requests        Hits      Misses      Errors
-----------------------------------------------------------
         94           0          80          14           0
#
# netstat -c
Network Buffer Cache Statistics:
-------------------------------
Current total cache buffer size: 325312
Maximum total cache buffer size: 325312
```

```
Current total cache data size: 207450
Maximum total cache data size: 207450
Current number of cache: 25
Maximum number of cache: 25
Number of cache with data: 25
Number of searches in cache: 2063
Number of cache hit: 1044
Number of cache miss: 81
Number of cache newly added: 25
Number of cache updated: 0
Number of cache removed: 0
Number of successful cache accesses: 1069
Number of unsuccessful cache accesses: 56
Number of cache validation: 0
Current total cache data size in private segments: 0
Maximum total cache data size in private segments: 0
Current total number of private segments: 0
Maximum total number of private segments: 0
Current number of free private segments: 0
Current total NBC_NAMED_FILE entries: 49
Maximum total NBC_NAMED_FILE entries: 49
```

Cache misses were mainly caused by dynamic data generated by the Web server, thus not cached into Network Buffer Cache by the server.

8.14.2.1 IBM HTTP Server Config File Directives

The following directives can be used to control FRCA in the IBM HTTP Server config file /etc/httpd.conf:

• AfpaEnable	Enable FRCA for this Web server
• AfpaCache on	Enable Web server file caching
• AfpaLogFile /afpa-log V-ECLF	Location and format of log file
• AfpaMinCache 0	The minimum file size to cache
• AfpaMaxCache 100000	The maximum file size to cache
• AfpaLogging on	Enable logging
• AfpaBindLogger -1	CPU the logger thread is bound to (-1=none)

8.14.2.2 frcactrl Command

The `frcactrl` command controls and configures the FRCA kernel extension. The kernel extension must be loaded before starting any Web servers that want to use FRCA.

For the IBM HTTP Server, no configuration commands are required since configuration data is stored in the IBM HTTP Web Server configuration files and automatically passed to FRCA through an API. The IBM HTTP Web Server automatically loads files into the Network Buffer Cache.

Other Web servers may gain a performance advantage from the FRCA without knowing the FRCA APIs but using the `frcactrl` command. The command is needed to make the kernel extension listen to the socket opened by the Web server and to load the required files into the cache. If the server is restarted, it is necessary to reissue the command.

The command syntax is:

```
Usage: frcactrl {load|unload}
       frcactrl open Ip_Address Port [Virtual_Host]
                                Server_Name Virtual_Root Log_File [bin]
       frcactrl close Ip_Address Port [Virtual_Host]
       frcactrl loadfile Ip_Address Port [Virtual_Host] Document_Root File
       frcactrl stats [reset] [interval]
       frcactrl logging {on|off} [CPU_Id]
       frcactrl logfmt File
       frcactrl {start|stop} Ip_Address Port [Virtual_Host]
       frcactrl revaltimeout <sec>
       frcactrl pctonintr <% on interrupt>
       frcactrl nsisupport {on|off}
```

where:

- Ip_Address is the IP address the Web server is listening to. If the server listens to any IP address and not to a specific one, use the 0.0.0.0 address. You can check the IP address the server is bound using the `netstat -a` command.
- Port is the port number the server listens to (normally port 80)
- Virtual_Host represents the virtual name of the Web server. It is used when a server listens to different IP addresses or virtual host names and replies differently to each of them. See the IBM HTTP Server documentation for a description of the different types of virtual hosting.
- Server_Name is the name of the Web server. This will be included in the HTTP response header.
- Virtual_Root is the root of the file tree that contains the information provided by the server.
- Log_File is the file where FRCA logs all the requests satisfied by FRCA and not passed to the Web server. It is by default an ASCII file, but it is possible to create a binary file if the bin option is provided.

The `load` and `unload` subcommands are used to load and unload the FRCA kernel extension. They are the only commands you need to execute if you use the IBM HTTP Server. The Web server must be started after the load of

FRCA; if you unload the kernel extension, you must stop the Web server, redo the load and then start the Web server in order to set FRCA.

The other subcommands are basically required for the Web servers that do not use the FRCA API. They need to use the `open` subcommand to make FRCA bind at the server's socket and the `start` subcommand to make FRCA start listening to HTTP request and look into the cache.

When the Web server is not able to add data to the Network Buffer Cache using API, Web data must be selected and inserted into the cache using the `loadfile` subcommand, providing all the required information needed by FRCA to associate the client's request with the cache entry. The IP address provided *must* always be a valid network address on the machine, never use 0.0.0.0. This is because of security issues when using virtual hosts. FRCA permits multiple loads of a file with different IP server addresses. Only one copy of the file will be present in memory, but there will be multiple keys to access it.

If you want to stop using the FRCA, you may issue the `stop` and `close` subcommands to clean the socket configuration as it was before FRCA was activated.

Logging activity may be started or stopped using the `logging` subcommand. If you have selected a binary log format, you can convert an existing log from binary to ASCII. The Cpu_Id is used to bind the logging thread to a specific CPU on multiprocessor systems. On a highly tuned system, an administrator may choose to direct network interrupts to specific CPUs, which will handle all Web requests. The administrator can then specify that the logging thread runs on a CPU that is not servicing Web requests, and is thus guaranteed that the logging thread will get enough CPU time.

The `stats` subcommand is useful to monitor global FRCA activity. It allows you to see how many requests have been made to FRCA and how many have been satisfied by FRCA or passed to the Web server. You can have additional details on Network Buffer Cache usage issuing the `netstat -c` command.

The `revaltimeout` subcommand is used to set the system wide revaluating timeout for objects. The default is 10 seconds. If content is not constantly changing on the server, this should be set to a very large value for optimal performance.

The `pctonintr` subcommand is used to set what percentage of requests will be serviced on interrupts. If there is no danger of saturation from a network adapter, this value can be set to 100% for optimal performance.

The `nsisupport` subcommand can be used to disable the name space scope used for virtual hosting. If virtual hosting is not being used on a system this may be disabled, however the performance increase is not that great, and this is not recommended.

Example of using FRCA with a non-FRCA enabled Web server:

```
# frcactrl load
# apachectl start              NOTE: we assume a non-FRCA Apache
# frcactrl open 9.3.240.42 80 apache /usr/apache/htdocs /logs/frca.log
# frcactrl start 9.3.240.42 80
# frcactrl loadfile 9.3.240.42 80 /usr/apache/htdocs index.html
```

A user may also specify all files in a directory with the * wildcard (an asterisk).

8.15 TCP Checksum Offload on ATM 155 Mbps PCI Adapter (4.3.2)

The PCI ATM 155 adapter has a hardware TCP/UDP checksum capability. In AIX 4.3.2, the ATM network device driver is modified to use this feature.

In the new ATM 155 adapter device driver, the workload of TCP data checksum processing is offloaded from the AIX TCP/IP protocol stack to the adapter itself. A related enhancement automatically remembers the mapping of virtual addresses to physical addresses for the entire networking buffer pool to save address translation during networking I/O operations.

TCP checksum offload for ATM reduces the amount of time spent on the main CPU computing checksums, thereby reducing latency and allowing the system to handle more work, in particular, to handle more packets in the same amount of time. This results in performance improvements.

The SpecWeb96 benchmark is published on the 7017-S70 using ATM interfaces. It benefits from this checksum offload capability.

TCP and the other network layers continue to perform as they do today, particularly on adapters that do not support checksum offload. All TCP functions are the same.

In order to improve performance for TCP over ATM, small packets do not have their checksum processing offloaded since setting up the offload will probably use as much processor time as computing the checksum. There is no gain, and may actually be a loss, in offloading checksum processing for these small packets.

There is a packet size threshold where it is cheaper to do the checksum processing completely on the CPU. The driver only offloads checksums for packets larger than this threshold. This threshold is maintained internally as a variable or constant.

Currently, AIX checksums transmitted packets before copying them to the network adapter, and checksum received packets after they have been copied from the adapter to processor memory. Checksumming packets on the adapter increases the possibility of undetected data errors during the copy from or to the processor. This possibility is very small. The NDD provides a way for you to turn off the checksum offload capability for an adapter in cases where errors are believed to be occurring.

8.15.1 Limitations

The feature of checksum offload only applies to TCP on ATM over IP Version 4. It means:

- This driver does not offload checksum processing for IPv6 TCP connections,
- This driver does not offload checksum processing for UDP datagrams.
- The checksum processing is not allowed when IPSEC is running since IPSEC may encrypt data, and the checksum for the data should be done before it is encrypted. Offloading, in this case, will cause the checksum to be calculated after the data has been encrypted, which is wrong. When IPSEC is running, the checksum must be calculated before the data is encrypted.

8.15.2 Command Changes

The command `ifconfig` is changed to handle a new interface option, checksum_offload. Table 41 lists the new options.

Table 41. Ifconfig New Options for Checksum Offload

checksum_offload	Enables checksum offload if the adapter associated with the interface supports it. This is the default for adapters that support checksum offload.
-checksum_offload	Disables checksum offload. This is the default for adapters that do not support checksum offload.

8.16 TCP Checksum Offload on Gigabit Ethernet Adapter (4.3.3)

The Gigabit Ethernet adapter has a hardware TCP checksum capability. In AIX 4.3.3 the Ethernet network device driver has been modified to use such feature.

Taking advantage of adapter’s TCP checksum capability increases performance because the system CPU no longer has to calculate the checksum for every TCP segment. With the high packet rate that can be reached by Gigabit Ethernet, the system gains a great reduction in the amount of CPU time spent per packet, and is able to handle more packets per second.

Using the Network Buffer Cache to cache data sent with send_file() or with the in-kernel HTTP get engine, it is possible to achieve zero data touching by off loading TCP checksum calculations to the adapter. This avoids loading the data into the level 1 and level 2 caches and then flushing the data from those caches prior to DMA.

A new ODM attribute is provided in AIX Version 4.3.3 to enable HW checksum of received TCP packets: rx_checksum. It is a yes/no value (default yes) specifying if the device driver should tell the adapter to do receive check summing for TCP packets. Transmit check summing is done if the transmit checksum flag is set in the mbuf. The adapter does not handle IP checksums since it does not save any time.

A user can disable or enable checksum off load on a per interface basis using `ifconfig`:

```
ifconfig <interface> [ checksum_offload | -checksum_offload ]
```

The checksum_offload parameters activates the adapter’s checksum capability while the -checksum_offload parameter makes AIX compute checksum on TCP packets.

Checksum offload is made only for TCP packets using IPv4 and it is disabled when IPSEC is active since TCP packets may be encrypted and checksum must be made before encryption.

8.17 Thread-Based Application Connection Enhancement (4.3.2)

The thundering herd problem is a well-known, industry-wide performance problem that effects some Web servers. When multiple threads call accept() function on the same socket, all of these threads are woken up for a single new connection. For example, if there are 256 threads sleeping in accept(), only one thread needs to handle the new connection. However, currently all 256 threads are woken up. Since only one thread receives the connection, the other 255 immediately go back to sleep. This results in wasted CPU time due to the unnecessary scheduling, running, and sleeping of the 255 threads.

In AIX Version 4.3.2, only a single thread is awakened, thus reducing CPU overhead. And the system now is able to handle higher loads of socket connections. Industry multithreaded network server application developers will appreciate that AIX Version 4.3.2 overcomes the thundering herd problem of wasted CPU cycles resulting from waking up multiple threads of a network server when new connections arrive.

8.18 Kernel Enhancement for High Speed Network Adapters (4.3.3)

The incoming IP packets are normally processed on an interrupt level. For an inbound TCP segment, the device driver code, the demuxer processing, the IP layer processing, the TCP layer processing, the append to the socket buffer, and the notification of the application is done in a single call at interrupt level through the entire protocol stack. This is the code path with the minimal length and therefore the most efficient approach.

With high speed network adapters like the Gigabit Ethernet it is possible that a new packet is always available from the adapter, and the device driver (SLIH, second level interrupt handler) repeatedly calls handling routines without returning to the first level interrupt handler (FLIH). As a consequence, the CPU that received the interrupt remains on interrupt level as long as the SLIH finds a new packet pending when the previous one has been handled. The interrupted process or a lower priority interrupt handler as well as any user or kernel level processes bound to this CPU will not be run until packets stop arriving and the device driver returns to the FLIH.

On uni-processor machines there is no other solution to this problem than using a CPU that is fast enough to cope with the bandwidth of such adapter. For multiprocessor machines, a new packet handling model has been introduced in AIX 4.3.3 that provides better network throughput for high speed network adapters when receiving high volume data streams. This is achieved by splitting up the current receive code path at the transition point from the demuxer to the IP layer.

The SLIH processes an incoming packet only up to this transition point, queues the packet, and then returns to process the next packet or returns from interrupt. The time spent on interrupt is decreased improving the throughput of the adapter. For each CPU there is a queue and a dedicated kernel thread that completes packet handling.

New options *thread* and *-thread* have been added to the `ifconfig` command to allow thread mode processing on or off at runtime. Since the changing of the processing mode implies replacing the protocol filters registered with the respective demuxer, these flags can only be changed if the interface is in detached state.

For example, in order to switch the en0 interface with IP address 10.0.0.1 and netmask 255.255.255.0 from interrupt to thread mode, you have to execute the following commands:

```
# ifconfig en0 detach
# ifconfig en0 thread
# ifconfig en0 10.0.0.1 netmask 255.255.255.0
```

The use of detach option causes the interface to go down and to loose its IP configuration, so you have to reconfigure the IP address and netmask. If any TCP/IP route, including the default one, was using the interface, it has been deleted from the kernel and it must be added again with the `route add` command.

The new thread model activation is shown in the output of the `ifconfig` command by the THREAD flag:

```
# ifconfig en0
en0: flags=e080863,20<UP,BROADCAST,NOTRAILERS,RUNNING,SIMPLEX,MULTICAST,
GROUPRT,64BIT,THREAD>
        inet 10.0.0.1 netmask 0xffffff00 broadcast 10.0.0.255
```

As the first interface activates the thread model, one packet queue and one kernel thread are created for each CPU in the system.

8.19 IBM 10/100 Mbps PCI Ethernet Adapter Device Driver (4.3.2)

In previous AIX levels, the Ethernet device driver copies data from mbufs to its own buffers. Recent prototypes have shown that transmitting data directly from mbufs would lead to better performance.

AIX 4.3.2 provides a new device driver for IBM 10/100 Mbps PCI Ethernet adapter. The new device driver reduces data copy operations from network buffers to its private buffers. The network performance in AIX 4.3.2 benefits from enhancements in this new PCI network device drivers. These enhancements reduce CPU demand, thus increasing overall server capacity.

The device driver supports the PCI 10/100 Ethernet adapter for AIX Versions 4.2.1 and 4.3.0. The new PCI 10/100 Ethernet is modified to support AUI and BNC ports starting at AIX 4.3.2 and is backwards compatible. The performance enhancements for reducing the data copies in AIX 4.3.2 is backwards compatible to AIX 4.2.

The device driver is conform to the Common Data Link Interface (CDLI) interface specifications.

This adapter now supports the twisted pair, AUI and BNC ports.

The speeds supported are:

- 10 (10 Mbps, half-duplex)
- 20 (10 Mbps, full-duplex)
- 100 (100Mbps, half-duplex)
- 200 (100Mbps, full-duplex)
- Auto-negotiate.

8.19.1 Packaging

The PCI 10/100 Ethernet adapter software package includes:

- devices.pci.23100020.rte
- devices.pci.23100020.diag

8.19.2 Configuration Parameters

The following parameters are user configurable:

Transmit Queue Size (tx_que_size)

The device driver supports a user-configurable transmit queue. This is the queue the adapter uses (not an extension of the adapter's queue). It is configurable among the values of 16, 32, 64, 128, and 256, with a default of 256.

Because of the configurable size of the adapter's hardware queue, the driver does not support a software queue.

Receive Queue Size (rx_que_size)

The device driver supports a user-configurable receive queue. This is the queue the adapter uses (not an extension of the adapter's queue). It is configurable among the values of 16, 32, 64, 128, and 256, with a default of 256.

Receive Buffer Pool Size (rxbuf_pool_size)

The device driver supports a user-configurable receive buffer pool size. The buffer is the number of pre-allocated mbufs for receiving packets. The minimum size of the buffer is the receive queue size, and the maximum is 2 KB, with a default value of 384.

Media Speed (media_speed)

The device driver supports speeds of 10 (10 Mbps, half-duplex), 20 (10 Mbps, full-duplex), 100 (100 Mbps, half-duplex), 200 (100 Mbps, full-duplex), and auto-negotiate on twisted pair. On the AUI port the device driver will support speeds of 10 (10 Mbps, half-duplex) and 20 (10 Mbps, full-duplex). The BNC port will only support 10 (10 Mbps, half-duplex). This attribute is user-configurable, with a default of auto-negotiate on twisted pair.

Enable Alternate Address (use_alt_addr)

The device driver supports a configuration option to turn on and off use of an alternate network address. The values are yes and no, with a default of no. When this value is set to yes, the alt_addr parameter will define the address.

Alternate Network Address (alt_addr)

For the network address, the device driver accepts the adapter’s hardware address or a configured alternate network address. When the use_alt_addr configuration option is set to yes, this alternate address is used. Any valid individual address can be used, but a multicast address cannot be defined as a network address.

Inter-Packet Gap (ip_gap)

The inter-packet gap bit rate setting controls the aggressiveness of the adapter on the network. A smaller number will increase the aggressiveness of the adapter, while a larger number will decrease the aggressiveness (and increase the fairness) of the adapter. If the adapter statistics show a large number of collisions and deferrals, this number should be increased. Valid values range from 96 to 252, in increments of 4. The default value of 96 results in IPG of 9.6 microseconds for 10 Mb and 0.96 microseconds for 100 Mb media speeds. Each unit of bit rate introduces an IPG of 100 ns at 10 Mb and 10 ns at 100 Mb media speed.

8.19.3 Trace

The device driver sets up three trace hook IDs (defined in /usr/include/sys/cdli_entuser.phxent.h). The first hook traces the transmit path, the second trace the receive path, and the third will trace everything else.

The device driver has three trace points: transmit, receive, and other, and has a trace table of at least 1000 entries when complied normally. It traces entry/exit from transmit/receive routines at the interrupt and user layers, as well as on entry to the interrupt handler when compiled normally.

Table 42 lists these hook IDs.

Table 42. Hook IDs of 10/100 Ethernet PCI Adapter

Trace	**Trace hook ID**
Transmit	0x2E6
Receive	0x2E7
Other	0x2e8

8.19.4 Error Logging

The device driver logs errors in the error log whenever one of the following errors occur: hardware failures, DMA operations, and memory faults.

Following are the error log entries returned by the 10/100 Mbps PCI Ethernet device driver:

ERRID_PHXENT_ADAP_ERR

Indicates that the adapter is not responding to initialization commands. User-intervention is necessary to fix the problem.

ERRID_PHXENT_ERR_RCVRY

Indicates that the adapter received a temporary error requiring that it enter network recovery mode. It has reset the adapter in an attempt to fix the problem.

ERRID_PHXENT_TX_ERR

Indicates that the device driver has detected a transmission error. User-intervention is not required unless the problem persists.

ERRID_PHXENT_PIO

Indicates the device driver has detected a program I/O error. The driver was unable to fix the problem. User-intervention is required.

ERRID_PHXENT_DOWN

Indicates that the device has been shutdown due to an irrecoverable error. The device is no longer functional due to the error. The irrecoverable error that caused the device to be shutdown is logged immediately before this error log entry. User-intervention is required to bring the device back online.

ERRID_PHXENT_EEPROM_ERR

Indicates that the device driver has detected an error in the EEPROM of the device. The driver will not become available. Hardware support should be contacted.

8.20 SDLC/BSC Support for 4-Port PCI Adapter (4.3.2)

This enhancement enables the existing AIX SLDC and BSC (Binary Synchronous Communications) protocols to run on the 4-port PCI adapter IBM ARTIC960Hx. It provides a high performance multiprotocol WAN connectivity on PCI based machines. Both SDLC and Bisync can run on the same adapter on a per port basis.

BSC is required for banks that have Automated Teller Machines (ATM) and want to upgrade to latest RS/6000 models. Newer systems like the 7017-S70 no longer have ISA slots.

The 4-port PCI Adapter has up to four leased lines connectivity. Each port can communicate at 2.0 Mbps. It replaces the ISA adapter (#2701). The adapter is implemented for use in systems that support 64 bit wide PCI local bus operating at 33 or 66 MHz.

4-Port Multi-Interface PMC is the daughter card connected to a 4-port PCI adapter and provides the hardware to support up to four network links. For each port of the adapter, the physical interface of the attached cable is auto selected. There are four connection ports on the adapter. Each function as individual network devices simultaneously. The user can configure which physical interface (V.24, V.35, V.36, or X.21) that should be used per port or the adapter can autoselect the physical interface based on the cable connected.

Installation of AIX using this adapter is also be supported. The code to support this is part of the system microcode. The AIX product device driver will run on the adapter after control has been passed to AIX.

8.20.1 Packaging

SDLC and Bisync are both shipped with base AIX.

Following PTFs are needed for using the SDLC and Bisync protocol:

- AIX 4.3.2 with IX81860
- AIX 4.2.1 with IX81861

8.20.2 Trace

The device driver will issue trace points for the following conditions:

- All error conditions
- At the beginning and end of each main function in the main path
- At the beginning and end of each function that is tracking buffers outside of the main path
- Debugging purposes. These trace points are only enabled when the driver is compiled with -DDEBUG turned on, and therefore, the driver can contain as many of these trace points as desired.

The main goals for having trace points in a CDLI device driver are to be able to monitor drivers for errors and to track packets as they move through the driver. To this end, it is important that trace points be placed at the beginning and end of each main routine that does processing on packets. Also, it is important to try and trace the flow of buffers (for example, mbufs) as they flow through the system. There should also be trace points placed at each point where an error could occur.

The device driver also has trace points to support the netpmon program (component cmdperft). There are generally five trace points:

WQUE	An output packet has been queued for transmission.
WEND	The output of a packet is complete.
RDAT	An input packet has been received by the device driver.
RNOT	An input packet has been given to the demuxer.
REND	The demuxer has returned.

All CDLI drivers must register for a trace hook IDs.

Note: CDLI device drivers may register for more than one trace hook ID. In the case of multiple trace hook IDs, one could be used for transmit, one for receive, and another for errors.

8.20.3 Error Logging

The device driver will do error logging when the following situations occur:

- An error on PIO operations
- A hardware failure on the device
- An error on DMA operations
- Network errors
- Other device specific errors

8.21 Open Network Computing (ONC+)

The ONC+ technology has been licensed from SunSoft and is being included within AIX to meet customer requirements. This technology contains many different functional components; the main ones being: NFS Version 3, NIS+, CacheFS, TIRPC, and AutoFS. Not all of these components are included with this release of AIX. NFS V3 was introduced in AIX Version 4.2.1. CacheFS was introduced in AIX Version 4.3.0, AutoFS was included in AIX 4.3.1 and NIS+ was introduced in AIX 4.3.3.

8.21.1 CacheFS

CacheFS is a local disk cache mechanism for NFS clients. It provides the ability for an NFS client to cache file system data on its local disk, thereby avoiding use of the network and NFS server when the data is accessed and is not in physical memory. This improves NFS server performance and scalability by reducing server and network load. Designed as a layered file system, CacheFS provides the ability to cache one file system on another. In an NFS environment, CacheFS increases the clients per server ratio, reduces server and network loads, and improves performance for clients particularly on slow links.

CacheFS is contained in the bos.net.cachefs fileset, which is not automatically installed when installing AIX.

8.21.1.1 How CacheFS Works

After creating a CacheFS file system on a client system, the system administrator specifies which file systems are to be mounted in the cache. When a user on that client attempts to access files that are part of the back file system, those files are placed in the cache. Note that the cache does not get filled until a user requests access to a file or files. Therefore, the initial request to access a file will be at normal NFS speeds, but subsequent accesses to the same file will be at local JFS file system speeds. Refer to Figure 43 to see the relationship of the components in CacheFS.

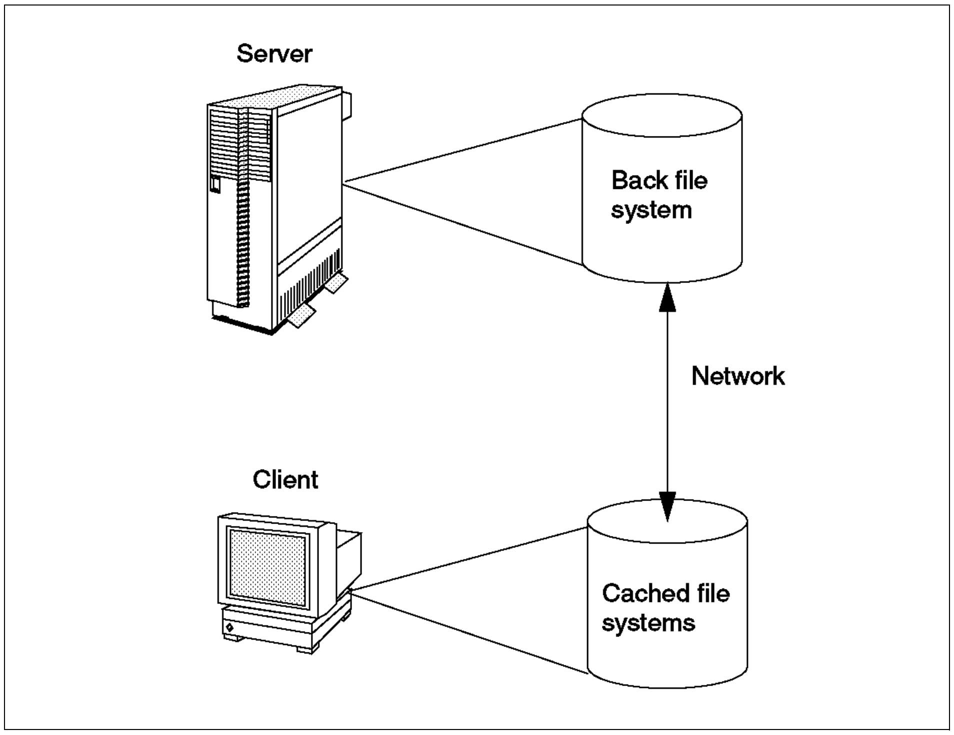


Table 43. CacheFS Components

8.21.1.2 Configuring CacheFS

There are two steps involved in setting up a cached file system:

1. Create the local cache.
2. Specify and mount the file systems to cached.

Creating the Local Cache File System

A local cache file system is created by using the `cfsadmin` command with the `-c` flag, as shown in the following steps:

1. Login as root.
2. Create a cache using the `-c` flag of the `cfsadmin` command

```
# cfsadmin -c -o <parameters> <cache-directory>
```

where `parameters` specify resource parameters, and `cache-directory` is the name of the directory where the cache should be created. A list of configurable parameters is shown in Table 44 on page 441. Although the cache is referred to as a *cache file system*, it is not a file system in the true sense. It is, in fact, a cache directory, which resides on a normal JFS. For this reason, if you are creating a large cache file system, it is advisable to create a dedicated JFS to be used for this purpose. This is because the cache file system is created with parameters that indicate the percentage of the underlying file system it is allowed to use.

3. Mount the back file system in the cache

```
# mount -V cachefs -o backfstype=nfs,cachedir=/<cache-directory> \
  remhost:/<remote-directory> <local-mount-point>
```

where `remhost:/<remote-directory>` is the name of the remote host and file system where the data resides, and `<local-mount-point>` is the mount point on the client where the remote file system should be mounted.

CacheFS can also be administered using SMIT.

Cached File Systems Consistency Checking

To ensure that the cached directories and files are kept up to date, CacheFS periodically checks consistency of files stored in the cache. To check consistency, CacheFS compares the current modification time to the previous modification time. If the modification times are different, all data and attributes for the directory or file, are purged from the cache, and new data and attributes are retrieved from the back file system.

When a user requests an operation on a directory or file, CacheFS checks if it is time to verify consistency. If so, CacheFS obtains the modification time from the back file system and performs the comparison.

Note: It is important to remember that CacheFS is intended to be used as a mechanism for reducing network and server workload. Because the remote file system data is held locally on the client, and consistency is only checked at intervals, it means that the data on the server can change, and the client is unaware of this for a period of time. You should, therefore, only use it for read-only, or read-mostly, file systems where the file system contents do not change rapidly.

One example where CacheFS would be suitable is in a CAD environment where master-copies of drawing components can be held on the server and cached-copies kept on the client workstation when in use.

8.21.1.3 CacheFS Commands

New commands specific to CacheFS are:

- `cfsadmin`
- `fsck_cachefs`

cfsadmin

The `cfsadmin` command provides for the following CacheFS administration functions:

- Creating cached file systems
- Deleting cached file systems
- Listing cache contents and statistics
- Changing CacheFS resource parameters

The `cfsadmin` command syntax is:

```
cfsadmin -c [ -o cachefs-parameters ] cache_directory
cfsadmin -d [ cache_id | all ] cache_directory
cfsadmin -l cache_directory
cfsadmin -s [ mount point | all ] cache_directory
cfsadmin -u [ -o cachefs-parameters ]
```

Table 44 details the `cfsadmin` options.

Table 44. cfsadmin Options

Flag	Description
-c	Creates a cache under the directory specified by `cache_directory`. The directory must not exist prior to cache creation.
-d	Removes the file system specified by `cache_id` and releases its resources or removes all file systems if `all` is specified.
-l	Lists file systems stored in the specified cache with their statistics.
-s	Requests a consistency check on the specified file system, or all file systems, if `all` is specified.

Flag	Description
-u	Updates resource parameters of the specified cache directory. Note that parameter values can only be increased, decreasing requires cache removal and recreation.
-o	CacheFS resource parameters see Table 45 for details.

Table 45 details CacheFS resource parameters.

Table 45. CacheFS Resource Parameters

Parameter	Description
maxblocks=n	Maximum amount of storage space that CacheFS can use, expressed as a percentage of the total number of blocks in the front file system. Default=90%
minblocks=n	Minimum amount of storage space that CacheFS is always allowed to use, expressed as a percentage of the total number of blocks in the front file system. Default=0%
threshblocks=n	A percentage of the total blocks in the front file system beyond which, CacheFS cannot claim resources once its block usage has reached the minblocks level. Default=85%
maxfiles=n	Maximum number of files that CacheFS can use, expressed as a percentage of the total number of inodes in the front file system. Default=90%
minfiles=n	Minimum number of files that CacheFS is always allowed to use, expressed as a percentage of the total number of inodes in the front file system. Default=0%
maxfilesize=n	Largest file size, expressed in megabytes, that CacheFS is allowed to cache. Default=3

fsck_cachefs

The `fsck_cachefs` command checks the integrity of cached file systems. By default it corrects any CacheFS problems it finds. Unlike the standard `fsck` command there is no interactive mode.

The `fsck_cachefs` command syntax is:

```
fsck -F cachefs [ -m | -o noclean ] cache_directory
```

Table 46 provides the available flags for this command.

Table 46. fsck_cachefs Options

Flag	Description
-m	Check, but do not repair.
-o noclean	Force a check on the cache even if there is no reason to suspect there is a problem.
cache-directory	The name of the directory where the cache resides.

The following example shows the actions required to allow an NFS client machine to mount the remote file system /images from the server aix4xdev and use the CacheFS mechanism to improve performance.

The /images file system on the server is a large file system, so a dedicated JFS will be used to store the local cache.

1. Use SMIT to create a JFS of the required size to act as the cache. For example, /cacheFS.
2. Mount the JFS to be used for the cache.

   ```
   mount /cacheFS
   ```

   Create an empty cache structure to be used as the cache. This is created using the cfsadmin command. The argument is the name of the cache directory object you want to create. The object should not exist.

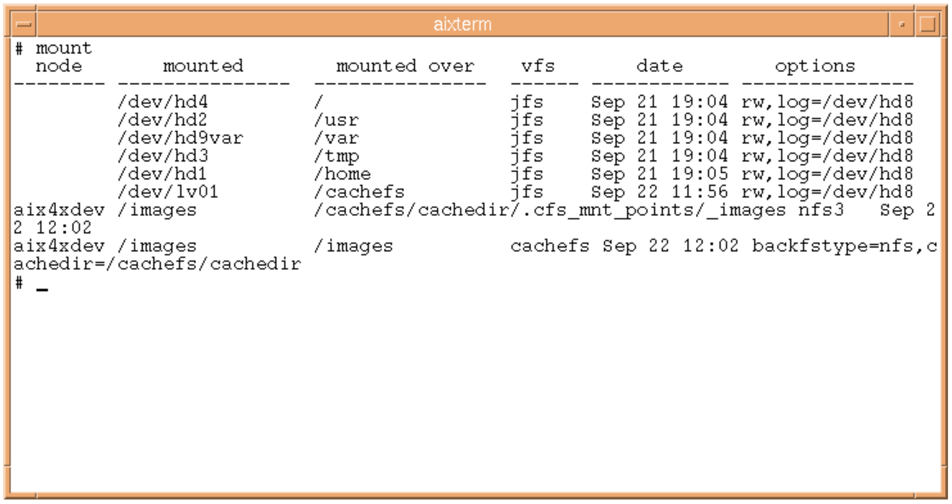
   ```
   cfsadmin -c /cacheFS/cachedir
   ```

   The command uses default values for the various resource thresholds of the cache object. If you want to alter these, you should create the cache object using SMIT with the cachefs_admin_create fastpath.
3. Mount the remote file system from the server as a cacheFS file system type, specifying the cache object to use for backing store and the name of the mount point that will be used to access the file system.

   ```
   mount -V cachefs -o backfstype=nfs,cachedir=/cachefs/cachedir
   aix4xdev:/images /images
   ```

   The result of the action can be checked using the `mount` command with no arguments to list all mounted file systems. Three new entries are present. The first entry is for the JFS used to store the cache object. The second shows the remote file system mounted onto the cache object as an NFS mount. The remote file system should not be accessed through this mount point. The third new file system mount shows /images as a CacheFS mount on the cache object.

Figure 78 shows the mount output of a CacheFS enabled system.



```
# mount
  node       mounted        mounted over    vfs       date        options
-------- --------------- --------------- ------ ------------ ---------------
         /dev/hd4        /               jfs    Sep 21 19:04 rw,log=/dev/hd8
         /dev/hd2        /usr            jfs    Sep 21 19:04 rw,log=/dev/hd8
         /dev/hd9var     /var            jfs    Sep 21 19:04 rw,log=/dev/hd8
         /dev/hd3        /tmp            jfs    Sep 21 19:04 rw,log=/dev/hd8
         /dev/hd1        /home           jfs    Sep 21 19:05 rw,log=/dev/hd8
         /dev/lv01       /cachefs        jfs    Sep 22 11:56 rw,log=/dev/hd8
aix4xdev /images         /cachefs/cachedir/.cfs_mnt_points/_images nfs3   Sep 2
2 12:02
aix4xdev /images         /images         cachefs Sep 22 12:02 backfstype=nfs,c
achedir=/cachefs/cachedir
# _
```

Figure 78. Output of mount Command Showing CacheFS

8.21.2 AutoFS (4.3.1)

AutoFS is the component of ONC+ that provides automatic mounting of NFS file systems. The automounting file system, AutoFS, mounts file systems when access is requested and unmounts the file system after a few minutes of inactivity, thus saving the network overhead traffic required to maintain the NFS connection. AutoFS allows you to break the connection when the file system is not being used and restart it again automatically when access is next desired.

8.21.2.1 How AutoFS Works

AutoFS is a client-side service. It is implemented using three components to accomplish automatic mounts. The components are:

- The `automount` command
- The autofs kernel extension
- The `automountd` daemon

The `automount` command is called at system startup time from /etc/rc.tcpip. It loads the autofs kernel extension if it is not already loaded and reads the master map information from the file /etc/auto_master. The `automount` command then passes the information it read from the master map to the autofs kernel extension. It then starts the `automountd` daemon and terminates.

The autofs kernel extension reads the information passed to it from the `automount` command and maintains an internal table of the autofs mounts. These autofs mounts are not automatically mounted at startup time. They are points under which file systems can be mounted in the future.

When a client attempts to access a file system that is not presently mounted, the autofs kernel extension intercepts the request and gets `automountd` to mount the requested directory. The `automountd` daemon locates the directory, mounts it within autofs, and replies. On receiving the reply, autofs allows the waiting request to proceed. Subsequent references to the mount are redirected by the autofs. No further participation is required by `automountd`.

With this implementation of automatic mounting, the `automountd` daemon is completely independent from the `automount` command. Because of this separation, it is possible to add, delete, or change map information without first having to stop and start the `automountd` daemon process. Once the file system is mounted, further access does not require any action from `automountd`.

8.21.2.2 AutoFS Maps

AutoFS uses files referred to as maps for administration of network files. The map files contain information about file systems and the names of the hosts on the network where the file systems reside. Maps can be available locally or through a network name service like NIS. AutoFS uses three types of maps:

- Master maps
- Direct maps
- Indirect maps

8.21.2.3 Master Maps

The auto_master map associates a directory with a map. It is a master list specifying all the maps that AutoFS should know about.

Each line in the master map `/etc/auto_master` has the following syntax:

```
mount-point map-name [ mount-options ]
```

Mount-point

mount-point is the full (absolute) path name of a directory. If the directory does not exist, AutoFS creates it if possible. If the directory exists and is not empty, mounting on it hides its contents. In this case, AutoFS issues a warning message.

Map-name
map-name is the map AutoFS uses to find directions to locations, or mount information. If the name is preceded by a slash (/), AutoFS interprets the name as a local file. Otherwise, AutoFS searches for the mount information using the search specified in the name service switch configuration file.

Mount-options
mount-options is an optional, comma-separated list of options that apply to the mounting of the entries specified in map-name unless the entries in map-name list other options. The mount-options are the same as those for a standard NFS mount, except that bg (background) and fg (foreground) do not apply.

The auto_master map also has two special entries. They are:

```
+auto_master
/net           -hosts          -nosuid
```

The +auto_master entry is a reference to an external NIS master map. If one exists, then its entries are read as though they occurred in place of the +auto_master entry.

The /net entry is a mechanism that allows an NFS client to access all file systems exported by a server. For example, accessing /net/fred will cause the AutoFS system to mount all of the file systems exported by the machine fred under the root mount point /net/fred.

8.21.2.4 Direct Maps

A direct map is an automount point. With a direct map, there is a direct association between a mount point on the client and a directory on the server. Direct maps have a full path name and indicate the relationship explicitly. Lines in direct maps have the following syntax:

```
      mount-point [ mount-options ] location
```

Key
The key is the path name of the mount point in a direct map.

Mount-options
The mount-options are the options you want to apply to this particular mount. They are required only if they differ from the map default.

Location
The location is the location of the file system. Specified as `server:pathname`.

An example of typical `/etc/auto_direct` map is:

```
/usr/local      -ro \
    /bin                   goanna:/export/local/bin \
    /share                 goanna:/export/local/share \
    /src                   goanna:/export/local/src
/usr/man        -ro        echidna:/usr/man \
                           platypus:/usr/man \
                           wombat:/usr/man \
/usr/games      -ro        koala:/usr/games
/usr/spool/news -ro        wallaby:/usr/spool/news \
                           wombat:/var/spool/news
```

8.21.2.5 Indirect Maps

An indirect map uses a substitution value of a key to establish the association between a mount point on the client and a directory on the server. Indirect maps are useful for accessing specific file systems like home directories. The auto_home map is an example of an indirect map. Lines in indirect maps have the following syntax:

```
key [ mount-options ] location
```

Key

key is a simple name (no slashes) in an indirect map.

Mount-options

The mount-options are the options you want to apply to this particular mount. They are required only if they differ from the map default.

Location

The `location` is the location of the file system, specified as server:pathname.

An example of a typical `auto_home` map is:

```
dale                        wombat:/export/home/dale
victor                      taipan:/export/home/victor
andrew                      emu:/export/home/andrew
swamy                       wallaby:/export/home/swamy
julia                       redback:/export/home/julia
tony                        goanna:/export/home/tony
george       -rw,nosuid     koala:/export/home/george
```

As an example, assume that the previous map is on host `echidna`. If user george has an entry in the password database specifying his home directory as /home/george, then whenever he logs into computer echidna, AutoFS mounts the directory/export/home/george residing on the computer koala.

8.21.3 NFS Server Performance Enhancement (4.3.2)

The NFS server performance of AIX 4.3.2 is enhanced with the implementation of a vnode cache in the JFS component of the kernel. The cache enables the NFS server code to translate an NFS file handle to a local vnode structure more efficiently than previous versions of AIX. As a result, the NFS server code spends less time holding a VFS lock word, which in turn, increases the available throughput of the NFS server.

8.21.4 NIS+ (4.3.3)

AIX version 4.3.3 provides a port of SUN’s NIS+ version 3 software.

NIS+ expands the network name service provided by NIS. NIS+ enables to store information about workstation addresses, security information, mail information, Ethernet interfaces, and network services in central locations where all workstations on a network can access it. This configuration of network information is referred to as the NIS+ namespace.

The NIS+ namespace is hierarchical and can be configured to conform to the logical hierarchy of an organization. An NIS+ namespace can be divided into multiple domains, each of which can be administered autonomously. Clients may have access to information in other domains as well as their own if they have the appropriate permissions.

NIS+ uses a client/server model to store and have access to the information contained in an NIS+ namespace. Each domain is supported by a set of servers. The principal server is called the *master* server and the backup servers *replicas*. The network information is stored in standard NIS+ tables in an internal NIS+ database. Both master and replica servers run NIS+ server software and both maintain copies of NIS+ tables. Changes made to the NIS+ data on the master server are incrementally and automatically propagated to the replicas.

NIS+ includes a sophisticated security system to protect the structure of the namespace and its information. It uses authentication and authorization to verify whether a client's request for information should be fulfilled. Authentication determines whether the information requester is a valid user on the network. Authorization determines whether a particular user is allowed to have or modify the information requested.

8.21.4.1 Differences between NIS and NIS+

NIS+ differs from NIS in several ways. It has many new features and the terminology for similar concepts is different. The following table gives an overview of the major differences between NIS and NIS+.

Table 47. NIS and NIS+ Differences

NIS	NIS+
Machine name and user's name can be the same	Machine name and user names must be unique. Furthermore, you cannot have a dot (.) in your machine or user name.
Domains are flat: no hierarchy.	Domains are hierarchical: data stored in different levels in the namespace.
Names and commands are case sensitive.	Names and commands are not case sensitive.
Data is stored in two-column maps.	Data is stored in multi-column tables.
Uses no authentication.	Uses DES authentication.
An NIS record has a maximum size of 1024 bytes. This limitation applies to all NIS map files. For example, a list of users in a group can contain a maximum of 1024 characters in single-byte character set file format.	The NIS+ record has no limit.
Provides single choice of network information source.	Client chooses information source: NIS, NIS+, DNS, or local /etc files.
Updates are delayed for batch propagation.	Incremental updates are propagated immediately.

NIS+ is designed to replace NIS, not to enhance it. NIS was intended to address the administration requirements of smaller client/server computing networks. Typically, NIS works best in environments with no more than a few hundred clients, a few multipurpose servers, only a few remote sites, and trusted users.

The size and complexity of modern client/server networks require new, autonomous administration practices. NIS+ was designed to meet the requirements of networks that typically range from 100-10,000 multivendor clients supported by 10-100 specialized servers located in sites throughout the world. In addition, the information they store can change rapidly.

Because more distributed networks require scalability and decentralized administration, the NIS+ namespace was designed with hierarchical domains that may be administered independently.

Although this division into domains makes administration more autonomous and growth easier to manage, it does not make information harder to access. Clients have the same access to information in other domains as they would have had under one umbrella domain. A domain can even be administered from within another domain.

8.21.4.2 NIS Compatibility Mode

NIS-compatibility mode enables an NIS+ server to answer requests from NIS clients while continuing to answer requests from NIS+ clients. NIS+ does this by providing two service interfaces. One responds to NIS+ client requests, while the other responds to NIS client requests.

This mode does not require any additional setup or changes to NIS clients that are not even aware that the server that is responding is not an NIS server. However there are some differences including the fact that the NIS+ server running in NIS-compatibility mode does not support the `ypupdate` and `ypxfr` protocols and thus it cannot be used as a replica or master NIS server.

Instructions for setting up a server in NIS-compatibility mode are slightly different than those used to set up a standard NIS+ server and NIS-compatibility mode has security implications for tables in the NIS+ namespace. Since the NIS client software does not have the capability to provide the credentials that NIS+ servers expect from NIS+ clients, all their requests end up classified as unauthenticated. Therefore, to allow NIS clients to access information in NIS+ tables, those tables must provide access rights to unauthenticated requests. This is handled automatically by the utilities used to set up a server in NIS-compatibility mode.

8.21.4.3 Commands and Directories

NIS+ provides a full set of commands for administering a namespace. The following table summarizes them. For a complete description of syntax and options, see their command descriptions.

Table 48. NIS+ Namespace Administration Commands

Command	Description
`nisaddcred`	Creates credentials for NIS+ principals and stores them in the cred table.
`nisaddent`	Adds information from /etc files or NIS maps into NIS+ tables.

Command	Description
niscat	Displays the contents of NIS+ tables.
nischgrp	Changes the group owner of an NIS+ object.
nischmod	Changes an object’s access rights.
nischown	Changes the owner of an NIS+ object.
nischttl	Changes an NIS+ object’s time-to-live value.
nisdefaults	Lists an NIS+ object’s default values: domain name, group name, workstation name, NIS+ principal name, access rights, directory search path, and time-to-live.
nisgrep	Searches for entries in an NIS+ table.
nisgrpadm	Creates or destroys an NIS+ group, or displays a list of its members. Also adds members to a group, removes them, or tests them for membership in the group.
nisinit	Initializes an NIS+ client or server.
nisln	Creates a symbolic link between two NIS+ objects.
nisls	Lists the contents of an NIS+ directory.
nismatch	Searches for entries in an NIS+ table.
nismkdir	Creates an NIS+ directory and specifies its master and replica servers.
nismkuser	Creates an NIS+ user.
nispasswd	*Not supported* in AIX. Use the passwd command.
nisrm	Removes NIS+ objects (except directories) from the namespace.
nisrmdir	Removes NIS+ directories and replicas from the namespace.
nisrmuser	Removes an NIS+ user.
nissetup	Creates *org_dir* and *groups_dir* directories and a complete set of (unpopulated) NIS+ tables for an NIS+ domain.
nisshowcache	Lists the contents of the NIS+ shared cache maintained by the NIS+ cache manager.
nistbladm	Creates or deletes NIS+ tables, and adds, modifies or deletes entries in an NIS+ table.
nisupdkeys	Updates the public keys stored in an NIS+ object.

Command	Description
passwd	Changes password information stored in the NIS+ *passwd* table.

NIS+ configuration files are stored in several directories, that change from client to server. The following table shows the directory usage.

Table 49. Where NIS+ files are Stored

Directory	Where	Contains
/usr/bin	All machines	NIS+ user commands
/usr/lib/nis	All machines	NIS+ administrator commands
/usr/sbin	All machines	NIS+ daemons
/usr/lib/	All machines	NIS+ shared libraries
/var/nis/data	NIS+ server	Data files used by NIS+ server
/var/nis	NIS+ server	NIS+ working files
/var/nis	NIS+ client machines	Machine-specific data files used by NIS+

8.21.4.4 Structure of the NIS+ Namespace

The NIS+ namespace is the way information is stored by NIS+. The namespace can be arranged in a variety of ways to suit the needs of an organization. For example, if an organization had three divisions, its NIS+ namespace would likely be divided into three parts, one for each division. Each part would store information about the users, workstations, and network services in its division, but the parts could easily communicate with each other. Such an arrangement would make information easier for the users to access and for the administrators to maintain.

Although the arrangement of an NIS+ namespace can vary from site to site, all sites use the same structural components: directories, tables, and groups. These components are called NIS+ objects. NIS+ objects can be arranged into a hierarchy, like a UNIX file tree. For example, Figure 79 shows, on the left, a namespace that consists of three directory objects, three group objects, and three table objects; on the right it shows a file system that consists of three directories and three files.

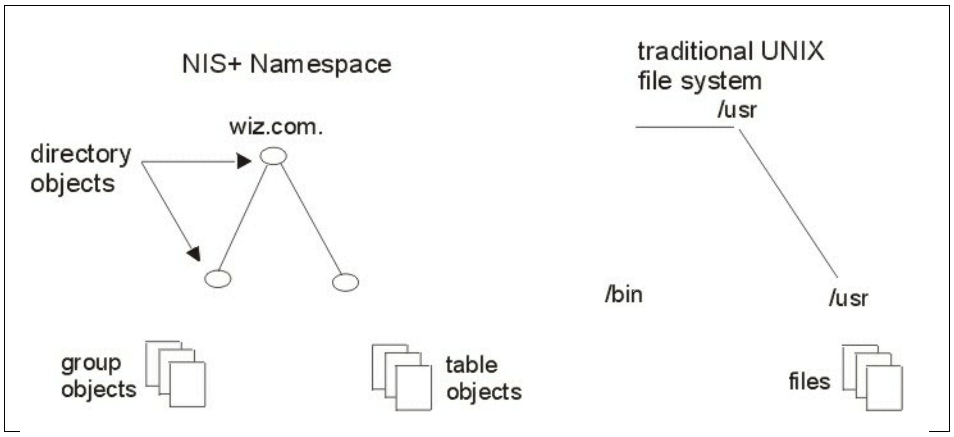


Figure 79. NIS+ Namespace Hierarchy

Although an NIS+ namespace resembles a traditional UNIX file system, it has five important differences:

- Although both use directories, the other objects in an NIS+ namespace are tables and groups, not files.
- The NIS+ namespace is administered only through NIS+ administration commands or graphical user interfaces designed for that purpose, such as Web-Based System Manager or SMIT tools. It cannot be administered with standard UNIX file system commands or graphic user interfaces.
- The names of AIX file system components are separated by slashes (/usr/bin), but the names of NIS+ namespace objects are separated by dots (wiz.com.).
- The root of an AIX file system is reached by stepping through directories from right to left (/usr/src/file1), while the root of the NIS+ namespace is reached by stepping from left to right (sales.wiz.com.).
- Because NIS+ object names are structured from left to right, a fully qualified name always ends in a dot. Any NIS+ object ending in a dot is assumed to be a fully qualified name. NIS+ object names that do not end in a dot are assumed to be relative names.

Directories

Directory objects are the skeleton of the namespace. When arranged into a tree-like structure, they divide the namespace into separate parts. A directory hierarchy is similar to an inverted tree, with the root of the tree at the top and the branches toward the bottom. The topmost directory in a namespace is the root directory. If a namespace is flat, it has only one directory, but that directory is nevertheless the root directory. The directory objects beneath the root directory are simply called directories.

When identifying the relation of one directory to another, the directory beneath is called the child directory and the directory above is called the parent directory.

Any NIS+ directory that stores NIS+ groups is named *groups_dir*. Any directory that stores NIS+ system tables is named *org_dir*. An example is in Figure 80.

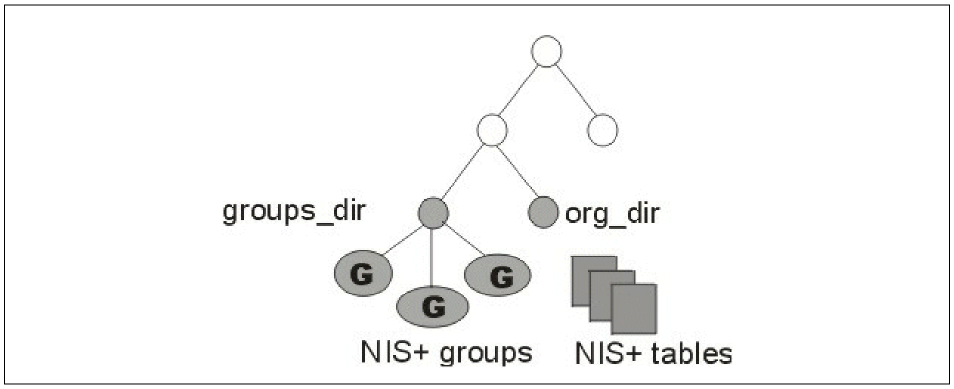


Figure 80. NIS+ System Tables

You can arrange directories, tables, and groups into any structure that you like. However, NIS+ directories, tables, and groups in a namespace are normally arranged into configurations called domains that are designed to support separate portions of the namespace.

Domains
An NIS+ domain consists of a directory object, its org_dir directory, its groups_dir directory, and a set of NIS+ tables.

NIS+ domains are not tangible components of the namespace. They are simply a convenient way to refer to sections of the namespace that are used to support real-world organizations. For example, assume that the Wizard Corporation has a Sales division and an Manufacturing division. To support those divisions, its NIS+ namespace would most likely be arranged into three major directory groups, with a structure such as the one shown in Figure 81.

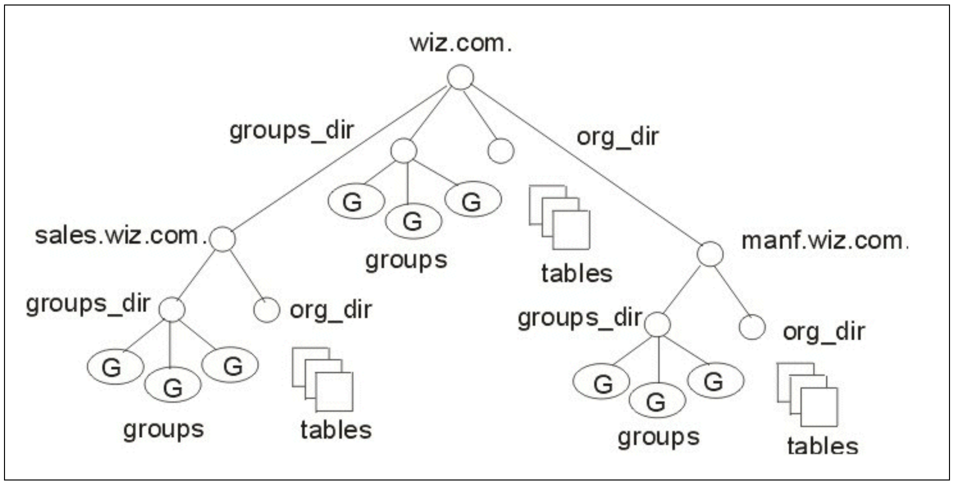


Figure 81. NIS+ Domain Hierarchy

Instead of referring to such a structure as three directories, six subdirectories, and several additional objects, referring to it as three domains (wiz.com., sales.wiz.com., and manf.wiz.com.) is more convenient.

Tables

Tables are the objects that contain informations. They have a column-entry structure, accept search paths, can be linked together, and can be set up in several different ways. NIS+ provides pre-configured system tables, and you can also create your own tables. The following table lists the pre-configured NIS+ tables.

Table 50. Pre-configured NIS+ Tables

Table Name	Description
hosts	Network address and host name of every workstation in the domain
bootparams	Location of the root, swap, and dump partition of every diskless client in the domain
passwd	Password information about every user in the domain.
cred	Credentials for principals who belong to the domain
group	The group name, group password, group ID, and members of every UNIX group in the domain
netgroup	The netgroups to which workstations and users in the domain may belong

Table Name	Description
mail_aliases	Information about the mail aliases of users in the domain
timezone	The time zone of every workstation in the domain
networks	The networks in the domain and their canonical names
netmasks	The networks in the domain and their associated netmasks
ethers	The Ethernet address of every workstation in the domain
services	The names of IP services used in the domain and their port numbers
protocols	The list of IP protocols used in the domain
RPC	The RPC program numbers for RPC services available in the domain
auto_home	The location of all user’s home directories in the domain
auto_master	Automounter map information
sendmailvars	The mail domain
client_info	Information about NIS+ clients

These tables store a wide variety of information, ranging from user names to Internet services. Most of this information is generated during a setup or configuration procedure.

A table contains information only about its local domain. For instance, tables in the wiz.com. domain contain information only about the users, clients, and services of the wiz.com. domain. The tables in the sales.wiz.com. domain store information only about the users, clients, and services of the sales.wiz.com. domain. And so forth.

If a client in one domain tries to find information that is stored in another domain, it has to provide a fully qualified name or the NIS_PATH environment variable is used.

Groups

Group definitions are, like UNIX groups, collections of NIS+ principals and are used to allow access to NIS+ tables. They share nothing with UNIX groups that are present on the server. The most important NIS+ group is the *admin* group that contains all the principals that are allowed administrative operations in the domain’s namespace.

8.21.4.5 Servers

Every NIS+ domain is supported by a set of NIS+ servers. The servers store the domain's directories, groups, and tables, and answer requests for access from users, administrators, and applications. Each domain is supported by only one set of servers. However, a single set of servers can support more than one domain.

A server that supports a domain is associated with the domain's main directory, as you can see in Figure 82.

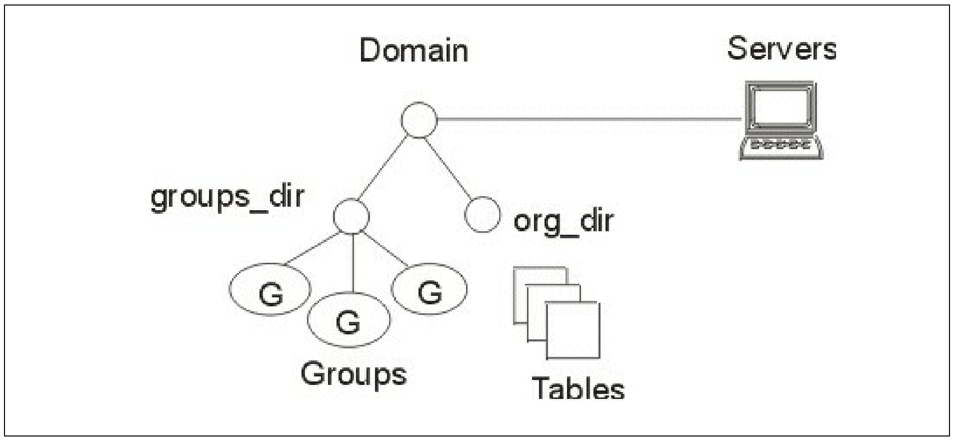


Figure 82. NIS+ Server Position in a Domain

This connection between the server and the directory object is established during the process of setting up a domain. When that connection is established, the directory object stores the name and IP address of its server. This information is used by clients to send requests for services.

Two types of servers support an NIS+ domain: a master and its replicas. The master server of the root domain is called the *root master server*. A namespace has only one root master server. The master servers of other domains are simply called master servers. Likewise, there are root replica servers and regular replica servers.

Both master and replica servers store NIS+ tables and answer client requests. The master, however, stores the master copy of a domain's tables. The replicas store only duplicates. The administrator loads information into the tables in the master server, and the master server propagates it to the replica servers.

This arrangement has two benefits. First, it avoids conflicts between tables because only one set of master tables exists; the tables stored by the replicas are only copies of the masters. Second, it makes the NIS+ service much more available. If either the master or a replica is down, another server can act as a backup and handle the requests for service.

Note

If the master server is unavailable and the NIS+ domain is being served solely by a replica, you can obtain information from the NIS+ tables, but changes to the original tables can be made only when the master server is available.

An NIS+ master server implements updates to its objects immediately; however, it tries to *batch* several updates together before it propagates them to its replicas. When a master server receives an update to an object, whether a directory, group, link, or table, it waits about two minutes for any other updates that may arrive. Once it is finished waiting, it stores the updates in two locations: on disk and in a transaction log (it has already stored the updates in memory).

The transaction log is used by a master server to store changes to the namespace until they can be propagated to replicas. A transaction log has two primary components: updates and time stamps, as shown in Figure 83.

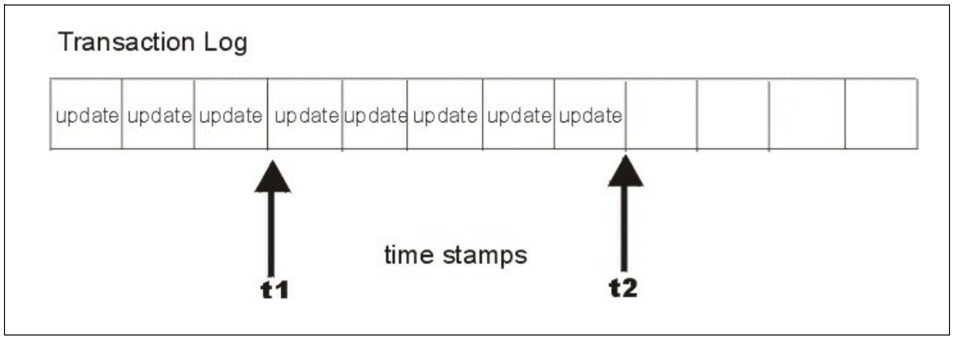


Figure 83. NIS+ Transaction Log

An update is an actual copy of a changed object. For instance, if a directory has been changed, the update is a complete copy of the directory object. If a table entry has been changed, the update is a copy of the actual table entry.

The time stamp indicates the time at which an update was made by the master server.

After recording the change in the transaction log, the master sends a message to its replicas, telling them that it has updates to send them. Each replica replies with the time stamp of the last update it received from the master. The master then sends each replica the updates it has recorded in the log since the replica’s time stamp, as described in Figure 84.

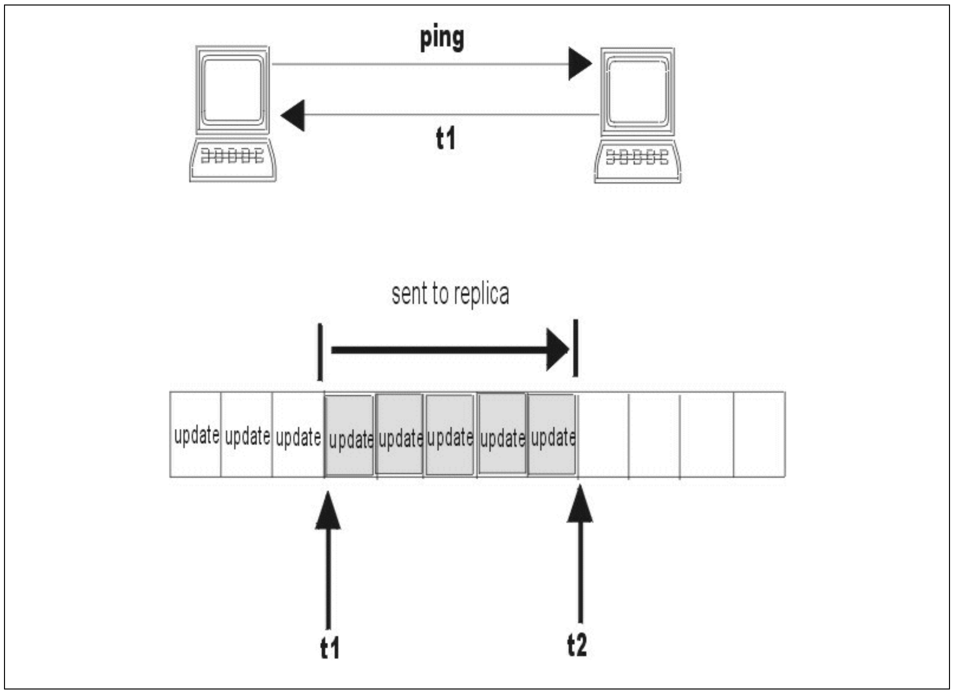


Figure 84. NIS+ Replica

When the master server updates *all* its replicas, it clears the transaction log. In some cases, such as when a new replica is added to a domain, the master receives a time stamp from a replica that is before its earliest time stamp still recorded in the transaction log. In this situation, the master server performs a *full resynchronization*, or *resync*. A resync downloads all the objects and information stored in the master down to the replica. During a resync, both the master and replica are busy. The replica cannot answer requests for information; the master can answer read requests but cannot accept update requests.

8.21.4.6 Client

An NIS+ client is a workstation that has been set up to receive NIS+ service. Setting up an NIS+ client consists of establishing security credentials, making it a member of the proper NIS+ groups, verifying its home domain, and, finally, running the NIS+ initialization script.

An NIS+ client can access any part of the namespace, if it has been authenticated and has been granted the proper permissions. Anyway, a client *belongs* to only one domain, which is referred to as its *home domain*. A client’s home domain is usually specified during installation, but it can be changed or specified later. All the information about a client, such as its IP address and its credentials, is stored in the NIS+ tables of its home domain.

When a client is initialized, it is given a *cold-start* file. The cold-start file gives a client a copy of a directory object that it can use as a starting point for contacting servers in the namespace. The directory object contains the address, public keys, and other information about the master and replica servers that support the directory. Normally, the cold-start file contains the directory object of the client’s home domain.

A cold-start file is used only to initialize a client’s *directory cache*. The directory cache, managed by an NIS+ facility called the *cache manager*, stores the directory objects that enable a client to send its requests to the proper servers.

By storing a copy of the namespace’s directory objects in its directory cache, a client can know which servers support which domains. To view the contents of a client’s cache, use the `nisshowcache` command.

To keep these copies up-to-date, each directory object has a time-to-live (TTL) field. Its default value is 12 hours. If a client looks in its directory cache for a directory object and finds that it has not been updated in the last 12 hours, the cache manager obtains a new copy of the object. You can change a directory object’s time-to-live value with the `nischttl` command.

The cold-start file provides the first entry in the cache. When the client sends its first request, it sends the request to the server specified by the cold-start file. If the request is for access to the domain supported by that server, the server answers the request.

If the request is for access to another domain, the server listed in the cache file tries to help the client locate the proper server. If the server has an entry for that domain in its own directory cache, it sends a copy of the domain’s directory object to the client. The client loads that information into its directory cache for future reference and sends its request to that server.

In the unlikely event that the server does not have a copy of the directory object the client is trying to access, it sends the client a copy of the directory object for its own home domain, which lists the address of the server's parent. The client repeats the process with the parent server, and keeps trying until it finds the proper server or until it has tried all the servers in the namespace.

Over time, the client accumulates in its cache a copy of all the directory objects in the namespace and thus the IP addresses of the servers that support them. When it needs to send a request for access to another domain, it can usually find the name of its server in its directory cache and send the request directly to that server.

8.21.4.7 Naming Conventions

Objects in an NIS+ namespace can be identified with two types of names: *partially qualified* and *fully qualified*. A partially qualified name, also called a *simple* name, is simply the name of the object or any portion of the fully qualified name. If during any administration operation you type the partially qualified name of an object or principal, NIS+ will attempt to expand the name into its fully qualified version.

A fully qualified name is the complete name of the object, including all the information necessary to locate it in the namespace, such as its parent directory, if it has one, and its complete domain name, including a trailing dot.

Domain Names

A fully qualified NIS+ domain name is formed from left to right, starting with the local domain and ending with the root domain:

```
wiz.com.
sales.wiz.com.
intl.sales.wiz.com.
```

The first line shows the name of the root domain. *The root domain must always have at least two labels and must end in a dot*.

Directory Object Names

A directory's simple name is the name of the directory object. Its fully qualified name consists of its simple name plus the fully qualified name of its domain (which always includes a trailing dot):

```
groups_dir (simple name)
groups_dir.manf.wiz.com. (fully qualified name)
```

The simple name is normally used from within the same domain, and the fully qualified name is normally used from a remote domain.

Tables and Group Names

Fully qualified table and group names are formed by starting with the object name and appending the directory name, followed by the fully qualified domain name. Remember that all system table objects are stored in an org_dir directory and all group objects are stored in a groups_dir directory. If you create your own NIS+ tables, you can store them anywhere you like.

Here are some examples of group and table names:

```
admin.groups_dir.wiz.Inc.       admin.groups_dir.wiz.com.
admin.groups_dir.sales.wiz.Inc. admin.groups_dir.sales.wiz.com.
hosts.org_dir.wiz.Inc.          hosts.org_dir.wiz.com.
hosts.org_dir.sales.wiz.Inc.    hosts.org_dir.sales.wiz.com.
```

Table Entry Names

To identify an entry in an NIS+ table, you need to identify the table object and the entry within it. This type of name is called an indexed name. It has the following syntax:

```
[column=value,column=value,...],table-name
```

Column is the name of the table column. Value is the actual value of that column. Table-name is the fully qualified name of the table object. Here are a few examples of entries in the hosts table:

```
[addr=129.44.2.1,name=pine],hosts.org_dir.sales.wiz.com.
[addr=129.44.2.2,name=elm],hosts.org_dir.sales.wiz.com.
[addr=129.44.2.3,name=oak],hosts.org_dir.sales.wiz.com.
```

You can use as few column-value pairs inside the brackets as required to uniquely identify the table entry.

Host Names

Host names may contain up to 24 characters. Letters, numbers, the dash (-) and underscore (_) characters are allowed in host names. Host names are not case sensitive. The first character of a host name must be a letter of the alphabet. Blank spaces or dots (.) are not permitted in host names.

Domains and hosts should not have the same name. For example, if you have a sales domain you should not have a machine named sales. Similarly, if you have a machine named home, you do not want to create a domain named home. This caution applies to subdomains, for example if you have a machine named west you do not want to create a sales.west.myco.com subdomain.

Name Expansion

Entering fully qualified names with your NIS+ commands can quickly become tedious. To ease the task, NIS+ provides a name-expansion facility. When you enter a partially qualified name, NIS+ attempts to find the object by looking for it under different directories. It starts by looking in the default domain. This is the home domain of the client from which you type the command. If it does not find the object in the default domain, NIS+ searches through each of the default domain’s parent directories in ascending order until it finds the object. It stops after reaching a name with only two labels.

You can change or augment the list of directories NIS+ searches through by changing the value of the environment variable NIS_PATH. NIS_PATH accepts a list of directory names separated by colons:

```
export NIS_PATH=sales.wiz.com.:manf.conf.wiz.:wiz.com.
```

The NIS_PATH variable accepts the special symbol, $. You can append the $ symbol to a directory name or add it by itself. If you append it to a directory name, NIS+ appends the default directory to that name.

8.21.4.8 Security

NIS+ security features protect the namespace from unauthorized access. Any entity that submits a request for an NIS+ service is called a *principal* and is identified by the credential it provides. Each NIS+ object is created with access rights that specify the types of operations principals are allowed to perform, like access rights are used for UNIX files.

Any operation on an NIS+ object falls into four categories (access rights: Read, Modify, Create and Destroy. Each object specifies the access rights of four categories of principals: Owner, Group, World and Nobody. You can use the `niscat -o` command to see the access right of each object. Those rights are given for the object or even parts of the object.

A principal may be the owner of the object or may belong to the group specified by the object. If not, it falls into the World group if it provides credentials recognized by the NIS+ server, otherwise the principal is considered belonging to the Nobody group. Typical Nobody principals are NIS clients that do not provide credentials.

NIS+ servers can operate at three security levels. These levels determine the types of credential must be submitted for its request to be authenticated. They are described in Table 51.

Table 51. NIS+ Security Levels

Security Level	Description
0	This level is designed for testing and setting up the initial NIS+ namespace. At this level any NIS+ principal has full access rights to all NIS+ objects in the domain.
1	Level 1 is designed for testing and debugging only.
2	Default and highest security level. It authenticates only using DES credentials. Requests that use LOCAL credentials or none at all are assigned the access rights granted to the Nobody class.

Credentials

NIS+ principals can have two types of credentials: LOCAL and DES. Non root users are called *client users* and may have both types. Root user on a specific workstation must be distinguished from the root user of another workstation: they are called *client workstation* and can only have DES credential.

DES credential information is stored in the cred table of the principal's home domain. LOCAL credentials are stored in the cred table in every domain: in order to log in to a remote domain, a client must have his LOCAL credentials in the cred table of the remote domain.

A LOCAL credential is simply used to map the principal with its UNIX user ID. The DES credential is more complex and is composed by a principal's secure RPC netname plus a verification field, as shown in Figure 85.

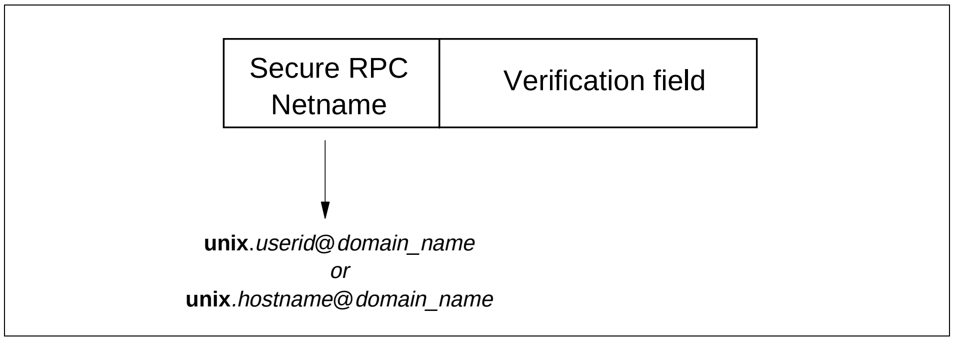

Secure RPC Netname
Verification field
unix.*userid@domain_name*
or
unix.*hostname@domain_name*

Figure 85. DES Credential

The secure RPC netname portion of the credential is the part used to actually identify the NIS+ principal. It begins with the prefix *unix* and the second field is the UNIX user ID for client users or the hostname for client workstations. The last field is the principal’s home domain, that is the domain that contains the user password entry and DES credential.

Note

Host names cannot begin with a digit. A wrong NIS+ host name like *4tune* creates a unix.4tune@domain netname that is considered as a unacceptable 4tune UNIX user ID.

The verification field of the credential contains a key that identifies the principal and makes sure the credential is not forged.

NIS+ credential are stored in the cred table contained in the org_dir directory. Each domain has its own cred table that stores the credentials of the client workstations that belong to the domain and client users that are allowed to log in.

The cred table has five columns:

Table 52. NIS+ cred Table Content

Name	Auth. Type	Auth. Name	Public Data	Private Data
Principal name of client user	LOCAL	user ID	group ID list	None
Principal name of client user or client workstation	DES	Secure RPC netname	Public key	Private key

The second column, Authentication Type, determines the type of values found in the other columns. If the authentication type is LOCAL, the other columns contain the client user’s principal name, the UNIX user ID, the list of group ID the user belongs to and the last field is empty. If the authentication is DES, the other columns contain the principal name, the secure RPC netname and the public and private keys used for secure authentication.

Credentials are created by the administrator using the `nisaddcred` command. In order to create a LOCAL credential the command simply extracts the client used id and group ID from the client’s login record and places it in the domain’s cred table.

When the command created the DES credential information, it first creates the secure RPC netname from the user ID and places the information in the cred table. Then it creates a pair of random but mathematically related 192-bit authentication keys (the key-pair) using the Diffie-Hellman cryptography scheme. One is the public key that is placed into the Public Data field of the cred table, the other is the private key that is encrypted with the network password of the user.

The network password must be provided by the administrator and is needed to access the private key. It may be different from the login password of the user, but they are normally kept equal.

Client's Access to Tables

When a client accesses an NIS+ table of a server, it must provide the its DES credential. The credential is created using the client's DES private key, the server's public key and a time stamp.

The encrypted private key of the client is retrieved from its home server's cred table and is then decrypted using the client's network password. The client must provide its network password using the `keylogin` command. If the login password is equal to the network password, `keylogin` is not needed. The private key is stored locally for future NIS+ requests.

The server's public key is retrieved from the cred table of the server's home server and is stored in the local workstation cache for future needs of the workstation.

The client uses its own private key and the server's public keys to generate a DES key, encrypts the time stamp with such key and uses the result to create the DES credential.

Server's Credential Verification

The server reverses the clients' encryption process. It uses the secure RPC netname portion of the credential to look up the principal's public key in the cred table. Then, using its own private key and the principal's public key it decrypts the DES key. Finally it uses the DES key to decrypt the time stamp. The time stamp is within a predetermined tolerance of the time the server received the message, the server authenticates the request.

This process satisfies the server. However, to let the client know that the information it receives indeed comes from a trusted server, the server encrypts the time stamp with the DES key and sends it back to the client.

Note

The authentication process uses time stamps. In order to avoid any problems, all servers and clients should have local clock aligned.

8.21.4.9 Setup Example

In this section we provide an example NIS+ configuration in a small environment, starting with a very basic setup and adding more and more complexity. We work on three machines, rootmaster, austinmaster and nisclient, whose network configuration is shown in Figure 86.

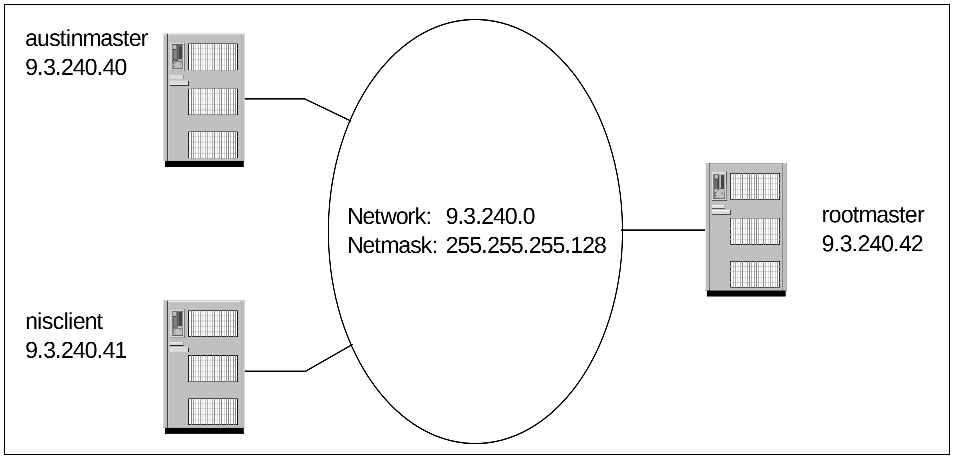


Figure 86. Network Configuration for NIS+ Example

On each machine we install the bos.net.nisplus fileset that provides both the client and the server NIS+ functions. The NIS fileset are also installed as they are prerequisite of NIS+:

```
# lslpp -l bos.net.nis\*
  Fileset                      Level  State      Description
  ----------------------------------------------------------------------------
Path: /usr/lib/objrepos
  bos.net.nis.client         4.3.3.0  COMMITTED  Network Information Service
                                                 Client
  bos.net.nis.server         4.3.3.0  COMMITTED  Network Information Service
                                                 Server
  bos.net.nisplus            4.3.3.0  COMMITTED  Network Information Services
                                                 Plus (NIS+)

Path: /etc/objrepos
  bos.net.nis.client         4.3.3.0  COMMITTED  Network Information Service
                                                 Client
  bos.net.nis.server         4.3.3.0  COMMITTED  Network Information Service
                                                 Server
  bos.net.nisplus            4.3.3.0  COMMITTED  Network Information Services
                                                 Plus (NIS+)
#
```

The Root Master Server

The first operation is to make rootmaster the NIS+ root master server, that is the server that provides information to the highest domain level in NIS+ hierarchy.

In order to ease our work, we add to /etc/hosts the name and IP addresses of all the machines in our environment and we add two users, nisplus1 and nisplus2, to the rootmaster machine. The their password is set:

```
# mkuser nisplus1
# passwd nisplus1
Changing password for "nisplus1"
nisplus1’s New password:
Enter the new password again:
# mkuser nisplus2
# passwd nisplus2
Changing password for "nisplus2"
nisplus2’s New password:
Enter the new password again:
```

As a first step we create a working directory where we put the configuration files used to create NIS+ tables. In out example we only use passwd, group and hosts:

```
# mkdir /nis+files
# cp /etc/passwd /etc/group /etc/hosts /nis+files/
```

Then we remove from the /nis+files/passwd file all the system users that we do not want to be distributed by NIS+:

```
# cat /nis+files/passwd
nisplus1:!:206:1::/home/nisplus1:/usr/bin/ksh
nisplus2:!:207:1::/home/nisplus2:/usr/bin/ksh
```

If you do not delete the system IDs, they will be all distributed by NIS+ except for the root ID.

Most of the NIS+ commands are located in the /usr/lib/nis directory, so we suggest to add it to the PATH environment variable:

```
export PATH=$PATH:/usr/lib/nis
```

We need to choose the root domain name that must follow the NIS+ naming conventions: it must be made at least by two words divided by a dot. In our example we choose the ibm.com name.

The following command creates the root master. The -r option indicates that a root master server should be set up and the -d option provides the domain name. Note that a trailing dot is added to the domain name.

```
# nisserver -r -d ibm.com.
This script sets up this machine "rootmaster" as an NIS+
root master server for domain ibm.com..

Domain name               : ibm.com.
NIS+ group                : admin.ibm.com.
NIS (YP) compatibility     : OFF
Security Level               : 2=DES

Is this information correct? (type 'y' to accept, 'n' to change)
y

This script will set up your machine as a root master server for
domain ibm.com. without NIS compatibility at security level 2.

Use "nisclient -r" to restore your current network service environment.
Do you want to continue? (type 'y' to continue, 'n' to exit this script)
y

setting up domain information "ibm.com."...

0513-044 The keyserv Subsystem was requested to stop.
0513-059 The keyserv Subsystem has been started. Subsystem PID is 8824.
0513-004 The Subsystem or Group, rpc.nisd, is currently inoperative.
0513-004 The Subsystem or Group, rpc.nispasswdd, is currently inoperative.
0513-004 The Subsystem or Group, nis_cachemgr, is currently inoperative.
running nisinit ...
This machine is in the ibm.com. NIS+ domain.
Setting up root server ...
All done.

starting root server at security level 0 to create credentials...
0513-059 The rpc.nisd Subsystem has been started. Subsystem PID is 10392.

running nissetup to create standard directories and tables ...
org_dir.ibm.com. created
groups_dir.ibm.com. created
passwd.org_dir.ibm.com. created
group.org_dir.ibm.com. created
auto_master.org_dir.ibm.com. created
auto_home.org_dir.ibm.com. created
bootparams.org_dir.ibm.com. created
cred.org_dir.ibm.com. created
ethers.org_dir.ibm.com. created
hosts.org_dir.ibm.com. created
mail_aliases.org_dir.ibm.com. created
sendmailvars.org_dir.ibm.com. created
netmasks.org_dir.ibm.com. created
netgroup.org_dir.ibm.com. created
networks.org_dir.ibm.com. created
protocols.org_dir.ibm.com. created
rpc.org_dir.ibm.com. created
services.org_dir.ibm.com. created
timezone.org_dir.ibm.com. created
client_info.org_dir.ibm.com. created

adding credential for rootmaster.ibm.com...
Please enter the login password for root:
```

```
Wrote secret key into /etc/.rootkey

creating NIS+ administration group: admin.ibm.com. ...
adding principal rootmaster.ibm.com. to admin.ibm.com. ...

restarting NIS+ root master server at security level 2 ...
0513-044 The rpc.nisd Subsystem was requested to stop.
0513-059 The rpc.nisd Subsystem has been started. Subsystem PID is 10394.
starting NIS+ password daemon ...
0513-059 The rpc.nispasswdd Subsystem has been started. Subsystem PID is 16672.
starting NIS+ cache manager ...
0513-059 The nis_cachemgr Subsystem has been started. Subsystem PID is 6214.

This system is now configured as a root server for domain ibm.com.
You can now populate the standard NIS+ tables by using the
nispopulate script or /usr/lib/nis/nisaddent command.
```

The NIS+ master is now active and all system tables have been created, but they are all empty. We now fill them using the following `nispopulate` command. The -F flag specifies that the tables take their data from files, the -p flag specifies the directory containing the source file and -d flag specifies the domain name (note again the trailing dot):

```
# nispopulate -F -p /nis+files -d ibm.com.

NIS+ domain name          : ibm.com.
Directory Path                    : /nis+files

Is this information correct? (type 'y' to accept, 'n' to change)
y

This script will populate the standard NIS+ tables for domain
ibm.com. from the files in /nis+files:auto_master auto_home ethers group hosts networks
passwd protocols services rpc netmasks bootparams netgroup aliases timezone

**WARNING:  Interrupting this script after choosing to continue
may leave the tables only partially populated.  This script does
not do any automatic recovery or cleanup.
Do you want to continue? (type 'y' to continue, 'n' to exit this script)
y

**WARNING: file /nis+files/auto_master does not exist!
  auto_master table will not be loaded.

        ... warnings skipped ...

populating group table from file /nis+files/group...
group table done.

populating hosts table from file /nis+files/hosts...
hosts table done.

Populating the NIS+ credential table for domain ibm.com.
from hosts table.

dumping hosts table...
loading credential table...


The credential table for domain ibm.com. has been populated.
The password used will be nisplus.


**WARNING: file /nis+files/networks does not exist!
  networks table will not be loaded.

populating passwd table from file /nis+files/passwd...
passwd table done.

Populating the NIS+ credential table for domain ibm.com.
from passwd table.

dumping passwd table...
loading credential table...


The credential table for domain ibm.com. has been populated.

The password used will be nisplus.


**WARNING: file /nis+files/protocols does not exist!
  protocols table will not be loaded.

        ... warnings skipped ...

nisplus


Credentials have been added for the entries in the
hosts and passwdtable(s).  Each entry was given a default
network password (also known as a Secure-RPC password).
This password is:

                  nisplus

Use this password when the nisclient script requests the
network password.


nispopulate failed to populate the following tables:
 auto_master auto_home ethers networks protocols services rpc netmasks bootparams
netgroup mail_aliases timezone
```

Several warnings are received because we did not put all the files that the `nispopulate` command expected in our working directory. This causes no harm: the corresponding tables are not changed.

It is important to notice that the network credential have been added to the cred table and a default password has been defined: nisplus. This is the password that must be provided the first time the secure-RPC password will be asked.

The root master server is now active and ready to serve clients. The rootmaster machine has also been configured to be an NIS+ client and we can look at the tables using the `niscat` command:

```
# niscat hosts.org_dir
```

```
loopback loopback 127.0.0.1  loopback (lo0) name/address
loopback localhost 127.0.0.1
rootmaster rootmaster 9.3.240.42
rootmaster 433c 9.3.240.42
austinmaster austinmaster 9.3.240.40
austinmaster 433a 9.3.240.40
nisclient nisclient 9.3.240.41
nisclient 433b 9.3.240.41
#
# niscat passwd.org_dir
nisplus1:4fger.AuQ4UdY:206:1::/home/nisplus1:/usr/bin/ksh:10816::::::
nisplus2:NfYgWSDhVGa0A:207:1::/home/nisplus2:/usr/bin/ksh:10816::::::
```

Note the name of the tables. The name passwd.org_dir provides the relative position of the table in the NIS+ hierarchy. The full name of the table is password.org_dir.ibm.com. and the trailing ibm.com. was implicit because the rootmaster machine is in the ibm.com domain.

The Clients

Now we are ready to configure the other two machines as NIS+ clients. The procedure is the same for all clients and makes use of the `nisclient` command on the clients. The -i flag initializes the client, the -d flag provides the domain name (note the trailing dot) and -h flag provides the name of the rootmaster machine. Do not use the IP address of the NIS+ machine, only the name: the command will ask the IP address.

```
# nisclient -i -d ibm.com. -h rootmaster

CAUTION: When setting up the replicas, -h option is required and the nisserver -R command
has to run on master server not on the client machine.
Once initialization is done, you will need to reboot your
machine.

Do you want to continue? (type ’y’ to continue, ’n’ to exit this script)
y

0513-004 The Subsystem or Group, nis_cachemgr, is currently inoperative.
Type server rootmaster’s IP address:
9.3.240.42
setting up domain information "ibm.com."...


At the prompt below, type the network password (also known
as the Secure-RPC password) that you obtained either
from your administrator or from running the nispopulate script.
Please enter the Secure-RPC password for root:
Please enter the login password for root:

Your network password has been changed to your login one.
Your network and login passwords are now the same.

0513-059 The nis_cachemgr Subsystem has been started. Subsystem PID is 12174.
Client initialization completed!!
Please reboot your machine for changes to take effect.
```

The command first asks for the secure-RPC password for root. The default password has been previously set by the `nispopulate` command to be *nisplus*.

Then it asks for the local root password, that is the password you used to login into the client machine. When the command ends, the secure-RPC password of the client workstation is set to be the same of your root login password.

Once the command finishes, it is mandatory to reboot the machine.

The clients are now able to contact the root master and to use the NIS+ tables. However, the user authentication do not make use of NIS+ by default, so you are not able to login as nisplus1, for example. You can login as nisplus1 on the rootmaster machine because the user is defined in the local /etc/passwd file, but neither rootmaster makes use of NIS+ for authentication.

The /etc/security/user file defines how users are authenticated. The default stanza in the file contains the default user definitions. The primary authentication method (auth1) is defined to be SYSTEM which is defined as compact. This means that /etc/passwd or NIS is used. If you want to activate NIS+ authentication for every user, you have to change the SYSTEM attribute to NISPLUS:

```
default:
        ...
        SYSTEM = "NISPLUS"
```

You can also decide that selected users do not make use of NIS+ authentication. Just create a stanza in /etc/security/user with the name of the user and define the SYSTEM variable to be, for example, compact.

```
notnis:
        SYSTEM = "compat"
```

Now you can login on nisclient machine, for example, using nisplus1 user.

Clients contact the NIS+ server to access the tables. You can see the name of the server the client uses issuing the `nisshowcache` command:

```
$ /usr/lib/nis/nisshowcache

Cold Start directory:
        dir_name:'ibm.com.'

NisSharedCacheEntry[1]:
        dir_name:'org_dir.ibm.com.'

Active servers:
        rootmaster.ibm.com. loopback udp 9.3.240.42.130.47 remote
        rootmaster.ibm.com. inet tcp 9.3.240.42.129.151 local
        rootmaster.ibm.com. inet udp 9.3.240.42.130.47 local
```

If the server is not responding, the request has a timeout of 90 seconds then fails. In order to have better availability, a replica server should be defined. In our environment we can define austinmaster as a replica server of rootmaster.

The Replica Server

The first action is to configure the replica server as a client, like in the preceding example. Then the `rpc.nisd` daemon must be started on the replica server:

```
# startsrc -s rpc.nisd
0513-059 The rpc.nisd Subsystem has been started. Subsystem PID is 12132.
```

You have to be sure that the daemon will be started at the next boot, uncommenting the following lines in /etc/rc.nfs

```
if [ -x /usr/sbin/rpc.nisd ]; then
        start rpc.nisd /usr/sbin/rpc.nisd
fi
```

Finally, you have to configure *on the master server* (in our case rootmaster) the new replica server using the `nisserver` command. The -R flag indicates that a replica server should be set up, the -d flag specifies the domain name (note the trailing dot) and -h specifies the replica name.

```
# nisserver -R -d ibm.com. -h austinmaster
This script sets up an NIS+ replica server for domain
ibm.com.

Domain name               : ibm.com.
NIS+ server           : austinmaster

Is this information correct? (type 'y' to accept, 'n' to change)
y

This script will set up machine "austinmaster" as an NIS+
replica server for domain ibm.com. without NIS compatibility.
The NIS+ server daemon, rpc.nisd, must be running on without
with the proper options to serve this domain.

Do you want to continue? (type 'y' to continue, 'n' to exit this script)
y

Added "austinmaster.ibm.com." to group "admin.ibm.com.".

The system austinmaster is now configured as a replica server for ibm.com
domain
```

A New User

New users may be added using the `nismkuser` command on the master server. The syntax of the command is the same of `mkuser` command, except for the fact you can also define the password. For example, we can add the federico NIS+ user with password qwerty in the following way:

```
# nismkuser password=qwerty federico
```

The user is added to the master server. The update is scheduled for the replica server. If you want the update to be started immediately, you may force it issuing the `nisping` command:

```
# nisping -C ibm.com.
Checkpointing replicas serving directory ibm.com. :
Master server is rootmaster.ibm.com.
        Last update occurred at Fri Aug 13 11:26:22 1999

Master server is rootmaster.ibm.com.
checkpoint scheduled on rootmaster.ibm.com..
Replica server is austinmaster.ibm.com.
checkpoint scheduled on austinmaster.ibm.com..
```

You can now login as federico in all machines in the domain that have activated NIS+ user authentication. The user's directory has not been created by the `nismkuser` command and you will be located in /home/guest until you create the home environment for the user, for example using AutoFS.

8.22 TCP Selective Acknowledgments (4.3.3)

Although TCP has a mechanism to recover from loss of single segment in a window, in case of multiple segments loss, the sender generally has to retransmit not only the lost segments but also segments received normally. This causes significant impact on throughput especially when the network is unreliable or congested. To avoid these unnecessary retransmission, another mechanism called selective acknowledgments (SACK) which is defined in RFC 2018 is implemented on AIX4.3.3. By enabling SACK, the sender only has to retransmit segments that have been really lost during the transmission.

8.22.1 How SACK Works

Figure 87, and the discussion that follows, illustrates how SACK works:

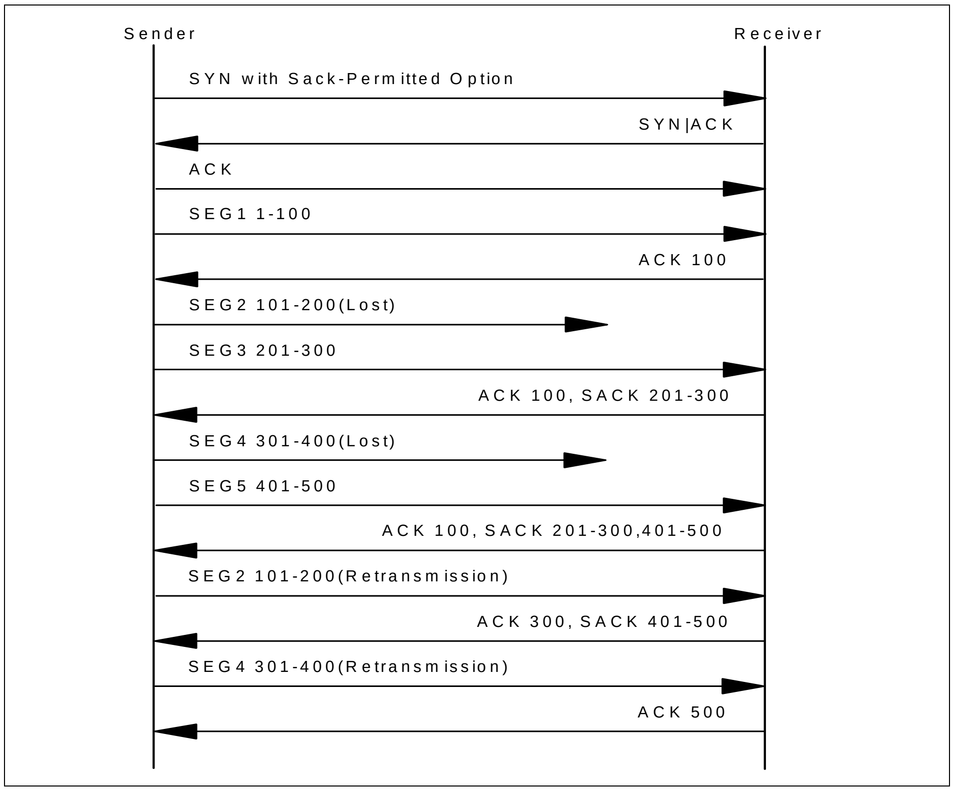


Figure 87. SACK Message Flow

When establishing the TCP connection, one or both side of the connection (in this case the sender only) sends SYN packet with the Sack-Permitted Option. The option means that the sender is ready to receive TCP packets with SACK Option in the TCP header if the receiver side has implemented the option. The receiver can also send SYN|ACK packets with Sack-Permitted Option. In this case, the sender also sends its packets with SACK information.

After the connection establishment, the sender begins data transmission. In the previous figure, segment 1 is successfully sent but segment 2 is lost. Because the TCP window on sender’s side is not filled up yet, the sender sends segment 3 and it is received. At this point, the ACK value sent from receiver is 100 (segment 1 only) because of the cumulative nature of TCP acknowledgments. But the packet also includes SACK information indicating that the data within segment 3 (201-300) has been received. Then the sender knows that it only has to retransmit segment 2 after retransmission timeout.

The above situation (one lost segment) is also resolved by TCP's fast retransmit/fast recovery mechanism. But in case of multiple drop, sender has no way to know the fact and has to wait for retransmission timer to expire. For example, when segment 4 is lost and segment 5 is received, the ACK value sent back to the sender is still 100. The sender will wait for the retransmission timer to expire and retransmit segment 2, 3, 4, and 5. With SACK the sender knows that only segment 2 and 4 have been lost and then retransmits these segments.

8.22.2 New no Option

A new option `sack` is added to `no` command. The option may have one of the following two values:

0	SACK is not enabled. This is the default.
1	SACK is enabled.

8.23 DHCP/DDNS Upgrade (4.3.3)

A dynamic host configuration protocol (DHCP) server assigns an IP address and a set of operating TCP/IP parameters to a DHCP client based on a set of defining tokens from the client. Sometimes this process includes updating the dynamic DNS capable server with any new IP address and name mappings.

Enhancements are added to the DHCP server program. They include:

- Dynamic DNS update daemon
- User defined extension

8.23.1 Dynamic DNS Update Daemon

On previous versions of AIX, the DHCP server updates the Dynamic DNS server by executing system() call to fork() and exec() the command defined in configuration file. Typically the command is `/usr/sbin/dhcpaction` or `/usr/sbin/dhcpaction8`. On AIX 4.3.3, the command `/usr/sbin/nsupdate` is upgraded to run as a daemon when it updates the dynamic DNS server. The daemon is an alternative to `/usr/sbin/dhcpaction` or `/usr/sbin/dhcpaction8` commands and the DHCP server communicates with the daemon using sockets so that it does not have to execute system() call. This mechanism contributes to a significant performance improvement.

8.23.2 User Defined Extension

AIX 4.3.3's DHCP server provides a way to extend DHCP server using a user defined extension. The extension, also called the DHCP server API, is provided primarily for integrating DHCP server with IP management software packages. The DHCP server API specifies the routines, their prototypes, and the desired function of each symbol to be exported by the user defined module. The user defined routines are called by the DHCP server at appropriate times to give the IP management software packages better accounting information.

For more information about user defined extension, see *DHCP Server API* in *AIX Version 4.3 Communication Programming Concepts*, SC23-4124.

8.24 Quality of Service Support (4.3.3)

A new concept for regulating traffic flows called Quality of Service (QoS) is introduced to AIX 4.3.3. The demand for QoS arises from such applications as digital audio/video applications or real-time applications. There are two QoS models currently under standardization by IETF: Integrated Service (IS) and Differentiated Service (DS).

8.24.1 Integrated Service

Integrated Services (IS) is a dynamic resource reservation model for the Internet defined in RFC1633. Hosts use a signaling protocol called resource reservation protocol (RSVP) to dynamically request a specific quality of service from the network. An important characteristic of IS is that this signaling is per-flow and reservations are installed per-hop. This means that a host requests resource reservations per logical connection and resource on each hop (routers and hosts) is reserved. Each router reserves resources to classify packets on multiple flows, schedules output of packets based on classes and so on. This model is well-suited for meeting the dynamically changing needs of applications but it has significant scaling issues such that it cannot be deployed on routers on backbone networks.

8.24.2 Differentiated Service

Differentiated Services defined in RFC 2474 removes the per-flow and per-hop scalability issues, replacing them with a simplified mechanism of classifying packets. Rather than a dynamic signaling approach, DS uses six bits in the IP type of service (TOS) byte to separate packets into classes. The particular bit pattern in the IP TOS byte is called the *DS codepoint* and is used by routers to define the quality of service delivered at that particular hop, in much the same way routers do IP forwarding using routing table lookups. The treatment given to a packet with a particular DS codepoint is called a per-hop behavior (PHB) and is administered independently at each network node. When the effects of these individual, independent PHBs are concatenated, this results in an end-to-end service.

The above two models could be viewed as competing technologies. However, a recent trend sees these two complement each other. IS is likely to be used in stub networks and DS is likely to be used in core networks.

8.24.3 Policy-Based Networking

In order for network equipments that provide QoS features from various vendors to interoperate correctly, it is necessary to standardize the policy scheme for QoS. An Internet Draft, <draft-rajan-policy-qosschema-01.txt> addresses this issue. A policy condition is characteristics of a packet and a policy action is an action the packet receives when it meets a policy condition. A policy condition is defined by five characteristics of a packet. They are Source IP address, source port number, destination IP address, destination port and protocol type (TCP or UDP). A policy action includes:

- Permission (accept or deny)
- Token bucket parameters defining in-profile traffic
- TOS byte value for in-profile traffic
- TOS byte value for out-of-profile traffic

From an administrator’s point of view, a policy is essentially configuration parameters to regulate certain types of traffic flow.

Policy-based networking applies both IS and DS QoS models.

8.24.4 AIX Integration

QoS is implemented on AIX as follows:

QoS Kernel Extension(/usr/lib/drivers/qos)
This kernel extension enables QoS support on AIX host and permits the installation of policies and enforces these installed policies through the use of traffic conditioning and packet marking techniques. It is loaded and unloaded cfgqos and ucfgqos configuration methods.

qosstat Command
The `qosstat` command displays information about the installed QoS policies.

Policy Agent(/usr/sbin/policyd)
This user level daemon provides support for policy management and interfaces with the QoS kernel extension to install, modify, and delete policy

rules. Policy rules may be defined in the local configuration file (/etc/policyd.conf) or retrieved from a central network policy server using LDAP or both. `Policyd` is activated and deactivated through use of SRC.

/etc/policyd.conf

The contents of policyd.conf fall into three categories:

ReadFromDirectory	This statement directs `policyd` to download policies from LDAP server.
ServiceCategories	This statement specifies the type of service that a flow of IP packets should receive as they traverse the network.
ServicePolicyRules	This statement specifies characteristics of IP packets, that are used to match a corresponding service category.

/etc/policyd.conf Example

The following example regulates FTP traffic from host 9.3.240.42 from 8:00 to 20:00.

```
ServiceCategories  restricted
{
   PolicyScope     DataTraffic
   MaxRate         1000
   MinRate         100
   MaxTokenBucket  100
   Priority        1
   OutgoingTOS     00000000
   FlowServiceType Guaranteed
   MaxFlows        10
}

ServicePolicyRules   ftptraffic
{
   PolicyScope             DataTraffic
   Direction               Both
   Permission              Allowed
   ProtocolNumber          6
   TimeOfDayRange          8:00-20:00
   SourceAddressRange      9.3.240.42-9.3.240.42
   DestinationPortRange    20-20
   ServiceReference        restricted
}
```

The following screen output shows that FTP traffic is restricted to 71.85 KB/s even in a LAN environment.

```
# /etc/methods/cfgqos
# startsrc -g qos
0513-059 The rsvpd Subsystem has been started. Subsystem PID is 22856.
0513-059 The policyd Subsystem has been started. Subsystem PID is 5176.
# ftp 9.3.240.41
Connected to 9.3.240.41.
220 433b FTP server (Version 4.1 Tue Apr 13 18:58:50 CDT 1999) ready.
Name (9.3.240.41:root): root
331 Password required for root.
Password:
230 User root logged in.
ftp> put /unix /dev/null
200 PORT command successful.
150 Opening data connection for /dev/null.
226 Transfer complete.
3672616 bytes sent in 49.92 seconds (71.85 Kbytes/s)
local: /unix remote: /dev/null
ftp>
```

At this point, the output of `qosstat` command should look like:

```
Policy Rule Handle 1:
Filter specification for rule index 1:
        protocol:                    TCP
        source IP addr:              9.3.240.42
        destination IP addr:         INADDR_ANY
        source port:                 ANY_PORT
        destination port:            20
Flow Class for rule index 1:
        service class:     Diff-Serv
        peak rate:         100000000 bytes/sec
        average rate:      125000 bytes/sec
        bucket depth:      4096 bytes
        TOS (in profile):  0
        TOS (out profile): 0
Statistics for rule index 1:
        total number of connections:         0
        total bytes transmitted:             3672616
        total packets transmitted:           2530
        total in-profile bytes transmitted:  3672616
        total in-profile packets transmitted: 2530
```

RSVP Agent(/usr/sbin/rsvpd)

This user level daemon implements the RSVP signaling protocol semantics. RSVP is a protocol that transfers QoS and policy control parameters. Note that RSVP transfers these parameters as opaque data. In other words, RSVP dose not understand the meaning of the parameters. A specific request for resource reservation is made through use of RAPI (RSVP API). Rsvpd is activated and deactivated through use of SRC.

/etc/rsvpd.conf

This configuration file defines parameters for `rsvpd`. /etc/rsvpd.conf defines parameters related to entitlement and capacity only.

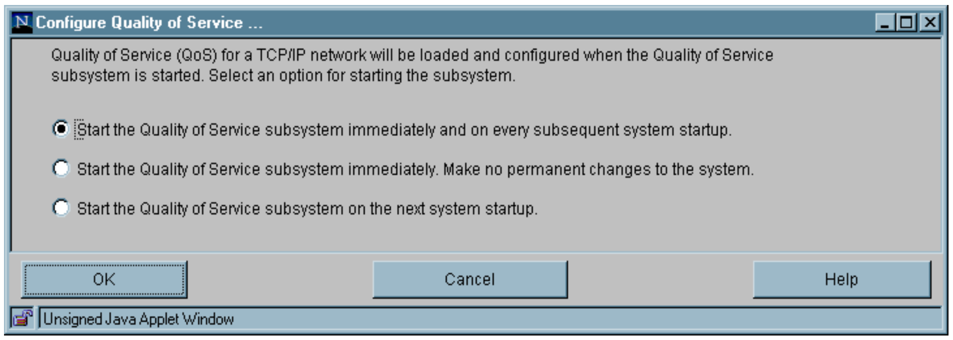
RAPI

RSVP API (RAPI) is an application programming interface for RSVP Applications may use the RAPI APIs to communicate with the RSVP agent, rsvpd. Following is a list of functions included in /usr/lib/librapi.a.

rapi_session()	Creates an API session. It opens a socket connection between calling application and `rsvpd`.
rapi_sender()	Notifies that the calling application intends to send a flow of data for which receivers may make reservations.
rapi_reserve()	Makes resource reservation based on flow specifications, filters and so on.
rapi_release()	Removes the reservation and the API session.

8.24.4.1 Web-Based System Manager Integration

A configuration panel is added to start and stop QoS subsystem from Web-Based System Manager. Figure 88 shows the panel. The panel can be displayed by right-clicking the mouse button on any of four items under **TCP/IP (IPv4 and IPv6)** on **Network** panel.



Quality of Service (QoS) for a TCP/IP network will be loaded and configured when the Quality of Service subsystem is started. Select an option for starting the subsystem.

[✓ Start the Quality of Service subsystem immediately and on every subsequent system startup.]

Start the Quality of Service subsystem immediately. Make no permanent changes to the system.

Start the Quality of Service subsystem on the next system startup.

Figure 88. QoS Configuration Panel on WSM

8.25 Name Resolver Dynamic Loading (4.3.3)

On previous levels of AIX, the name resolver routine's mapping rule only has the options of named/BIND (bind or dns), NIS (nis), and /etc/hosts (local). They are specified in the environment variable NSORDER, Configuration files: /etc/netsvc.conf and /etc/irs.conf. A new option is added to allow users to dynamically load their own resolver routines to be used instead of or in conjunction with named/BIND, NIS, and /etc/hosts.

The format of the user option is:

```
<key>[4/6]
```

where the <key> is the name of the shared object module to be dynamically loaded into the executing process's address space. The number after the key is the address family which can be none, 4 or 6. If no number is specified, the family=AF_UNSPEC; if the number is 4, then the family=AF_INET4; if the number is 6, then the family=AF_INET6. For example, /etc/netsvc.conf can be specified:

```
hosts=local,nis,bind,bos,david4,jason6
```

david4 indicates that gethostbyname() routine will query the *david* resolver module (david.so) searching for an IP version 4 host address.

See sample code in /usr/samples/tcpip/dynload for more information on creating the shared object module.

8.26 Socks Enhancements (4.3.3)

Socks is a widely used method to access from clients inside firewall to Internet. It is considered to be a de facto standard and the protocol between socks clients and servers is defined in RFC 1928 *Socks Protocol Version 5*. Typically socks server software runs on a firewall machine or on a stand alone machine in demilitarized zone. Compared with Web proxy servers that typically relay HTTP, FTP, and SSL traffic, socks is more generalized and has no limitation on its supported protocols (although they depend on socks server implementation) and generally requires a few, sometimes no changes on client programs' side.

On AIX 4.3.3, two socks features, socks client library and automatic socksification are added.

8.26.1 Socks Library

Socks library provides a standardized method to develop socksified applications on AIX. It implements the Socks Version5 protocol. The following is the list of functions provided:

socks5tcp_connect()	Creates a TCP connection to a destination server through socks server.
socks5tcp_bind()	Requests socks server to create a listening socket that receives connection from a destination server. This call is necessary when the destination server initiates another connection as in the case of FTP data connection.
socks5tcp_accept()	Requests socks server to block waiting for an incoming connection from the destination server.
socks5udp_associate()	Creates a UDP association on socks server keyed to a destination server. This call is an alternative to pseudo connect() in normal UDP programming without using socks and must be called before any socks5udp_sendto() calls.
socks5udp_sendto()	Sends a UDP packet to the destination server through a socks server.
socks5_getserv()	Retrieves the address of the socks server to use when connecting or sending to a destination server.

Note: Only the 00 authentication method (no authentication) as described in RFC 1928 is implemented.

8.26.2 Automatic Socksification

By using AIX 4.3.3’s automatic socksification feature, application programs can connect to destination servers through socks servers without changing their code. This feature is implemented by modifying connect() system call to be socks aware. Administrators must setup /etc/socks5c.conf file and applications must set SOCKS5C_CONFIG environment variable to use that feature.

/etc/socks5c.conf

This configuration file defines mappings between destination servers and a socks server. Each line consists of the following syntax:

```
<pattern> <server>[:<port>]
```

where <pattern> can be:

- Network address with prefix length such as 1.1.1.1/24
- Subnet address
- IP v6 address
- Partial subnet name
- Network name such as ibm.com
- Full domain name

for example:

```
10.10.10.10/32 10.10.10.11:1080
```

means that all connection to 10.10.10.10 and subsequent data traffic flow through socks server 10.10.10.11 using port 1080.

SOCKS5C_CONFIG
If set without value, automatic socksification is enabled and the configuration file /etc/socks5c.conf is used. If set with a value pointing to another configuration file, the file is used. If not set, automatic socksification is not enabled.

Example
The following is an example of a telnet session from 9.3.240.1 through socks server 9.53.195.78 to 9.116.209.221.

The configuration file /etc/socks5c.conf should look like this:

```
9.116.209.221/32 9.53.195.78:1080
```

```
# telnet 9.116.209.221
Trying...
telnet: connect: A remote host did not respond within the timeout period.
# export SOCKS5C_CONFIG=/etc/socks5c.conf
# telnet 9.116.209.221
Trying...
Connected to 9.116.209.221.
Escape character is '^]'.


 telnet (flaulen)





AIX Version 4
(C) Copyrights by IBM and by others 1982, 1996.
login:
```

8.27 Sendmail Enhancements (4.3.3)

The latest release of sendmail, Version 8.9.3 is ported to AIX 4.3.3. Compared with Version 8.8.8 which is shipped with AIX 4.3.2, Version 8.9.3 contains bug fixes, several new configuration options, and anti-spam features.

8.27.1 What is Spam?

e-mail spam is copies of similar messages sent to people who do not want the message. It typically contains commercial advertisements or malicious messages. It wastes CPU time and disk space of mail servers, bandwidth of network lines, and readers' time and money. Sometimes spammers use mail servers to relay their messages to numerous destination addresses. They send only one message to mail severs and mail servers broadcast the message to the destination addresses listed in RCPT TO: field. This method is called *third party relay*. In some cases, spammers hide their original e-mail addresses or original host addresses by changing some fields in their messages to nonexistent ones or those of other organizations especially when the messages are malicious.

8.27.2 Anti-Spam Features

On AIX 4.3.3, anti-spam features are not activated by default. So it is necessary to generate or modify sendmail.cf file with anti-spam features. Some of the features are:

- FEATURE(relay_hosts_only)

  By enabling this feature, sendmail only accept exact hosts listed in configuration database for relaying.

- FEATURE(rbl)

  This feature directs sendmail to reject hosts found in the real-time blackhole list. By default sendmail tries to look up maps.vix.com.

- FEATURE(accept_unqualified_senders)

  When this feature is enabled, sendmail accepts mail which sender addresses without domain names (*name* instead of *name@domain*). By default sendmail does not accept this mail (on AIX 4.3.3 this feature is enabled).

- FEATURE(accept_unresolvable_domain)

  When this feature is enabled, sendmail accepts mail that has an unresolveable return address. By default, sendmail does not accept this mail (on AIX 4.3.3 this feature is enabled).

- FEATURE(promiscuous_relay)

  This feature allows replaying from any site to any site. Activating this feature is not encouraged (on AIX 4.3.3 this feature is enabled).

8.27.3 Generating Customized sendmail.cf

In this section, we provide an example that demonstrates a simple anti-replay feature of sendmail 8.9.3. Assume the configuration in Figure 89.

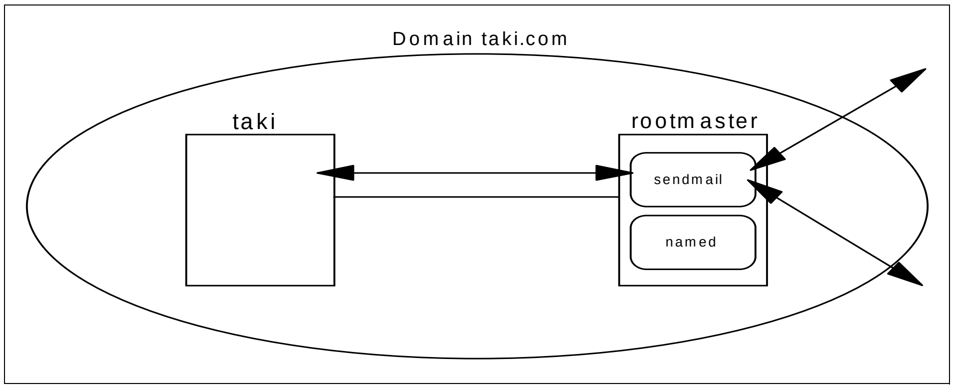


Figure 89. Example Sendmail Configuration

The hosts taki and rootmaster belong to domain taki.com. The host rootmaster is the sendmail hub server for the domain as well as the master

DNS server. The host taki is a sendmail *nullclient*. It sends all the mail to rootmaster for relaying.

To generate customized sendmail.cf on rootmaster, it is necessary to customize /usr/samples/tcpip/sendmail/cf/aix433.mc file. You can change, add or modify the file to suite your site’s requirements. In our example, the customized file looks like the following example:

```
divert(0)dnl
OSTYPE(aix433)dnl
FEATURE(allmasquerade)
DOMAIN(generic)dnl
MAILER(local)dnl
MAILER(smtp)dnl
MAILER(uucp)
```

Next, issue following commands to generate customized sendmail.cf:

```
# cd /usr/samples/tcpip/sendmail/cf
# m4 ../m4/cf.m4 aix433.mc > sendmail.cf
```

Create /etc/sendmail.cw file that contains fully qualified name for rootmaster:

```
rootmaster.taki.com
```

Create /etc/mail/relay-domains file that contains domains to which you allow for relaying:

```
good.domain.com
```

Then, restart sendmail:

```
# startsrc -s sendmail -a"-bd -q30m"
```

On taki, create and send an e-mail to someone in good.domain.com

```
$ comp
To: mario@good.domain.com
cc:
Subject: Hello
--------
This is a test message.
--------

What now? send
```

You should see following syslog output on rootmaster.

```
Aug 31 10:59:17 rootmaster sendmail[22440]: KAA22440: from user
<atsushi@rootmaster.taki.com>: size is 304, class is 0, priority is 30304,
and nrcpts=1, message id is <199908311601.LAA05372@taki.taki.com>,
protocol=ESMTP, relay=taki.taki.com [9.3.240.41]
Aug 31 10:59:17 rootmaster sendmail[22440]: KAA22440:
to=<mario@good.domain.com>, delay=00:00:00, mailer=esmtp, stat=queued
```

These messages mean the mail for mario@good.domain.com has been queued for delivery. On the other hand, when the destination address is mario@bad.domain.com, you will see following messages:

```
Aug 31 11:03:03 rootmaster sendmail[22994]: LAA22994: ruleset=check_rcpt,
arg1=<mario@bad.domain.com>, relay=taki.taki.com [9.3.240.41], reject=550
<mario@bad.domain.com>... Relaying denied
```

See /usr/samples/tcpip/sendmail/README for more information about anti-spam features and general configuration methods.

Note

The original /etc/sendmail.cf file shipped with AIX 4.3.3 has been modified after the system is generated. So it is impossible to generate the file using default aix433.mc file. Save /etc/sendmail.cf to restore the default.

8.27.4 Enhancements Added by IBM

The following enhancements are added by IBM against sendmail 8.9.3.

- Name Resolution Order

  Sendmail uses AIX host resolution ordering mechanism. First, NSORDER environment variable is queried, if it is not defined, sendmail searches /etc/netsvc.conf and then /etc/irs.conf. Sendmail's /etc/services.switch file is searched last. If none of them found or host entry is not defined, then the system wide default ordering is used.

- IPv6 Support

  IPv6 has is supported as a underlying protocol.

8.28 AIX Fast Connect Release 2

AIX Fast Connect for Windows is an optional feature of AIX 4.3. With this feature, AIX servers act as print or file servers for Windows and OS/2 clients. The new release of AIX Fast Connect is shipped with AIX 4.3.3.

8.28.1 AIX Fast Connect Overview

AIX Fast Connect provides print and file services to clients by implementing the Server Message Block (SMB) protocol. SMB uses Network Basic Input/Output System (NetBIOS) over TCP/IP. AIX Fast Connect offers following functions:

- Tight integration with AIX and use of features, such as threads, kernel I/O, file system, and security.
- SMB-based file and printer services. This allows centralized administration, backup, and so forth for common data, such as user data and applications.
- Maintenance and administration is integrated into standard AIX tools.
- Web-Based System Manager (WSM) and System Management.
- Interface Tool (SMIT). It is also possible to use command the line (`net` command).
- IBM World-wide service and support.
- Client authentication can be done on a local machine or with passthrough authentication to the NT domain. Guest users are supported.
- Supports resource browsing protocol. The server can announce its resources, but it cannot be a master browser.
- Supports WINS client and proxy and implements NetBIOS Name Server (NBNS).
- Traces and log capabilities.
- Documentation is an integral part of the AIX Version 4.3 Base Documentation*.*
- Support for Unicode.
- AIX long file name to DOS file name mapping support.

8.28.2 AIX Fast Connect Release 2 New Function

AIX Fast Connect Release 2 provides following new functions:

- DCE DFS access

  AIX Fast Connect Release 2 supports DCE DFS integration. It uses DCE APIs to authenticate incoming SMB session requests. Once a session is authenticated, Fast Connect acquires login context for the DCE principal. The acquired login context is set in the process servicing the connection. This allows the file system requests to be made under the acquired login context. This feature allows PC clients to access get authenticate using

DCE security server and access DFS under the login context of the PC client logged in as DCE principal.

- CIFS Logon (NetLogon) Support

  Network logon support allows centralizing the user accounts, startup scripts, home directories, and configuration policy of Windows systems participating in a workgroup to a single AIX system running AIX Fast Connect Server. This support does not allow an AIX Fast Connect Server to act as a Windows NT Domain Controller.

- AIX Access Control Lists (ACLs) Support

  AIX Fast Connect release 2 supports AIX ACLs. Access to files and directories are in effect when a PC client is accessing the files. Access controls are enforced by making file system calls from a process that has setuid which is same as the user who is accessing the file system resource.

- HACMP Support

  AIX Fast Connect release 2 included enhancements to support HACMP mutual takeover and rotating standby modes in addition to support of standby mode. To support mutual takeover, and rotating standby modes, the following enhancements is added to AIX Fast Connect.

  - Multiple server names (aliases) support
  - Ability to add and delete resources while server is stopped or running.

- National Language Support

  AIX Fast Connect release 2 supports following locales:

  Ca_ES, De_DE, En_US, Es_ES, Fr_FR, It_IT, Ja_JP, ZH_CN, Zh_CN, Zh_TW, ca_ES, cs_CZ, de_DE, en_US, es_ES, fr_FR, hu_HU, it_IT, ja_JP, ko_KR, pl_PL, pt_BR, ru_RU, sk_SK, zh_CN, zh_TW

8.28.3 CIFS Logon Client Considerations

AIX Fast Connect currently supports Windows 95, Windows 98, and NT 4.0 as CIFS logon clients. The following considerations apply to each clients:

- Windows NT 4.0 clients

  Windows NT 4.0 clients must use IBM Primary Logon Client for NT. It can be downloaded from:

  `http://techsupport.services.ibm.com/asd-bin/doc/en_us/winntcl2/f-feat.htm`

- Windows 95/98 clients

They may use IBM Network Client for Windows 95 or use their native logon capability. IBM Network Client for Windows 95 can be downloaded from:

```
http://techsupport.services.ibm.com/asd-bin/doc/en_us/win95cl/f-feat.htm
```

- Plain text password support

  To enable plain text password support, you must change Windows clients' registry entry.

For more information about CIFS logon feature, see /etc/cifs/README file.

8.28.4 Sample Configuration

In this section, we provide a simple configuration example. The configuration scenario is as follows:

- The AIX server name is rootmaster and its domain name is dom433.
- The Windows 95 client is authenticated by the AIX server using AIX Fast Connect's CIFS logon capability.
- The client is automatically assigned the home directory on the server to local H: drive.

Server Setup

Edit /etc/cifs/cifsConfig file and change following fields:

- domainname

  Domain name for this server. In our example, it is dom433
- encrypt_passwords

  Set the value 0 to allow AIX-based user authentication.

Start the AIX Fast Connect server using following command:

```
# net start
```

Add an AIX Fast Connect user using following command. The user should already exist as an AIX user and the password should be the same as the user's AIX login password.

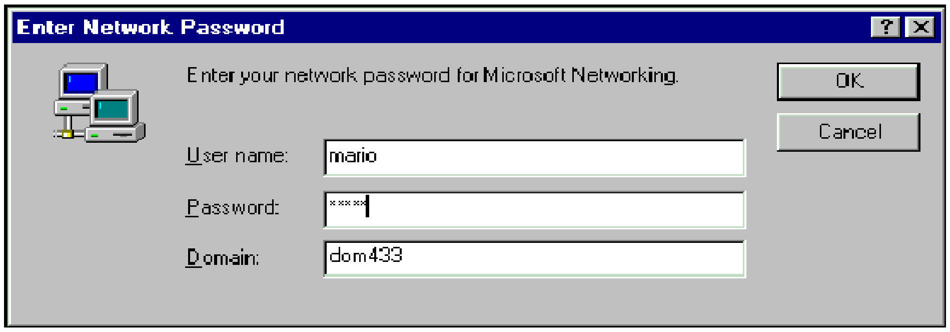
```
# net user mario -p /add /active:yes
Enter mario's password:xxxxx
Command completed successfully.
```

Change the default startup script to assign the user's home directory on the AIX server to local user's H: drive. Edit /var/cisfs/netlogon/startup.bat to include following lines:

```
net use H: \\rootmaster\home
echo "H: is now mapped to \\rootmaster\home
```

Client Logon Procedure

Boot the client machine. When the logon panel is displayed, change the domain name to dom433 and enter the assigned user name and password. Then press OK.


Enter Network Password

Enter your network password for Microsoft Networking.

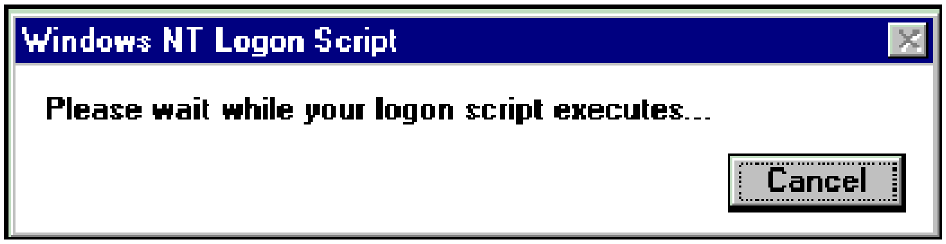
User name: mario

Password: *****

Domain: dom433

Figure 90. Window95 Logon Panel

Then you see a dialog as shown in Figure 91.


Windows NT Logon Script

Please wait while your logon script executes...

Figure 91. Windows NT Logon Script Message Dialog

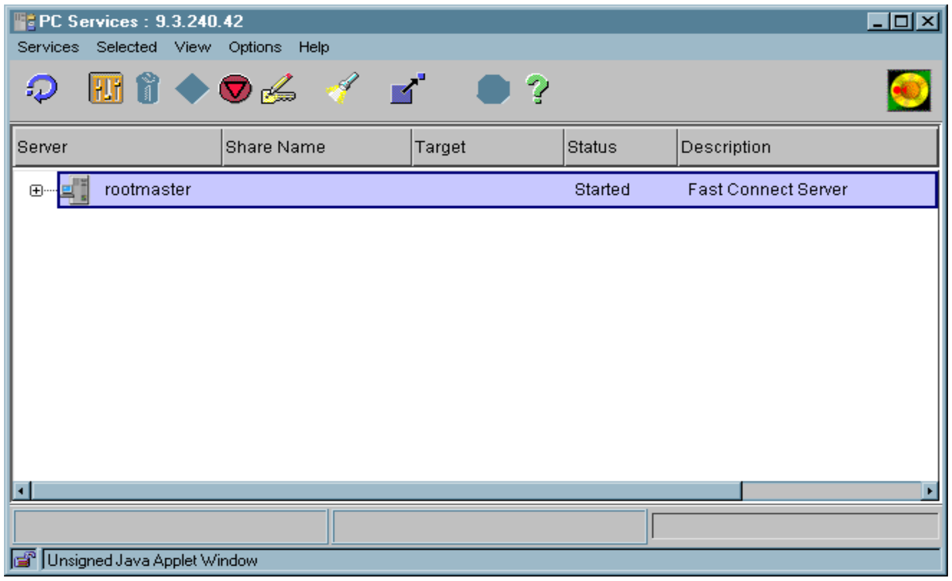
Use `net use` command to confirm H: drive is assigned.

```
c:\windows>net use
Status          Local name      Remote name
-------------------------------------------------------------------------------
OK              H:              \\ROOTMASTER\HOME
The command was completed successfully.
```

8.28.5 AIX Fast Connect WSM integration

AIX Fast Connect Release 2 is integrated into the Web-Based System Manager. You can start or stop the server, add or delete file or print shares, and perform user administration tasks through WSM. The following panel

(Figure 92) is launched by double-clicking **PC Services** icons on the Web-Based System Manager main panel.



Server	Share Name	Target	Status	Description
rootmaster			Started	Fast Connect Server

Figure 92. AIX Fast Connect WSM Panel

8.29 Cisco EtherChannel Support (4.3.3)

Etherchannel is an aggregation technology that allows you to combine multiple Ethernet adapters together to form a larger pipe. Etherchannel allows a server-to-switch connection throughput between 40 and 800 Mbps depending on the settings for the adapters, and thus helps address what may be a throughput bottleneck for you. This throughput would be an aggregate over many connections to different machines.

Documentation Location

Currently, the documentation for this new AIX 4.3.3 feature is in the file /usr/lpp/bos/README

8.29.1 How Etherchannel Works

The Etherchannel should look exactly like an Ethernet adapter to the upper layers. Any upper layer (IP, SNA, DLPI, to name a few) that can connect to an

Ethernet adapter through network services should work over an Etherchannel without any code changes.

The Etherchannel works by having all of the device drivers that are a part of the channel connected to the same switch, so long as it is a switch that supports Cisco’s Etherchannel. Traffic sent to the channel is sent on the network over one of the devices that are a part of the channel. Which one is used should not make a difference as to whether the packet makes it to its destination. Hence, the adapter to use is decided by hashing the lower bits of destination MAC address of the outgoing packet for non-IP traffic. For IP traffic, the lower bits of destination IP address are used to locate the adapter. Hashing of the addresses provides for the traffic between a particular source and destination to use the same interface or adapter. This is to avoid packets reaching the destination out of order as some higher layer protocols have problems handling out of order packets.

Likewise any packet received by one of the drivers should be sent to the user regardless of which device it was received on. It has one MAC address, and for IP it has one IP address.

When an adapter of an Etherchannel is down, traffic is re-routed through one of the other adapters and this is transparent to the upper layers. For the incoming traffic, packets can be received over any of the adapters.

Etherchannel provides a scalable server-to-switch bandwidth without having to move to new technology like Gigabit Ethernet.

Etherchannel operates at layer two below the protocol stack. The Etherchannel is implemented as a kernel extension. It is a pseudo device that attaches itself to the network services (of CDLI) such as the other real Ethernet device drivers.

The existing Ethernet adapters can be used to form a channel without any changes to the existing device drivers. The Etherchannel driver does not maintain any statistics. It uses the statistics maintained by the adapters.

8.29.2 Configuration example

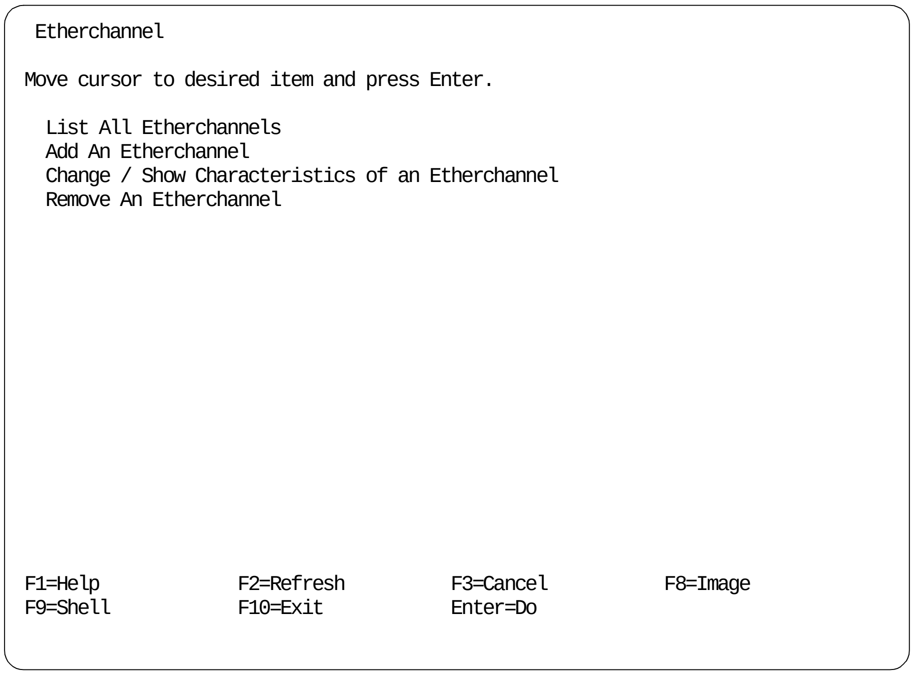
The Etherchannel can be configured with up to four Ethernet adapters (for example, ent0, ent2, ent3, and ent5). By default, all the adapters of the Etherchannel would be assigned the MAC address of the first adapter you select for the channel. You also have the option to set a MAC address for all the adapters. This is be done using the use_alt_addr and alt_addr attributes. You would configure an Etherchannel after the Ethernet device drivers are configured, but before any users of those devices are configured.

Supported Hardware

Any Ethernet adapter, except Gigabit Ethernet adapters, can be used for Etherchannel so long as it can be connected to the Switch that supports Etherchannel, such as Cisco's Catalyst 5505 Switch.

The software to provide the Etherchannel function is shipped as part of the devices.common.IBM.ethernet.rte fileset. The sequence to configure an Etherchannel, using SMIT, is as follows:

1. Go to the new Etherchannel SMIT screen by executing the `smitty` command, then selecting:
   1. **Devices**, then
   2. **Communications**, then
   3. **Etherchannel**, to get a screen similar to:

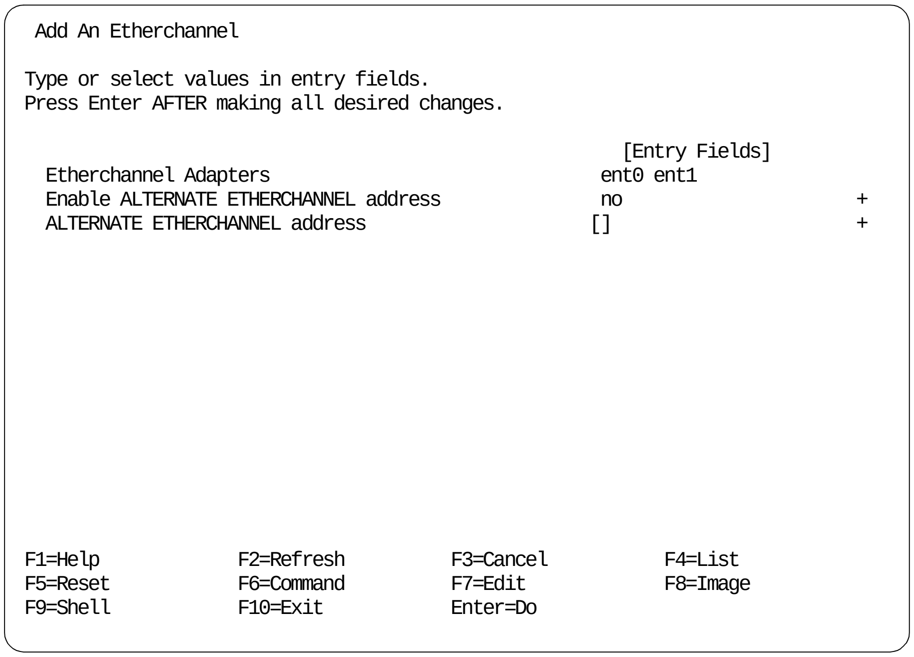


```
  Etherchannel

Move cursor to desired item and press Enter.

  List All Etherchannels
  Add An Etherchannel
  Change / Show Characteristics of an Etherchannel
  Remove An Etherchannel



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

2. Select **Add An Etherchannel.**

   This will prompt you to select an Ethernet adapter from the list of available Ethernet adapters. You need to select two to four adapters from this list to form your channel. In this example, only two adapters are being used to show you the configuration process.

1. For example, select **ent0** to obtain a screen similar to:



```
 Add An Etherchannel

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

                                                      [Entry Fields]
  Etherchannel Adapters                             ent0 ent1
  Enable ALTERNATE ETHERCHANNEL address             no                    +
  ALTERNATE ETHERCHANNEL address                   []                     +










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

From this screen the following command is executed:

```
mkdev -c ch -s EN -t ech -a adapter_names='ent0,ent1'
```

The output screen should be similar to:

```
                                   COMMAND STATUS



                  Command: OK                    stdout: yes
                 stderr: no


                  Before command completion, additional instructions may appear below.

                 ent5 Available
```

You can use the command `entstat -d` (which executes /usr/sbin/entstat.ethchan) to display the individual Ethernet adapter statistics

8.29.3 Performance Considerations

The following needs to be kept in mind when looking for performance improvements using Etherchannel:

- For clients connected across a router connected to the switch, even for IP traffic the throughput would be limited by the router-to-switch connection even though the server-to-switch throughput might show an improvement using the Etherchannel.
- A single FTP download to a client, for example, will not be completed in one fourth the time when using an Etherchannel with four adapters.
- Even the server-to-switch throughput increase depends on how much parallelism the protocol layer (including the interface layer) allows. If there is no parallel traffic in multiple adapters, then the throughput could be slower than if only one adapter was used instead of the Etherchannel. This is because the packet has to go through an additional layer (for example, the Etherchannel driver) before the packet goes out the wire.

8.30 ATM Enhancements (4.3.3)

Asynchronous Transfer Mode (ATM) networks provide a high bandwidth network infrastructure. However, to help protect the investment of existing Local Area Network (LAN) clients, such as those that use traditional Ethernet connections, ATM offers LAN Emulation, or LANE clients, so that an ATM adapter can provide a traditional network interface.

There are many sources of ATM information, such as The ATM Forum at the Internet site: `http://www.atmforum.com`. One important starting point is to review the AIX documentation which has been updated to describe two major enhancements in AIX. The two main articles you need to read are:

TCP/IP Network Adapter Cards, in the book *System Management Guide: Communications and Networks,* SC23-4127

Forum-Compliant ATM LAN Emulation Device Driver, in the book *AIX Version 4.3 Kernel Extensions and Device Support Programming Concepts,* SC23-4125

Another similar source of useful information is the file /usr/lpp/bos.atm/README that is shipped with AIX.

8.30.1 Multiprotocol over ATM

AIX Version 4.3.3 provides improved management and performance of an ATM LAN Emulation network. Device specific configurations are minimized with auto-discovery and device discovery protocol, while data paths are reduced from many hops between routers to a single hop between end clients. In AIX Version 4.3.3, Multiprotocol Over ATM (MPOA)supports both:

- Standard Ethernet
- IEEE 802.3 Ethernet

This enhancement adds the ATM Forum Technical Committee’s Multiprotocol Over ATM Version 1.0 to AIX, which is described in specification AF-MPOA-0087.000. You can read this specification from the Internet site:

`http://www.atmforum.com/atmforum/specs/approved.html`

(You may also want to refer to the LANE-2 specification, AF-LANE-0084.000.) The AIX version 4.3.3 implementation is based on that for the IBM 8210 Multi protocol Switched Services (MSS) product. You can find information about the 8210 if you go to the IBM Internet home page at `http://www.ibm.com/` and search for 8210.

8.30.1.1 Behavior and Implementation

To understand what MPOA does for you, the example from the AIX documentation is reproduced in Figure 93 on page 500, along with a discussion related to it.

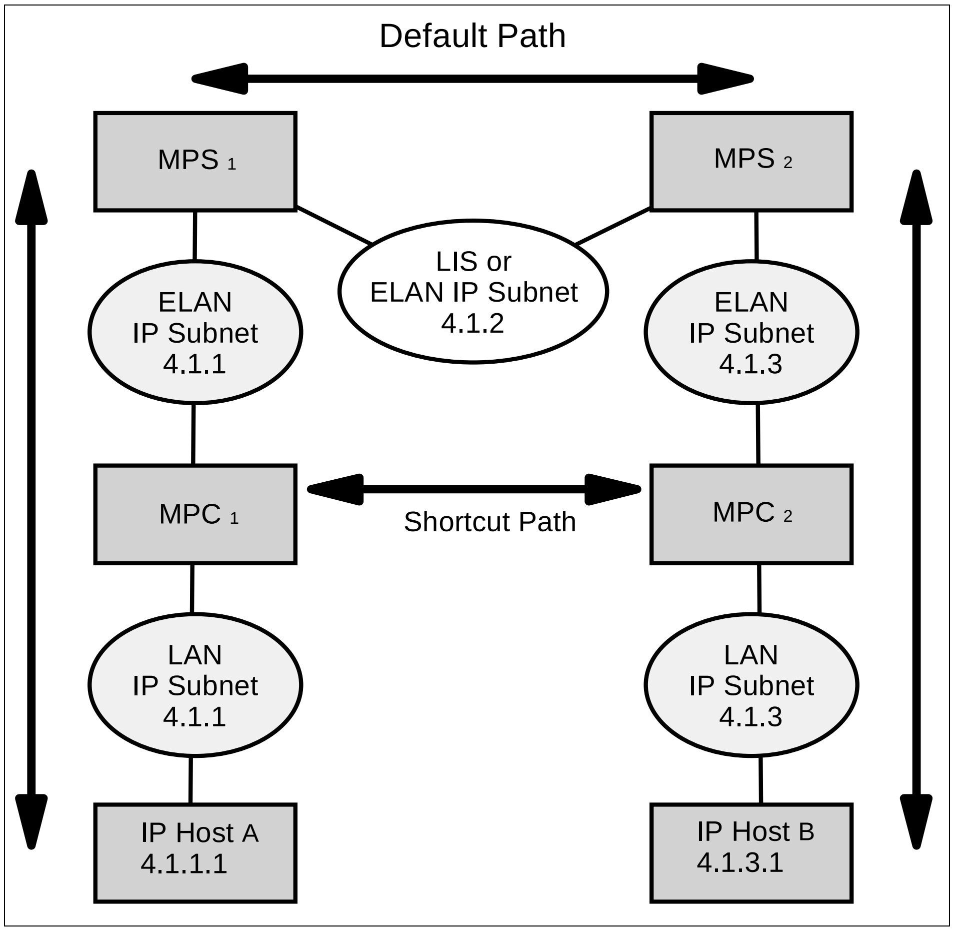


Figure 93. An Example of a Network That Benefits From MPOA

In Figure 93 on page 500, the devices on the Emulated LAN (ELAN) network can be described as edge devices, since they are an example of a device that can bridge packets between one or more LAN interfaces, such as Host A, and one or more LANE clients. The MPOA enhancement allows these edge devices to perform internetwork layer forwarding and establish direct communications without requiring that the LANE edge devices be full function routers.

So when using IP, packets from IP host A would usually have to travel through the 4.1.2 subnet to be routed through to IP host B. However, using MPOA, the 4.1.2 subnet can be bypassed. The MPC is the MPOA client that performs internetwork layer forwarding. The MPC obtains its forwarding information from the MPOA Server (MPS). The MPC detects the flow of packets being forwarded over an ELAN to a router that contains an MPS. When it recognizes a flow that could benefit from a shortcut by a bypass, such as from Host A to B, it uses a Next Hop Resolution Protocol (NHRP)-based query-response protocol to request the information required to establish a shortcut to the destination. If a shortcut is available, the MPC caches the information, sets up a shortcut VCC, and forwards frames for the destination over the shortcut that bypasses the router.

This MPOA mechanism thus separates internetwork layer route calculation and forwarding; the main benefits of this are:

- The MPC edge device does not have to be as complex since it does not do any internetwork layer route calculation.
- There are less devices doing the route calculation, so it is a more scalable and manageable environment.
- It allows efficient inter-subnet communication.

The MPC support is within the ATM LANE load module, that is the file /usr/lib/drivers/atmle_dd, since the two client types work hand in hand. In contrast, however, the MPC acts as a separate device from the normal LECs for simpler distinction and control by you. Therefore information about LEC and MPC is stored separately in the ODM, differentiated by the attribute client_type.

8.30.1.2 Configuration

This enhancement has been included in the bos.atm.atmle fileset. The LANE and MPOA objects are separate devices. The device name prefix for an MPOA Client device is mpc (for example, mpc0, mpc1). Configuration advice is provided in the AIX Documentation states that no configuration entries are required for the MPOA Client. Ease of use default values are provided for each of the attributes which are derived from ATM Forum recommendations. However, the following points should still be noted:

- Any operator configured elan_name should match exactly what is expected at the LECS/LES server when attempting to join an ELAN
- mpoa_enabled specifies whether Forum MPOA and LANE-2 functions should be enabled for this LE Client. The default is No (LANE-1).
- Seven new ODM attributes have been added for MPOA, but they all have default values derived from ATM Forum recommendations. Of these seven, note that:

- auto_cfg specifies whether the MPOA Client is to be automatically configured using the LANE Configuration Server (LECS), and its default value is no.

8.30.1.3 Important MPOA Issues

The following is a list of some issues to be aware of when you use the current implementation of MPOA on AIX:

- IP is the only protocol stack supported. That is, SNA, IPX, NetBIOS directly over ATM are not supported.
- IP fragmentation is not supported
- MPC to NHRP client shortcuts are not supported.
- MPC to LANE client shortcuts are not supported.
- A single MPC will support the entire set of LECs and ATM drivers within a node. Additional MPCs cannot be used, since the extra complexity of doing this outweighs any gain in performance.
- A single unique selector byte is used by the MPC for its local ATM address to make all SVCs associated with MPC unique. The current AIX LANE implementation uses 0xFF for MPCs selector byte. This, however, ends up actually taking two slots away from the LECs and one slot away from C/IP, based on the way the MPC is configured. In general, you can have up to 256 LECs, or 1 MPC and up to 254 LECs.
- Only one LEC is designated as the primary LEC to provide configuration for the MPC using Lan Emulation Configuration Server (LECS). This gives you more control and consistency on what LECS is used, but may also restrict the total number of LECSs available for that function.
- The maximum number of Point-to-Multipoint Virtual Channel Connections (VCCs) is increased from 16 to 64 to allow the additional control and BUS VCs.
- The LEC will automatically register its MPC with the LES after JOIN procedures have completed. A retry timer is used to insure that an unanswered registration is reregistered. This time out value is initially set to the configured LANE control timer value, and is doubled (up to 16 times) for each retry.

8.30.2 Classical IP and ARP over ATM

The previous design based on Internet Engineering Task Force RFC1577 has been upgraded to support RFC2225. The 2225 client eliminates the single point of failure associated with the 1577 clients' ARP services. The ARP server is the host that converts an IP address to an ATM address. This server function and this 225 client enhancement is described in the article TCP/IP Network Adapter Cards on page 499.

The 2225 clients have the ability to switch to backup ARP servers when the current ARP server fails. When the primary comes back up, the 2225 clients switch back to the primary. So the 2225 clients can maintain continuous connectivity in an LIS in case of ARP server failures. Additionally, in case the network has distributed ARP servers, load balancing is possible by making groups of 2225 clients talk to different primary servers.

The 2225 client, unlike the 1577 client, is responsible for initiating the registration process. The 1577 server is responsible for the registration procedure which is done by sending INARP requests. The 2225 client initiates the registration process with an ARP request with the source and target protocol address being the same, that is the client IP address. The new way for the 2225 client to register to the ARP server is that you should put the same source address on the source and target fields. Note do not use the ARP server's address as target for this particular case. The 15 minute re-registration time is cannot be changed.

Chapter 9. Graphical Environment Enhancements

Prior to the release of AIX Version 4.3, IBM shipped X11 Release 5 and additional backward-compatibility libraries plus Motif Version 1.2. AIX Version 4.3 provides graphics updates to two existing products. The first update, Release 6 of the X Window System (X11R6), allows developers to use new extensions and enhancements to existing extensions. The second update, MotifNext or Motif 2.1, provides a range of new function including thread safe libraries and 64-bit enablement.

9.1 X Window System Architecture Review

The X Window System architecture consists of three basic components: The client, protocol, and server. These components will be the subject of the following sections.

9.1.1 Client

The X Client is a software application that requests services from another application called the server, possibly across a network. A typical example would be a stock control application that used the services of an X Station to display output and accept user input from a keyboard or other input device. It is normally written in a high-level language, commonly C, and is linked with one or more of the X libraries. The instructions that the application developer uses to interact with the server are very low-level and allow them to open a connection with the server, create and manipulate windows, draw elementary shapes (lines, rectangles, circles, characters, and so on), and handle events, such as mouse movements or keystrokes. To increase productivity, toolkits have been developed that perform complex operations by calling several elementary subroutines. The most elaborate toolkits, such as Athena or Motif, also contain widgets. Widgets are program objects with a predefined appearance and behavior, such as a menu push button. The standard toolkits provided with AIX are:

Xt	The basic X toolkit with hundreds of standard functions, also called intrinsics.
Xaw	The Athena Widget set from the X Consortium.
Xmu	Miscellaneous utilities.
Xext	The extension library.
Xi	The input extension library. It manages the peripherals, that is, keyboard and mouse.

Xm	The Motif library of widgets with a 3D-look.

9.1.2 Protocol

The protocol is a formal description of the conversation that is carried out between the client and the server. The server and clients exchange information using messages. These messages are contained in packets that can be transported over a network if necessary. There are four packet types:

Request	Used by the client to ask the server to perform an action.
Reply	Used by the server to answer the client after a request has been received.
Event	Used by the server to inform the client that something has happened that requires action by the client.
Error	Used to indicate that something went wrong.

9.1.3 Server

The server is a program that runs on the workstation in which a graphical display is attached. The server program, called X, consists of two parts. One part is device-independent (dix) and interprets requests from Xlib, schedules client activity, manages the return of events and input to the Xlib library, and performs other generic actions. The second part is device-dependent (ddx) and renders the 2D graphic operations defined by the X Window System for the specific display adapter. The loadddx graphic adapter interface (GAI) load module implements this interface. The server modules have strictly-defined function and peripheral support, and the only way to increase the capability of the X Server is through extensions. Extensions are new modules that can be loaded into the server and used to perform actions that the basic modules are incapable of.

For example, the core server (the core server is the name given to the server without any extensions) only supports the keyboard and mouse as peripherals. If you intend to use other peripherals such as a spaceball, dials, or scanner, you need to load an extension that is capable of handling these devices.

All of the extensions that are capable of being loaded will be discussed in the following section.

9.2 X Window System Release 6

The sixth release of the X Window System has been ported to AIX Version 4.3 from software provided by the X Consortium. The X Consortium is an independent, non-profit corporation, the successor to the MIT X Consortium, which was part of the MIT Laboratory for Computer Science. The actual source for the port was X11 Release 6.2, which is a proper subset of X11 R6.3 produced at the request of the OSF Common Desktop Environment (CDE) program. It was produced by the X Consortium and is being released by OSF simultaneously with CDE 2.1. Release 6.2 contains only the print extension and the Xlib implementation of vertical writing and user-defined character support. R6.3 is an update to R6.1. It is compatible with R6 and R6.1 at the source and protocol levels in all respects, and binaries are upward-compatible.

This section describes changes in X11 since Release 5. Release 6.2 contains new function in many areas. In addition, many errors have been corrected. Except where noted, all libraries, protocols, and servers are upward-compatible with Release 5. That is, R5 clients and applications should continue to work with R6 libraries and servers.

The following are new X Consortium standards in Release 6. Each is described in its own section below:

- X11 Security
- X Image Extension
- Inter-Client Communications Conventions Manual (update)
- Inter-Client Exchange Protocol
- Inter-Client Exchange Library
- X Session Management Protocol
- X Session Management Library
- Input Method Protocol
- X Logical Font Descriptions (update)
- SYNC Extension
- XTEST Extension
- BIG-REQUESTS Extension
- XC-MISC Extension
- X Keyboard Extension (XKB)

- RECORD Extension
- Double Buffer Extension (DBE)
- ICE X Rendezvous
- Print Extension
- Xlib Vertical Writing and User-Defined Characters

9.2.1 X11 Security

The X11R6 implementation provides five access control mechanisms, of which the two listed below are supported in AIX 4.3:

Host Access
Any client on a host in the host access control list is allowed access to the X server. The access control list is changed with the `xhost` command.

MIT-MAGIC-COOKIE-1
A 128-bit plain-text cookie is provided by the client with the connection setup information. The xdm program automatically configures the X server and client for each new login session.

9.2.2 X Image Extension

The sample implementation in Release 6 is a complete implementation of the full XIE 5.0 protocol except for the following techniques that are excluded from the sample implementation:

- ColorAlloc: Match, Requantize
- Convolve: Replicate
- Decode: JPEG (lossless)
- Encode: JPEG (lossless)
- Geometry: AntialiasByArea, AntialiasByLowpass

9.2.3 Inter-Client Communications Conventions Manual

X11 Release 6 includes Version 2.0 of the Inter-Client Communications Conventions Manual (ICCCM). This version contains a large number of changes and clarifications in the areas of window management, selections, session management, and resource sharing.

9.2.3.1 Window Management

The circumstances under which the window manager is required to send synthetic ConfigureNotify events have been clarified to ensure that any

ConfigureWindow request issued by the client will result in a ConfigureNotify event, either from the server or from the window manager. It also added advice about how a client should inspect events to minimize the number of situations where it is necessary to use the TranslateCoordinates request.

The window_gravity field of WM_NORMAL_HINTS has a new value, StaticGravity, which specifies that the window manager should not shift the client windows location when re-parenting the window.

The base size in the WM_NORMAL_HINTS property is now to be included in the aspect-ratio calculation.

The WM_STATE property now has a formal definition (it was previously only suggested).

9.2.3.2 Selections

The CLIENT_WINDOW, LENGTH, and MULTIPLE targets have been clarified. A number of new targets for Encapsulated PostScript and the Apple Macintosh PICT-structured graphics format have been added. Also, a new selection property type C_STRING (a string of nonzero bytes) was defined. This is in contrast to the STRING type, which excludes many control characters. A selection requester can now pass parameters with the request. Another new facility is *manager selections*. This use of the selection mechanism is not to transfer data, but to allow clients known as managers to provide services to other clients. Version 2.0 also specifies that window managers should hold a manager selection. Currently, the only service defined for window managers is to report the ICCCM version number in which the window manager complies. Now that this facility is in place, additional services can be added in the future.

9.2.3.3 Resource Sharing

A prominent new addition in Version 2.0 is the ability of clients to take control of colormap installation under certain circumstances. Earlier versions of the ICCCM specified that the window manager had exclusive control over colormap installation. This proved to be inconvenient for certain situations, such as when a client has the server grabbed. Version 2.0 allows clients to install colormaps after having informed the window manager. Clients must hold a pointer grab for the entire time they are doing their own colormap installation.

Version 2.0 also clarifies a number of rules about how clients can exchange resources. These rules are important when a client places a resource ID into a hints property or passes a resource ID through the selection mechanism.

9.2.3.4 Session Management

Some of the properties in Section 5 of ICCCM 1.1 are now obsolete, and new properties for session management have been defined.

9.2.4 ICE (Inter-Client Exchange)

ICE provides a common framework in which to build protocols. It supplies authentication, byte order negotiation, version negotiation, and error reporting conventions. It supports multiplexing multiple protocols over a single transport connection. ICElib provides a common interface to these mechanisms so that protocol implementors do not need to reinvent them. An `iceauth` program was written to manipulate an ICE authority file. It is very similar to the `xauth` program.

9.2.5 SM (Session Management)

The X Session Management Protocol (XSMP) provides a uniform mechanism for users to save and restore their sessions using the services of a network-based session manager. It is built on ICE. SMlib is the C interface to the protocol. There is also support for XSMP in Xt. A simple session manager, `xsm`, is included.

A new protocol, `rstart`, greatly simplifies the task of starting applications on remote machines. It is built upon already existing remote execution protocols such as `rsh`. The most important feature that it adds is the ability to pass environment variables and authentication data to the applications being started.

9.2.6 X Logical Font Description

The X Logical Font Description has been enhanced to include general 2D linear transformations, character set subsets, and support for polymorphic fonts.

9.2.7 SYNC Extension

The SYNC extension lets clients synchronize through the X server. This eliminates the network delays and the differences in synchronization primitives between operating systems. The extension provides a general counter resource, allowing clients to alter the value of a counter and block their execution until a counter reaches a specific threshold. For example, two clients share a counter initialized to zero; one client can draw some graphics and then increment the counter. The other client can block until the counter reaches a value of one and then draw some additional graphics.

9.2.8 XC-MISC Extension

A new extension, XC-MISC, allows clients to retrieve ID ranges from the server. Xlib handles this automatically. This is useful for long-running applications that use many IDs over their lifetime.

9.2.9 BIG-REQUESTS Extension

The standard X protocol only allows requests up to 2^{18} bytes long. A new protocol extension, BIG-REQUESTS, has been added that allows a client to extend the length field in protocol requests to be a 32-bit value. This is useful for PEX and other extensions that transmit complex information to the server. The BIG-REQUESTS have already been implemented by IBM as an extension to X11 Release 5.

9.2.10 Double Buffer Extension (DBE)

The Double Buffer Extension (DBE) provides a standard way to utilize double-buffering, allowing flicker-free animation.

The older Multi-Buffering extension is not linked in to the X server by default. It will move to unsupported status at the next release.

9.2.11 X Keyboard Extension

Prior to the introduction of the X Keyboard Extension (XKB) in X11R6, the core X protocol was used for keyboard interaction. The core X protocol has a number of limitations that make it difficult, or impossible, to properly support many common varieties of keyboard behavior. The X Keyboard Extension provides capabilities that are lacking, or cumbersome, in the core X protocol.

9.2.11.1 XKB Keyboard Extension Support for Keyboards

The X Keyboard Extension makes it possible to clearly and explicitly specify most aspects of keyboard behavior on a per-key basis. It adds the notion of a keyboard group to the global keyboard state and provides mechanisms to closely track the logical and physical state of the keyboard. For keyboard-control clients, XKB provides descriptions and symbolic names for many aspects of keyboard appearance and behavior.

In addition, the X Keyboard Extension includes additional keyboard controls designed to make keyboards more accessible to people with mobility impairments.

9.2.11.2 XKB Extension Components

The XKB extension is composed of two parts:

- A server extension.
- A client-side Xlibrary extension.

The server portion of the XKB extension encompasses a database of named keyboard components, in unspecified format, that may be used to configure a keyboard. Internally, the server maintains a *keyboard description* that includes the keyboard state and configuration (mapping). Keyboard is defined as the logical keyboard device, which includes not only the physical keys, but also potentially a set of up to 32 indicators (usually LEDs) and bells.

The *keyboard description* is a composite of several different data structures, each of which may be manipulated separately. The individual components are shown in Figure 94.

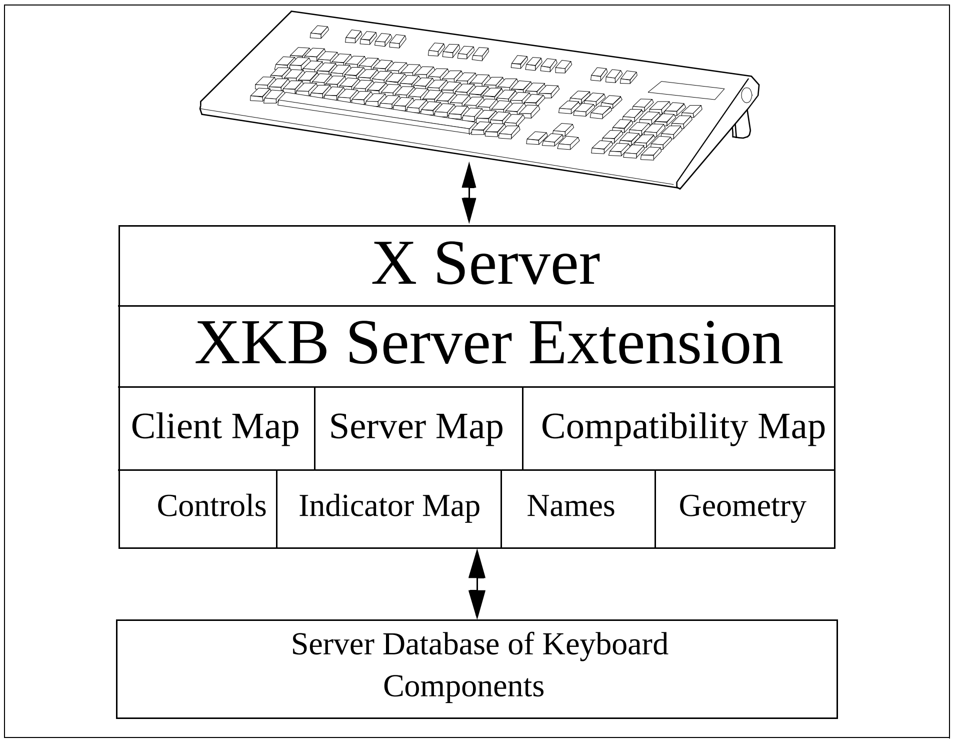


Figure 94. XKB Server Extension

Client Map The key mapping information needed to convert arbitrary keycodes to symbols.

Server Map	The key mapping information categorizing keys by function (which keys are modifiers, how keys behave, and so on).
Controls	Client configured quantities affecting how the keyboard behaves, such as repeat behavior and modifications for people with movement impairments.
Indicators	The mapping of behavior to indicators.
Geometry	A complete description of the physical keyboard layout sufficient to draw a representation of the keyboard.
Names	A mapping of names to various aspects of the keyboard, such as individual virtual modifiers, indicators, and bells.
Compatibility Map	The definition of how to map core protocol keyboard state to XKB keyboard state.

A client application interrogates and manipulates the keyboard by reading and writing portions of the server description for the keyboard. In a typical sequence, a client would fetch the current information it is interested in, modify it, and write it back. If a client wants to track some portion of the keyboard state, it typically maintains a local copy of the portion of the server keyboard description working with the items of interest and updates this local copy from events describing state transitions that are sent by the server. A client may request the server to reconfigure the keyboard either by sending explicit reconfiguration instructions or by telling it to load a new configuration from its database of named components. Partial reconfiguration and incremental reconfiguration are both supported

9.2.11.3 Groups and Shift Levels

The graphic characters, or control functions, that can be accessed by one key are logically arranged in groups and levels. For example, the Radio Group is a set of keys whose behavior simulates a set of radio buttons. Once a key in a radio group is pressed, it stays logically pressed until another key in the group is pressed, at which point the previously-pressed key is logically released. Consequently, at most one key in a radio group can be logically depressed at one time. A radio group is defined by a radio group index, an optional name, and by assigning each key in the radio group XKBKB_RadioGroup behavior and the radio group index.

9.2.11.4 Client Types

The X11R6 specification differentiates between three different classes of client applications as shown in Figure 95.

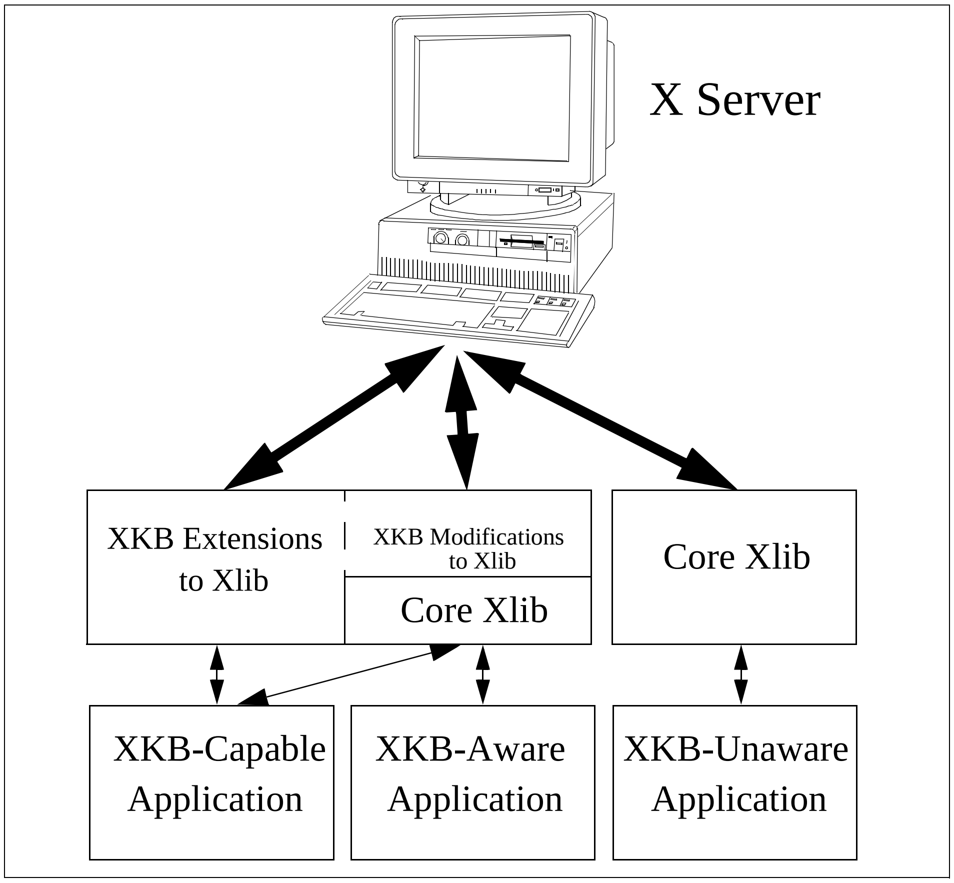


Figure 95. Types of XKB Clients

- XKB-aware applications. These applications make specific use of XKB function and Application Programming Interfaces (APIs) not present in the core protocol.
- XKB-capable applications. These applications do not use XKB extended function and APIs directly. However, they are linked with a version of Xlib that includes XKB and indirectly benefit from some of XKB's features.
- XKB-unaware applications. These applications do not make use of XKB extended function or APIs and require XKB's function to be mapped to core Xlib function to operate properly.

9.2.11.5 Protocol Errors

The XKB extension adds a single protocol error, BadKeyboard, to the core protocol errorset. Table 53 lists the protocol errors that can be generated and their causes.

Table 53. XKB Protocol Errors

Protocol Error	Error Cause
BadAccess	The XKB extension has not been properly initialized.
BadKeyboard	The device specified was not a valid core or input extension device.
BadImplementatio n	Incorrect reply from server.
BadAlloc	Unable to allocate storage.
BadMatch	A compatible version of XKB was not available in the server or an argument has correct type and range but is otherwise in error.
BadValue	An argument is out of range.
BadAtom	A name is neither a valid Atom nor None.
BadDevice	Device, Feedback Class, or Feedback ID is in error.

9.2.11.6 Extension Library Functions

The X Keyboard Extension replaces the core protocol definition of a keyboard with a more comprehensive one. The X Keyboard Extension library interfaces are included in Xlib.

Xlib detects the presence of the X Keyboard server extension and uses XKB protocol to replace some standard X library functions related to the keyboard. If an application uses only standard X library functions to examine the keyboard or process key events, it should not need to be modified when linked with an X library containing the X keyboard extension. All of the keyboard-related X library functions have been modified to automatically use XKB protocol when the server extension is present.

The XKB extension adds library interfaces to allow a client application to directly manipulate the new capabilities.

9.2.11.7 XKB Client Applications

With XKB, there are several new core clients:

- xkbcomp
- xkbevd

- `kbevd`
- `xkbprint`

9.2.12 X Record Extension

The purpose of this extension is to support the recording and reporting of all core X protocol and arbitrary X extension protocol.

The extension is used to record the core X protocol and arbitrary X extension protocol entirely within the X server itself. When the extension has been requested to record specific protocol by one or more recording clients, the protocol data is formatted and returned to the recording clients. The extension provides a mechanism for capturing all events, including input device events that do not go to any clients.

9.2.13 ICE X Rendezvous

The Inter-Client Exchange protocol (ICE) that became a standard in X11R6 specifies a generic communication framework for data exchange between arbitrary clients. The ICE protocol itself does not specify the manner in which two clients interested in communicating using ICE are made aware of each other's existence.

The ICE X Rendezvous protocol is one standard protocol by which two clients who have connections to a common X server can rendezvous. This new protocol is included in the ICE Protocol Specification document.

9.2.14 Print Extension

The print extension supports output to hardcopy devices using the core X drawing requests. The print extension adds requests for job and page control and defines how specific printer attributes are communicated between the server and printing clients. Printer attribute specifications are modeled after the ISO 10175 specification.

An X Client that wants to produce hardcopy output will typically open a second connection to an X print server, produce a print job, and then close the print server connection. The print server may be the same process as the display server (the term video server is sometimes used), although the implementation provided in R6.2 does not completely support video and print servers in the same binary.

9.2.14.1 Running an X Print Server

The print server is simply an X server with the print extension and special DDX implementations. The X print server is started like any other X server.

The following command line is an example for use with a typical configuration:

```
# Xprt :1 -ac
```

The options used in the example are:

:1 On a host that is running a video display server you will need to specify a different display from the default.

-ac Disable access control since no simple mechanism for sharing keys is provided.

The X print server also supports the following additional options:

-XpFile
Points to the directory containing the print server configuration files.

XPCONFIGDIR
Environment variable specifying alternative location of the print server configuration files.

The print server is configured through a directory of configuration files that define printer model types and instances of printer models. An example configuration tree is provided.

By convention, clients locate the print server using the environment variable XPRINTER. The syntax of XPRINTER is an augmented DISPLAY:

```
printerName@host:display
```

where `printerName` is one of the printer instances listed in the print server configuration files. The use of XPRINTER, and its syntax, is an application convention only; there is nothing in the supplied libraries that uses (or parses) this environment variable.

9.2.15 Xlib Vertical Writing and User-Defined Characters

The Xlib output method implementation has been enhanced to support the XOM drawing direction XOMOrientation_TTB_RTL. Vertical writing information, and other locale specific information, is read from the file <XLocaleDir>/%L/XLC_LOCALE.

The X[mb|wc]TextEscapement functions now return the text escapement in pixels for the vertical or horizontal direction depending on the XNOrientation XOCValue.

The X[mb|wc]DrawString functions will now render a character string in the vertical or horizontal direction depending on the XNOrientation XOCValue.

The Xlib NLS database implementation has been enhanced to support extended segments used for interchanging nonstandard code sets. Support has been added for control sequences and encoding names used in extended segments and conversion of glyph indexes when interchanging data in extended segments.

9.2.16 Xlib Library

Xlib now supports multithreaded access to a single display connection. Xlib functions lock the display structure, causing other threads calling Xlib functions to be suspended until the first thread unlocks. Threads inside Xlib waiting to read to or write from the X server do not keep the display locked, so, for example, a thread hanging on XNextEvent() will not prevent other threads from doing output to the server.

The display and GC structures have been made opaque to normal application code; references to private fields will get compiler errors. You can work around some of these by compiling with -DXLIB_ILLEGAL_ACCESS, but it is better to fix the offending code.

The Xlib implementation has been changed to support a form of asynchronous replies, meaning that a request can be sent to the server, and then other requests can be generated without waiting for the first reply to come back. This is used to an advantage in two new functions, XInternAtoms() and XGetAtomNames(), which reduce what would otherwise require multiple round trips to the server down to a single round trip. It is also used in some existing functions, such as XGetWindowAttributes(), to reduce two round trips to just one.

Support for using poll(), rather than select(), is implemented, selected by the HasPoll configuration option.

Table 54 provides the Xlib functions that are new in Release 6.

Table 54. New X11R6 Xlib Functions

_XAllocTemp()	_XFreeTemp()
IsPrivateKeypadKey()	XAddConnectionWatch()
XAllocIDs()	XCloseOM()
XCreateOC()	XContextualDrawing()

XConvertCase()	XDestroyOC()
XDirectionalDependentDrawing()	XDisplayOfOM()
XESetBeforeFlush()	XExtendedMaxRequestSize()
XGetAtomNames()	XGetOCValues()
XGetOMValues()	XInitImage()
XInitThreads()	XInternalConnectionNumbers()
XInternAtoms()	XLocaleOfOM()
XLockDisplay()	XOMOfOC()
XOpenOM()	XProcessInternalConnection()
XReadBitmapFileData()	XRegisterIMInstantiateCallback()
XRemoveConnectionWatch()	XSetIMValues()
XSetOCValues()	XSetOMValues()
XUnlockDisplay()	XUnregisterIMInstantiateCallback()

9.2.17 Xt Toolkit

Support has been added for participation in session management, with call-backs to application function in response to messages from the session manager. In addition, the entire library is now threadsafe. This allows one thread at a time to enter the library and also protects global data from concurrent use. Support is also provided for registering event handlers for events generated by X protocol extensions and for dispatching those events to the appropriate widget.

A mechanism has also been added for dispatching events for non-widget drawings (such as pixmaps used within a widget) to a widget.

Two new widget methods, for example, allocation and deallocation, allow widgets to be treated as C++ objects in a C++ environment.

A new interface allows bundled changes to the managed set of children of a *Composite*, reducing the visual disruption of multiple changes to geometry layout.

Several new resources have been added to shell widgets, making the library compliant with the Release 6 ICCCM. Parameterized targets of selections (new in Release 6) and the MULTIPLE target are supported with new APIs.

The client will be able to register callbacks on a per-display basis for notification of a large variety of operations in the X toolkit. This feature is useful to external agents, such as screen readers.

The file search path syntax has a new %D substitution that inserts the default search path, making it easy to prepend and append to the default search path.

The Xt implementation allows a configuration choice of poll() or select() for I/O multiplexing, selectable at compile time by the HasPoll configuration option.

The Release 6 Xt implementation requires Release 6 Xlib. Specifically, it uses the following new Xlib features: XInternAtoms() instead of multiple XInternAtom() calls where possible, input method support (Xlib internal connections), and tests for the XVisibleHint in the flags of XWMHints.

When linking with Xt, you now need to link with SMlib and ICElib as well. This is automatic if you use the XTOOLLIB make variable or XawClientLibs make variable in your makefiles.

This implementation no longer allows NULL to be passed as the value in the name/value pair in a request to XtGetValues(). The default behavior is to print the following error message and exit:

```
NULL ArgVal In XtGetValues
```

9.2.18 Xaw Toolkit

Some minor code corrections have been integrated. Text and Panner widget translations have been augmented to include keypad cursor keysyms in addition to the normal cursor keysyms.

The Clock, Logo, and Mailbox widgets have moved to their respective applications.

Internationalization support is now included. Xaw uses native widechar support when available; otherwise, it uses the Xlib widechar routines. Per-system specifics are set in XawI18n.h.

9.2.18.1 AsciiText

The name AsciiText is now a misnomer but has been retained for backward-compatibility. A new resource, XtNinternational, has been added. If the value of the XtNinternational resource is False (the default), AsciiSrc and AsciiSink source and sink widgets are created, and the widget behaves as it did for R5. If the value is True, MultiSrc and MultiSink source and sink widgets are created. The MultiSrc widget will connect to an input method server if one is available, or if one is not available, it will use an Xlib internal pseudo-input method that, at a minimum, does compose processing. Application programmers who want to use this feature will need to add a call to XtSetLanguageProc() to their programs.

The symbolic constant FMT8BIT has been changed to XawFmt8Bit to be consistent with the new symbolic constant XawFmtWide. FMT8BIT remains for backwards compatibility; however, its use is discouraged as it will eventually be removed from the implementation.

Two new resources have been added, XtNinternational and XtNfontSet. These resources have been added to the following widgets:

- Command widget
- Label widget
- List widget
- MenuButton widget
- Repeater widget
- SmeBSB widget
- Toggle widget

If XtNinternational is set to True, the widget displays its text using the specified font set.

9.2.19 Header Files

Two new macros are defined in Xos.h, X_GETTIMEOFDAY and strerror.

X_GETTIMEOFDAY is like gettimeofday() but takes one argument on all systems. Strerror is defined only on systems that do not already have it.

A new header file, Xthreads.h, provides a platform-independent interface to threads functions on various systems. When programming, include it instead of the system threads header file. Use the macros defined in it instead of the system threads functions.

Xalloca.h is solely responsible for defining ALLOCATE_LOCAL and DEALLOCATE_LOCAL. You should be able to add or update a platform's support for alloca() by editing this one file instead of finding and changing the multiple definitions that existed previously. Xpoll.h allows more portable, consistent select() and poll() use in the clients, including getting the fd_set properly defined. The servers still use select() on all systems, even those that have poll().

9.2.20 Fonts

There are three new Chinese bdf fonts: gb16fs.bdf, gb16st.bdf, and gb24st.bdf.

The Type 1 fonts contributed by Bitstream, IBM, and Adobe that shipped in /contrib in Release 5 have been moved into the core. Some of the *misc* fonts, mostly in the *Clean* family, have only the ASCII characters but were incorrectly labeled ISO8859-1. These fonts have been renamed to be ISO646.1991-IRV. Aliases have been provided for the Release 5 names.

The 9x15 font has new shapes for some characters. The 6x10 font has the entire ISO 8859-1 character set.

9.2.20.1 Font Library

The Type 1 rasterizer that shipped in /contrib in Release 5 is now part of the core.

There is an option to have the X server request glyphs only as it needs them. The X Server then caches the glyphs for future use.

Aliases in a fonts.alias file can allow one scalable alias name to match all instances of another font. The exclamation (!) character introduces a comment line in fonts.alias files.

A sample font authorization protocol, hp-hostname-1, has been added. It is based on host names and is non-authenticating. The client requesting a font from a font server provides (or passes through from its client) the host name of the ultimate client of the font.

Note: There is no check that this host name is accurate, as this is a sample protocol only.

The Speedo rasterizer can now read fonts with retail encryption. This means that fonts bought over-the-counter at a computer store can be used by the font server and X server.

9.2.20.2 Font Server

The font server has been renamed from `fs` to `xfs` to avoid confusion with an AFS program. The `fsconf` utility is also renamed to `xfsconf`. The default port has changed from 7000 (used by AFS) to 7100 and has been registered with the Internet Assigned Numbers Authority. Until AIX 4.2.1, the default port was

7500 to avoid conflict with the IBM X Station at the time R5 was first shipped on AIX. This is changed to the new X11R6 default port.

You can now connect to `xfs` using the local/transport.

9.2.21 X Input Method

For each of the different locales, there is a single default input method associated with it. The AIX X Window System, prior to the release of X11R6, used the following default X Input Methods (XIM):

Local	A local single byte input method
Thai	A local type Thai input method
X11R6 XIM Protocol	All of server type input methods

As well as these input methods, AIX provides a native input method, AIX IM, with the release of X11R6. The AIX implementation of X11R6 puts the AIX Input Method as the default input method.

9.2.21.1 XIM Module Loader

The introduction of the AIX Input Method as the default input method changes the loading priority of the different input method modules. The new order is show in the Table 55.

Table 55. XIM Module Loading Priorities

Priority	XIM Module	Locales	XMODIFIERS
1	AIX IM	AIX system	Depend on AIX IM
2	XC's Local	XLOCALEDIR	Local, none
3	XC's Thai	th	Don't care
4	XC's Proto	XIM server	Depend on IM servers

The following is a list of AIX XIM module features:

- AIX XIM module has the first loading priority.
- AIX XIM module tries to query specified language and IM modifier (XMODIFIERS) to the AIX Input Method.

9.2.21.2 AIX XIM Interface for Input Method Switching

X Consortium's sample code implements an input method switching mechanism at the top of XIM layer. All X input methods are registered at _XimImSportRec in the imImSw.c file. The _XimOpenIM() function invokes the checkprocessing method of the registered X input method to check that the X input method is supported by the specified locale and IM modifier. If the input method is supported, then the _XimOpenIM() function calls im_open method of the XIM. If it fails to open IM, then the im_free method is called to close it down.

The following section describes the methods provided by AIX's XIM module:

(*checkprocessing)() - _XimCheckIfAIXProcessing
This routine is called from the _XimOpenIM() function to check that a XIM module supports specified IM information (locale and modifiers).

This function returns True with the following conditions:

- If the XMODIFIERS is not specified, and the specified locale is supported by the AIX Input Method.
- If the specified XMODIFIERS and locale combination are supported by the AIX Input Method.

If the XMODIFIERS is specified, a combined AIX IM name is used to check if the specified locale and modifier combination are supported by the AIX IM. To make a combined AIX name, concatenate the IM modifier to the locale name. The AIX IM function accepts a language name that contains the IM modifier @im=.

For example:

```
$LANG = xx_XX

$XMODIFIERS = @im=foo
```

Combined Name = `xx_XX@im=foo`

AIX IM module = `xx_XX@foo`

(*im_open)() - _XimAIXOpenIM
This routine is called from the _XimOpenIM() function when an application calls XOpenIM().

(*im_free)() - _XimAIXIMFree
This routine is called from the im_free() function and is used to close down the input method.

9.2.22 Input Method Protocol

Some languages need complex pre-editing input methods. Such an input method may be implemented separately from applications in a process called an Input Method (IM) Server. The IM Server handles the display of pre-edit text and the user’s input operation. The Input Method (IM) Protocol standardizes the communication between the IM Server and the IM library linked with the application.

The IM Protocol is a completely new protocol based on experience with R5’s sample implementations. The following new features are added beyond the mechanisms in the R5 sample implementations:

- The IM Server can support any of several transports for connection with the IM library.
- Both the IM Server and clients can authenticate each other for security.
- A client can connect to an IM Server without restarting, even if it starts up before the IM Server.
- A client can initiate string conversion to the IM Server for reconversion of text.
- A client can specify some keys as hot keys, which can be used to escape from the normal input method processing regardless of the input method state.

The R6 sample implementation for the internationalization support in Xlib has a new plug-in framework with the capability of loading and switching locale object modules dynamically. For backward compatibility, the R6 sample implementation can support the R5 protocols by switching to IM modules supporting those protocols. In addition, the framework provides the following new functions and mechanisms:

X Locale database format

An X locale database format is defined, and the subset of a user's environment dependent on language is provided as a plain ASCII text file. You can customize the behavior of Xlib without changing Xlib itself.

ANSI C and non-ANSI C bindings

The common set of methods and structures are defined that bind the X locale to the system locales within libc, and a framework for implementing this common set under non-ANSI C base system is provided.

Converters

The sample implementation has a mechanism to support various encodings by plug-in converters and provides the following converters:

Lightweight converter for C and ISO 8859

Generic converter

High-performance converter for Shift-JIS and EUC

Converter for UCS-2 defined in ISO/IEC 10646-1

Locale modules
The library is implemented so that input methods and output methods are separated and are independent of each other. Therefore, an output-only client does not link with the IM code, and an input-only client does not link with the OM code. Locale modules can be loaded on demand if the platform supports dynamic loading.

Transport Layer
There are several kinds of transports for connection between the IM library and the IM Server. The IM Protocol is independent of a specific transport layer protocol, and the sample implementation has a mechanism to permit an IM Server to define the transports that the IM Server is willing to use. The sample implementation supports transport over the X protocol, TCP/IP, and DECnet.

There are IM Servers for Japanese, Korean, and internationalized clients using IM services.

9.2.23 New X Functions

The following sections discuss the newest AIX X Window System functions.

9.2.23.1 Input Method Values

Table 56 provides the input method values.

Table 56. New Input Method Values

XIM Value	Supported as XIM Value	Optional in R6
XNQueryInputStyle	Yes	
XNResourceName	Yes	
XNResourceClass	Yes	
XNDestroyCallback	Yes	
XNQueryIMValuesList	Yes	
XNQueryICValuesList	Yes	
XNVisiblePostion	No	Optional

XIM Value	Supported as XIM Value	Optional in R6
XNR6PreeditCallbackBehavior	No	Optional

9.2.23.2 Input Context Values

Table 57 provides the input context values.

Table 57. New Input Context Values

XIC Value	Supported as XIC Value	Optional in R6
XNArea	Yes	
XNAreaNeeded	Yes	
XNBackground	Yes	
XNBackgroundPixmap	Yes	
XNClientWindow	Yes	
XNColormap	Yes	
XNCursor	Yes	
XNDestroyCallback	Yes	
XNFilterEvents	Yes	
XNFocus Window	Yes	
XNFontSet	Yes	
XNForeground	Yes	
XNGeometryCallback	Yes	
XNHotKey	Yes	Optional
XNHotKeyState	Yes	
XNInputStyle	Yes	
XNLineSpacing	Yes	
XNPreeditCallbacks	Yes	
XNPreeditState	Yes	Optional
XNPreeditStateNotifyCallback	Yes	
XNResetState	Yes	Optional

XIC Value	Supported as XIC Value	Optional in R6
XNResourceClass	Yes	
XNResourceName	Yes	
XNStatusCallbacks	Yes	
XNStringConversion	No	Optional
XNStringConversionCallback	No	Optional

9.2.24 X Output Method

Locale-dependent text may include one or more text components, each of which may require different fonts and character set encodings. In some languages, each component might have a different drawing direction, and some components might contain context-dependent characters that change shape based on relationships with neighboring characters.

When drawing such locale-dependent text, some locale-specific knowledge is required; for example, what fonts are required to draw the text, how the text can be separated into components, and which fonts are selected to draw each component. Furthermore, when bidirectional text must be drawn, the internal representation order of the text must be changed into the visual representation order to be drawn. An X Output Method (XOM) provides a functional interface so that clients do not have to be aware of locale-dependent details.

Two different abstractions are used in the representation of the output method for clients:

- The abstraction used to communicate with an output method is an opaque data structure represented by the XOM datatype.
- The abstraction for representing the state of a particular output thread is called an output context. The Xlib representation of an output context is an *XOC*, which is compatible with XFontSet (see Section 9.2.25.1, “XLC_FONTSET Category” on page 530) in terms of its functional interface.

9.2.24.1 Output Method Functions

The following are the various output method functions:

- XOpenOM()
- XCloseOM()

- XSetOMValues()
- XGetOMValues()
- XDisplayOfOM()
- XLocaleOfOM()

9.2.24.2 Output Context Functions

An output context is an abstraction that contains both the data required by an output method and the information required to display that data. There can be multiple output contexts for one output method. The programming interfaces for creating, reading, or modifying an output context use a variable argument list. The name elements of the argument lists are referred to as XOCvalues. It is intended that output methods be controlled by these XOCvalues. As new XOCvalues are created, they should be registered with the X Consortium. An XOC can be used anywhere an XFontSet() can be used, and conversely.

The concepts of output methods and output contexts include broader, more generalized, abstraction than font set, supporting complex and more intelligent text display, and dealing not only with multiple fonts but also with context dependencies. However, XFontSet() is widely-used in several interfaces, so XOC is defined as an upwardcompatible type of XFontSet().

9.2.25 X11R6 NLS Database

The locale sensitive functions in Xlib refer to the X NLS database or X Locale Database. The Locale Database is the internationalized portion of Xlib but is not actually part of Xlib itself. This allows easier customizing of the X Locale Database without affecting the X environment.

In X11R6 the constituent elements of an X NLS database are:

- locale_name/XLC_LOCALE
- locale_name/Compose
- locale.alias
- locale.dir
- locale.ldx
- compose.dir
- tbl_data/charset_table

In X11R6 the location of the NLS database has changed from that of X11R5. Currently, the X11R5 NLS database is found under /usr/lib/X11/nls, and the X11R6 NLS database is found under /usr/lib/X11/locale. This is of importance from a binary-compatibility standpoint with X11R5 libX11.a provided for backwards-compatibility.

Each locale database file (XLC_LOCALE) contains one or more category definitions. In X11R6, the category XLC_FONTSET defines the XFontSet-related information, and the category XLC_XLOCALE defines the character classification and encoding conversion information.

9.2.25.1 XLC_FONTSET Category

The XLC_FONTSET category defines the XFontSet relative information. It contains the CHARSET_REGISTRY_CHARSET_ENCODING name and character mapping side (GL, GR, and so on), which is used in the X output method, see Section 9.2.24, “X Output Method” on page 528.

The XLC_FONTSET information is held in the class fsN (where N=0, 1, 2,...). The fsN class includes encoding information for the Nth character set or charset. For example, if there are four charsets available in the current locale, four fontsets, fs0, fs1, fs2, and fs3, should be defined. This class has two subclasses:

charset	Specifies encoding information to be used internally in Xlib for this fontset. An example of the charset format is ISO8859-1:GL
font	Specifies a code set of the font to be used internally in Xlib for searching appropriate fonts for this fontset. The left-most entry in this field has the highest priority.

The XLC_FONTSET category also contains two classes not documented in the X11R6 "X Locale Database Definition". These are:

object_name
Allows selection of which X Output Method (XOM) will be used.

on_demand_loading
Delays loading the fonts (not the font information) until they are actually needed. In some cases the on_demand_loading feature has been used to improve font performance.

9.2.25.2 XLC_XLOCALE Category

The XLC_XLOCALE category defines character classification, conversion, and other character attributes.

Note: The X11R5's locale database is not compatible with X11R6's locale database.

9.2.25.3 locale_name/Compose

The locale_name/Compose file contains the compose sequence rule of the locale for the local input method. This file is mapped to the compose.dir file according to a locale full name (see Section 9.2.25.4, “Configuration Files” on page 531). This file is not used by the AIX specific input methods. AIX supplies compose tables as part of the base operating system.

9.2.25.4 Configuration Files

All configuration files of the X11R6 NLS database are under the /usr/lib/X11/locale directory.

locale.alias	This file maps a simplified locale name, or alias, to a full locale name. For example, en_US to en_US.ISO8859-1. The locale.alias file is referenced first by Xlib to map the simplified platform-specific locale name to the X internal full locale name. Xlib uses the full locale name to lookup further information.
locale.dir	This file maps a full locale name to the locale database file that should be read. The locale.dir file is referenced after the locale.alias file to determine the locale definition file corresponding to the X internal locale name. For example, en_US.ISO8859-1 to iso8859-1/XLC_LOCALE. Note that /usr/lib/X11/locale is assumed, so libX11 should read in the file /usr/lib/X11/locale/iso88590-1/XLC_LOCALE when running in the en_US locale.
locale.ldx	This file is read by the AIX dynamic locale loader and maps a locale name to an AIX-specific XLC. For every locale, one of the following is specified: STATIC - Tells the AIX locale loader to use the statically linked AIX XLC object. DYNAMIC - Specifies that an XLC should be dynamically loaded. NONE - Tells the AIX locale loader to return without loading a locale.

9.2.25.5 tbl_data/charset_table

The tbl_data/charset_table file contains the charset conversion table for each charset. This file is used by the UTF locale in the sample implementation and is not used by AIX X11R6 XLC.

9.2.26 Command Line Interfaces

The following sections describe several command line interfaces.

9.2.26.1 xhost

Two new families have been registered: LocalHost, for connections over a non-network transport, and Krb5Principal, for Kerberos V5 principals.

To distinguish between different host families, a new xhost syntax, `family:name`, has been introduced. Names are as before, families are as follows:

```
inet:     Internet host
dnet:     DECnet host
nis:      Secure RPC network name
krb:      Kerberos V5 principal
local:   contains only one name, ""
```

The old-style syntax for names is still supported when the name does not contain a colon.

9.2.26.2 xrdb

Many new symbols are defined to tell you what extensions and visual classes are available.

9.2.26.3 twm

An interface for setting client priorities with the SYNC extension has been added.

9.2.26.4 xdm

There is a new resource, choiceTimeout, that controls how long to wait for a display to respond after the user has selected a host from the chooser.

Support has been added for a modular, dynamically-loaded, greeter library. This feature allows different dynamic libraries to by loaded by `xdm` at run time to provide different login window interfaces without access to the `xdm` sources. The name of the greeter library is controlled by another new resource, greeterLib.

When you log in through `xdm`, `xdm` will use your password to obtain the initial Kerberos tickets and store them in a local credentials cache file. The credentials cache is destroyed when the session ends.

9.2.26.5 xterm

The `xterm` terminal emulator has been minimally internationalized to use the Xlib built-in input method with 8-bit character sets.

There is now support for a few escape sequences from HP terminals, such as memory locking. The termcap and terminfo files have been updated to reflect this.

The logging feature of xterm has been removed. This change first appeared as a public patch to Release 5.

9.2.26.6 xset

The screen saver control option has two new suboptions to immediately activate, or deactivate, the screen saver:

```
xset s activate
xset s reset
```

9.2.26.7 imake

The command line option `-C filename` has been added to `imake`. This option specifies the name of the `.c` file that is constructed in the current directory. The default is Imakefile.c. Threads.tmpl will be added for multithreaded rules.

9.2.26.8 xsm

The `xsm` session manager has many enhancements. Advanced signal handling in `xsm` appears for the first time in release 6.1. The session manager `dtsession` from CDE is the default for AIX.

9.2.26.9 xmh

The `xmh` mail reader is now session-aware.

9.3 Motif Version 2.1

This section provides an overview of all the changes in Motif 2.1 with respect to Motif 1.2 (and Motif 2.0). Compatibility with CDE/Motif 1.2 was given great emphasis in this release, even at the expense of compatibility with OSF/Motif 2.0. Some OSF/Motif 2.0 applications may experience problems because of the changes.

The next section reviews the features most visible to a desktop end-user, the new widgets. The following sections concern the toolkit and detail the extensibility framework, new features in the toolkit, namely Uniform Transfer Model (UTM), reorganization of the menu system, as well as miscellaneous enhancements.

We also describe how the UIL technology has been enhanced to support the extensibility framework and how it has become architecture-neutral with regards to the underlying operating system and computer processor.

A load-only version of the Motif 1.2 libraries is shipped in the X11.motif.lib fileset. The Motif libraries shipping on AIX 4.3 contain the Motif 1.2 and 2.1 shared objects, so the default installation of Motif provides versions of Motif for existing applications (Motif 1.2) and for development of 64-bit and threaded Motif applications (Motif 2.1). However, there is no binary compatibility path for Motif 1.2 to Motif 2.1 since the shr4.o shared object shipping in libXm.a is Motif 1.2, and not Motif 2.1.

9.3.1 New Widgets

To match the function of other GUIs, OSF issued a request for widgets in July 1992. The widgets that were selected through this process are a multiple-font text widget (from Digital), a Container and a Note Book (both from IBM), and a combo-box also called drop down list, from Lotus. Finally, OSF also included a thermometer-like scale and a spin box widget. A graphical example of all the new widgets is presented in Figure 96.

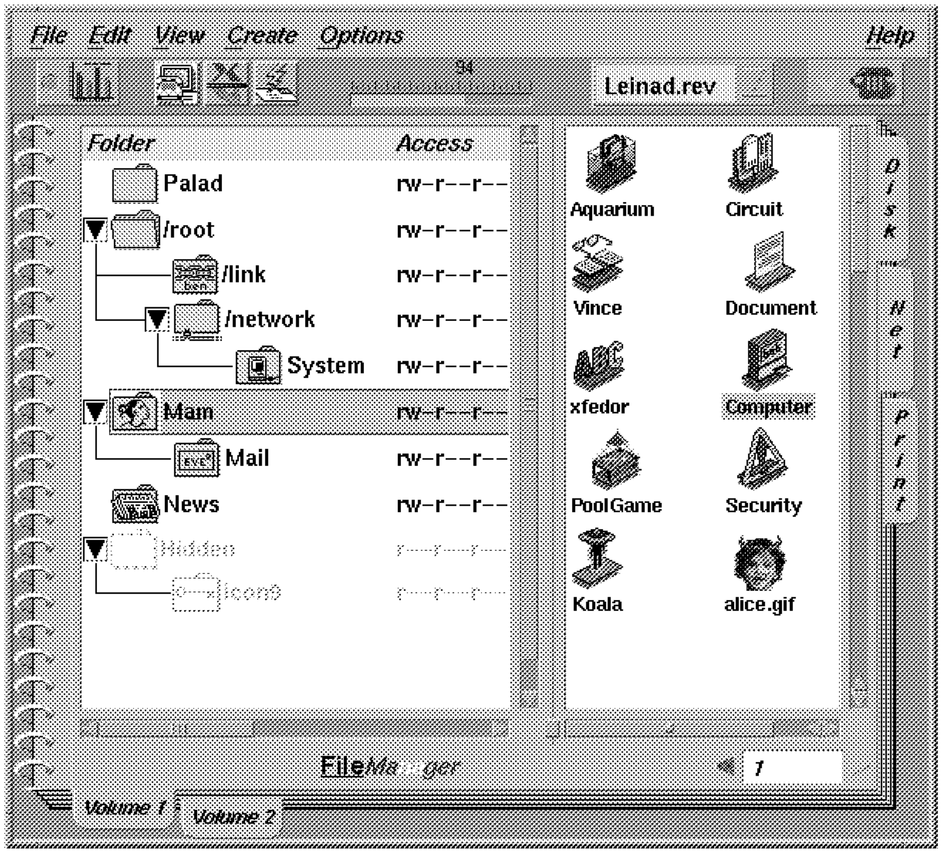


Figure 96. Example of the New Motif Widgets

9.3.1.1 Container Widget

The Container is a critical widget for the direct manipulation, object-oriented world toward which application software is moving. A container object offers views of the objects that the application can handle. Through direct manipulation, the user can select, move, copy, or delete these objects, drag them into other applications, or drop new objects into the container. The Container widget is a very powerful tool for application writers. Potentially, every application can be implemented with a Main Window and a Container. The menu bar of the Main Window specifies the generic operations on objects, and the container displays the objects. A double click on an object activates it, and popup menus can be available for specific operations on the objects.

A Container can contain hierarchical objects that can be decomposed into further objects. The Container widget offers three kind of views:

- An icon view, in which every object is represented by an icon.
- A hierarchical view, also called outline or tree view, which displays the object tree.
- A detail view, a tabular view where every object appears as a line in a table and properties of the object are listed in the columns.

Icon View A container may display the objects in the typical large icon view. In this view, every object (a top level object of the tree) is displayed as an icon with a label. The Container widgets support different types of layout for those icons. The most popular are:

Grid mode, where each icon fits into a grid cell.

Free mode, where the user can freely move the icons around.

As an illustration, see the right side of Figure 96 with the Aquarium and the Circuit icons.

Tree View In the tree view, every object at the top level of the hierarchy is displayed with a small icon. If the object has sub-objects attached, the hierarchical organization is displayed with lines and indentation reflecting the tree structure. This can be seen on the left side of Figure 96 with the folder structuring.

For very large hierarchies that would take too much space, the Container widget offers a browsing facility. For each level of the hierarchy, the application can first decide whether or not the information below that level should be displayed. Secondly it can tell the container whether or not the user should be allowed to browse further.

If the user is not allowed to explore that branch of the tree, a mini-icon is displayed called the collapsed pixmap.

If the user is allowed to explore that branch of the tree, a mini-icon is displayed called the expand pixmap. Clicking on the expand pixmaps displays one more level in the hierarchy. As this new level is displayed, the mini-icon is turned into a collapse pixmap. When you click on the icon again, that tree branch is removed from the view.

Detail View The detailed view displays a table where each row represents an object, and each item in the row is a property of that object. The application is responsible for providing the detail data and the rendition attributes for it to be displayed.

9.3.1.2 Note Book

The Note Book is another powerful widget, simulating a real notebook that allows the application to display one page among a stack of pages, maintaining a constant page size. The Note Book widget includes pages, tabs, a status area and the page scroller. It stacks its page children so that all page children occupy the same area just like real-world book pages. Tabs simulate notebook tabs. Major tabs are used for dividing pages into several sections, and minor tabs are used for subdividing the sections. The page scroller is used for switching the current page back and forth. The Note Book also provides tab scrollers for scrolling major and minor tabs when it cannot display all tabs. Tab scrollers are visible and enabled only when there is insufficient space to display all the major tabs or the minor tabs appropriate to the page. A complex implementation of a Note Book can be seen in Figure 96 on page 535.

Major tabs displayed on the side of the Note Book allow the user to quickly access data. Once a primary tab is chosen (for example, a chapter), the minor tabs appear in the orthogonal direction (for example, sections) and allow further indexing inside the relevant parts of the document.

Pages, tabs, status areas, and the page scroller are created by the application as children of the Note Book widget. Any Motif widget may be a page of the Note Book. A major tab, or a minor tab, may be attached to a page by creating a tab child of the Note Book and setting a constraint to the page number of the targeted page. Note that the notebook widget makes heavy use of the traits of its child widgets to invoke their class methods. A tab, either a major tab or a minor tab, must be a Motif widget with a trait that can be activated (such as a push button). Tabs in a Note Book are associated with a page number not attached to any actual page widget. Therefore, it is possible to have a tab with an empty page. Destroying a page widget leaves an empty page. It does not tear the page out of the XmNotebook.

The page scroller child is not associated with a certain page. There is only one valid page scroller per Note Book, and it must carry the navigator trait. Since the application of the Note Book can provide page numbers, it is possible to have duplicate pages and empty pages. An empty page is a page slot where no page is inserted. This page displays just a blank background unless the application provides some visual information to this area while processing. Note that this feature is very useful for applications that have to display hundreds (or even thousands) of pages. It is not necessary to have hundreds of physical pages in memory. An application may actually use only one physical empty page, provide adequate tab indexing for the real number of pages it supports, and only update the contents of the (single) physical page with the information when required.

The Note Book widget is a versatile tool. Typical uses are:

- The display of online documentation, with quick access to information by using the tabs index.
- Reducing the real estate occupied by an application on the screen by making visible only the information required at that particular moment. A Note Book allows a developer to very quickly implement any kind of agenda or calendar application. It is the optimal base for a card filer application and provides good support for a hypertext applications.
- Dialog boxes that have variants. For example, a property sheet dialog box for a document editor might have a font part and a margin part as pages of the Note Book.

A Note Book allows a developer to very quickly prototype all kinds of applications that display organized information, such as a card filer, an agenda, or a calendar application.

9.3.1.3 Combo Box

As its name indicates, Combo Box combines the capabilities of a single line text widget and a list widget that allows users to type in information and also provides a list of choices to complete the text entry field. The list can either be displayed at all times or dropped down by the user. That is why Combo Box is sometimes referred to as *drop-down list*. An example of a Combo Box is shown on the top right of Figure 96 with the label Leinad.rev.

When the list portion of the Combo Box is hidden, users are given a visual cue, a downward-pointing arrow next to the text field. These drop down Combo Boxes are useful when presentation space is limited or when users will complete the text entry field more often by typing text than by choosing the entry field text from the list. The drop down Combo Box list should popup in the visible area of the screen but stay aligned with the text field (that is, at the bottom-edge of the screen, the list will be *up*).

The application programmer provides an array of strings that will fill the list. Each string becomes an item in the list, with the first string becoming the item in position one, the second string becoming the item in position two, and so on. The list can actually be manipulated with the regular API of the XmList widget.

Similarly, the text entry may be accessed by using XmTextField API. The size of the list is set by specifying the number of items that are visible in the list (XmNvisibleItemCount). If the number of items in the list exceeds the number of items that are visible, a vertical scroll bar will appear that allows the user to scroll easily through a large number of items.

A Combo Box allows a single item to be selected in two ways: by scrolling through the List and selecting the desired item, or by entering text into the text entry field. Selecting an item from the list causes that item to be displayed in the text portion of the Combo Box. The single-line text field in a Combo Box can be either editable or noneditable. If the text field is editable, the user can type directly into it to enter a new list item. If the application needs to validate the entered text, it can do so by installing a XmNmodifyVerifyCallbacks on the text field. If the text field is noneditable, typing text invokes a matching algorithm that attempts to match the entered text with items in the list.

9.3.1.4 Spin Box

Although more modest, the Spin Box widget offers a quick way of selecting, cycling, or setting a value within a range. It can be used by a very large range of applications. The Spin Box function can be implemented using the existing Motif Arrow Button widget and a label widget. However, the Spin Box widget makes it much easier for the developer and uses much less memory.

9.3.2 Motif Changes in Behavior

The following changes have been made to the behavior of Motif 2.1:

- Take focus without activating on Ctrl+Btn1 in List and Button widgets.
- Supports an indeterminate state and associated visual in the existing toggle button and toggle button gadget. The indeterminate state is useful in application property sheets.
- Snap-back scrolling, a feature that allows users to have the scroll bar slider snap-back to its starting position when the mouse is dragged beyond a certain distance from the scroll bar area.

9.3.3 The Motif Extensibility Framework

One of the problems Motif developers are facing is the complexity of developing new widgets for their applications. Subclassing Motif widgets without the source code of the parent class is a challenge with Motif 1.2. Even developing a new class from the superclass source code is not always easy. There are four sets of issues for the widget developer:

- Developing a new widget from scratch can take a significant amount of time. For its internal widget development, OSF built a library of internal functions that are commonly used in the widget set. For example, there are functions to draw 3D shadows. If those functions were available to widget writers, it would speed up the development process.
- The Motif widget class methods are currently not documented. Widget writers have to browse into the superclass source to figure out the class methods and what they do. Documenting the widget class methods of the most frequently subclassed widgets helps widget development.
- Sometimes, it is not possible to have new widgets behave as expected when they are managed by a standard parent. One of the key issues is that, in a number of cases, manager widgets perform tests about the nature of their children. These tests are typically done by testing the class pointers of the children. When applied to a custom widget, they fail, and the widget does not behave properly.
- Motif subclassing requires use of the Xt object framework, implemented in C. Most C++ application programmers, now a significant number, want to write subclasses of the Motif widgets directly in C++.

OSF has addressed these issues by developing an extensibility framework for Motif that will significantly help OSF/Motif widget developers and application programmers. This framework consists of four parts:

- A new mechanism has been developed on top of intrinsics, called *Traits*.
- A new set of APIs has been documented by OSF for widget writers.
- A new book has been published by OSF with documentation on how to correctly subclass a Motif widget. This documentation is accompanied by a CD-ROM containing source code illustrations. The book also contains documentation for the existing class methods that can be reused by developers.
- A set of C++ base classes has been developed that makes it possible to derive subclasses of the Motif Manager and Primitive widgets (the most frequently derived) directly in C++ while retaining the ability to use Motif functions on those widgets, such as keyboard traversal function XmProcessTraversal().

9.3.3.1 Traits

The trait abstraction was first introduced by Xerox. A trait is a characteristic of an object that can be expressed by a set of methods/data applied to, or carried by the object holding that trait. At a higher level, a trait implements a behavior that can be shared by different objects of a system without requiring any particular hierarchical relationship between these objects or any particular implementation of the trait.

For example, consider the *I am a navigator* trait. This trait characterizes the fact that an object can set a value between a minimum and a maximum limit, regardless of the particular method used to set the value or the particular look and feel. A navigator can be implemented by a scroll bar, a thermometer-like control, or even a choice in the list of all possible values. The navigator provides the abstraction that guarantees that any object carrying this trait can be used to navigate arbitrarily between two values.

The I am a navigator trait implements methods that make it possible for clients of the object carrying the trait to set and get the min/max/current value independently of the implementation.

In Motif 2.1, both the XmScrollBar (an XmPrimitive subclass) widget and the XmSpinBox (an XmManager subclass) hold the I am a navigator trait, and they can both inherit, or specialize, the methods of that trait.

Therefore, the traits mechanism makes it possible to implement a form of multiple inheritance on top of the Xt object oriented framework, which only supports single inheritance. In the Motif 2.1 model, traits are seen as light abstract classes, sometimes called mix-ins.

Traits also make it possible to implement a form of polymorphism on the various widgets. For example, if a trait has a setvalue method, it can be used to set values of resources that actually have a different name. For example, a trait::setvalue() replaces XtSetValues() on multiple XmNresource names.

A side effect of traits is the potential reduction of code size in applications. In many cases, a Motif Manager widget would check the class of its children. Doing so requires access to the class pointer and at link time. Linking the class pointer global variable usually links chained modules in the application. Because those tests are replaced in the manager code with checking the children’s traits, the spaghetti effect is eliminated.

Traits also augment code reuse inside Motif. A widget writer can simply decide to inherit a trait from another widget. It then inherits the class methods implemented by the OSF engineers at no cost.

9.3.3.2 Uniform Transfer Model (UTM)

OSF/Motif 1.2 provides the application programmer and widget writer with a number of different mechanisms to interchange data between applications. There are four forms of data transfer supported within Motif:

- Primary transfer. When the user clicks on pointer button BTransfer, the data currently selected (possibly in another application) is immediately transferred at the pointer location.
- The clipboard. This is probably the most widely-known mechanism. When the user activates the cut key (or the cut command into the Edit menu) the selected data is cut into the clipboard. Activating the paste key inserts the data at the current location of the destination cursor.
- Drag and drop. The visual metaphor for data exchange. The user can select data, drag it over the screen and drop it onto a recipient.
- Secondary transfer, also referred to as quick-copy, where the user first selects a destination and then uses Alt-BTransfer to select data that is moved to the destination by releasing the pointer button.

In each case, a data transfer can be characterized by the fact that:

- Some data is owned by an application, called the source.
- Another application, called the destination, wants to acquire the data. To acquire the data, the destination usually sends a request to the owner. The destination is also called the requestor.

Each particular transfer mechanism in Motif requires a new programming effort to accommodate, and thus, there is a burden on the programmer if all modes are to be supported, as recommended by the *Motif Style Guide*.

In Motif 2.1, OSF has implemented a new mechanism to unify access to the different mechanisms. Moreover, this mechanism allows the application programmer the ability to extend the function of toolkit-supplied widgets in the area of data interchange.

The model is very simple:

- On every widget that can act as source emit data, a convertCallback is available.
- On every widget that can act as a destination (receive data), a selectionCallback is available.

The convert callback is called each time the user has requested data through any mechanism. The programmer only needs to provide a buffer containing the data and the name of the data type (an atom). Motif automatically transfers the data and frees the memory after the data has been received. Similarly, the selection callback tells the programmer that data has arrived.

9.3.3.3 Menu System Improvements

Motif popup menus were never simple to program with Motif 1.2. When a popup menu was created, it did not popup automatically on the screen when a button was pressed, some programming was required to make it popup. It had the sense of a task left incomplete, particularly since the code would popup a menu when invoked from the Menu key of the keyboard.

In Motif 2.1, the following was implemented:

- Programmers can freely install popup menus on arbitrary widgets in the tree. For each menu, it is only necessary to specify whether the menu is only valid for the widget it is installed upon or for children of that widget as well.
- The menu system automatically pops-up the appropriate popup menu when the user of the application presses the appropriate button within the application.

For the most simple specification of an automatic popup, nothing else needs to be done by the application programmer other than creating the popup menu associated with its manager.

9.3.4 Miscellaneous Enhancements

The following sections describe various Motif enhancements:

9.3.4.1 Printing

Starting with Release 2.1, Motif includes support for printing using an X protocol-based print server. This print server produces output in three formats: PCL, PostScript, and Raster.

9.3.4.2 Thread-Safe Libraries

Xm and Mrm are thread-safe-enabled. This means that the libraries themselves are threadsafe, and a multithreaded application need not do explicit locking when accessing these libraries.

9.3.4.3 File Selection Box

The Motif File Selection Box offers a powerful, but complex, filtering mechanism for file selection. It offers all the power of the UNIX file system to users. The CDE User Model group, however, has come up with a design that seems better-suited for the average user.

The CDE File Selection Box replaces the regular filter control with two controls: the base directory entry and a wild-card entry for file name pattern. Only the subdirectories are shown in the directory list and the file names in the file list. Hence, scroll bars are not needed in this case.

OSF did not want to make obsolete the documentation for existing applications. They also wanted to maintain the old Motif 1.2 behavior. For these reasons, they decided to maintain both the old and the new styles. However, the new CDE style has become the default since it is more intuitive for most users.

9.3.4.4 String Manipulation

Motif offers string manipulation through an abstract object called XmString. The XmString abstraction associates a multilingual string with the encoding information attached to each locale and rendition to be used for a particular substring.

In Motif 1.2, the XmString abstraction supports the display with multiple fonts. However, the programmer has to separately create each character string that uses a different font, associate that string with a tag, and match that tag with a font. In Motif 2.1, the rendition is both simplified and extended. It is extended to support multiple fonts as well as multiple colors and TAB marks. It is simplified with the availability of a new API. This new API offers construction of the XmString.

9.3.4.5 Toggle Button Enhancements

When used as check boxes, toggle buttons (on/off buttons) are often not intuitive in Motif 1.x releases. Users are sometimes confused whether the etched-in appearance means on or off. The 3D effect used for the etched-in look is not always easily recognized in unusual light conditions (for example, industrial or military environments) or screens with limited color range. This has lead to enhancing the ToggleButton widget to support these new features:

- A check mark can be used inside the toggle square area.
- Since the check mark sign is particular to the US, for other cultures check boxes can be crossed instead.
- In addition to the regular button colors, two additional colors can be used to indicate On or Off.
- In radio boxes, a circular shape can be used instead of the diamond shape.

An example of the new toggle buttons are shown in Figure 96 on page 535 in the upper-left and right corners.

9.3.4.6 Support for Right-to-Left Layout

Support has been added for languages written from right to left. Many of the Motif widgets now automatically reverse the geometry when the direction is right-to-left. For example, the Form widget automatically switches the left margin and the right margin, as well as the attachment constraints. The result is that developers can design applications using the internationalization facilities and specify a constrained layout that will work in both left-to-right and right-to-left environments.

9.3.4.7 Support for XPM Format

Motif 2.1 requires that applications are linked with the Xpm library. This library is freely available from the /contrib directory of the X Consortium and is also duplicated on the Motif tape. Xpm support makes it possible to support multi-color pixmap icons instead of two-color bitmaps.

9.3.4.8 Vertical Paned Window

The Motif 1.x Paned Window widget can only have horizontally-separated panes. Motif 2.1 adds a vertical mode.

9.3.4.9 Unit Conversion

New conversion methods have been added so that a user can specify a window size in inches, millimeters, or typographical points.

9.3.4.10 List Enhancements

When a selection is made from XmList, a new resource specifies whether XmList takes ownership of the primary selection. A keyboard navigation facility was added allowing the user to navigate directly to an item by typing the first character of that item. This is useful in lists organized in alphabetical order.

The Motif 1.2 List widget does not exhibit very good startup performance with large lists. An optimization was found in the algorithms and memory allocation strategy that reduces the startup time of a List widget by more than 30 percent on a typical workstation.

9.3.4.11 XmScreen Enhancements

Several resources have been added to the XmScreen widget to allow per-screen behavior.

A new resource is available to specify whether pixmaps or bitmaps should be used in an application. If bitmaps are used, the application is limited to two-color icons, or stipples, and it uses much less memory (typically eight times less on 8-bit color screens).

A set of resources was added to enable control of the color schemes by the application. The goal of these new resources makes it possible for an application to:

- Override the color calculation method used by Motif to generate 3D effects.
- Allocate the pixels in the colormap.

Color-intensive applications generally want to control the color schema and color allocation to optimize their own algorithms and use of the colormap. For example, on a system with 256 entries in the colormap, an application may want to reserve all 256 colors for its own usage. There are then no colors left for the Motif toolkit. With Motif 2.1, the application can plug-in its own allocation function. Then, each time Motif needs a new color, it calls that function. The application can then *loan* to Motif one of the colors it is already using.

A new screen resource allows an application to customize the bitmap that is used by Motif for the rendering of insensitive visuals. Support is also added for applications that want to display menus in overlay planes. Menus can now be displayed in a visual that is different than that of the application.

9.3.4.12 Virtual Bindings

As of Motif 1.2, users can customize the keyboard mapping by defining a keymap between virtual keys (for example, osfCancel) and real keys (for example, Cancel or Escape). They then run the `xmbind` command to bind the real keys to the virtual keys. However, a single real key can only be assigned to a single virtual key.

In Motif 2.1, each virtual key can be associated with multiple real keys, allowing users to use, for example, both the Escape key or the Cancel key to cancel an operation. Of course, errors can occur if a user incorrectly maps the same real key to multiple virtual keys.

9.3.4.13 Drag and Drop Enhancements

Several enhancements have been made to Drag and Drop. The following features are useful for sophisticated applications that are advanced in drag and drop technology.

A function has been added to query if a widget is a dropsite or not. A generic application-wide dragStartCallback has been added, so an application can be aware and take action for any drag occurring at any time in the application.

To provide better drag and drop feedback to the user, Motif 2.1 makes it possible for the drag icon to change in real time to show the effect that a drop would have if it occurred at the current location of the mouse cursor. The effect depends on the individual item, or items, that are being dragged.

Another enhancement that has no API, but is fairly visible to end users, is drag-scroll support. Users of small screens are often faced with the problem of not being able to drag data from one place and drop it in another because the target window is currently scrolled out of view. The user has to drop the icon on the desktop, scroll the window so that the destination is visible, and restart the drag. Drag scrolling prevents this, allowing the user to move the cursor over the scroll bar direction control and pause. After a customizable delay, the scrollable window automatically begins to scroll and makes the destination appear. As soon as the user moves the cursor, scrolling stops.

9.3.4.14 Scrolled Window and Scroll Bar Enhancements

The Scrollbar widget has been extended to support a look and feel like OpenStep, where both up and down controls are close together at one end of the scroll bar, as opposed to one control at each end. A new resource was added to give applications more control over the appearance of the slider.

It is practically impossible with Motif 1.2 for an application to implement a scrolled window that has a fixed title or a status bar inside the scrolled window area. Motif 2.1 makes it possible to have one or more arbitrary controls be displayed as non-scrollable in any direction.

Using this feature, users can simply implement a spreadsheet application with a column title that cannot be scrolled out vertically, a rows title that cannot be scrolled out horizontally, and a RowColumn widget for the cells. MainWindow, a subclass of ScrolledWindow, inherits this feature too.

9.3.4.15 Drawing Area

The drawing area is now traversable with the keyboard, making it possible for users to traverse to a drawing area and get its associated functions (for example, popup menus).

9.3.4.16 Performance Enhancements

A number of performance enhancements have been made and are described below.

XmString Performance

The memory used by an XmString that uses a single font and is smaller than 256 bytes (256 characters in Latin-based language environments) is less than it used to be. Note that this kind of string represents the majority of strings displayed in a typical user interface, for example, menu items, list items, and labels. The memory size of every XmString has also been reduced by four bytes (from Motif 1.2). Every string that consists of less than 256 characters uses a single encoding, single color, and single font.

Copying XmString is now managed by reference counting in 2.1, which makes XmString copy much faster and uses no memory. Since Motif does a lot of XmString copies internally, this should reduce the memory size used by many applications that do not release the memory of an XmString after they have passed it to a widget.

List Performance

The startup time for the List widget has been decreased compared to Motif 1.2. This is particularly noticeable for applications that handle large lists of several hundred items. See Section 9.3.4.10, “List Enhancements” on page 545.

X-Related Performance

In general, the amount of resources allocated in the X server, clients, and the number of round trips between client and server has been decreased.

Client-side memory has been decreased by using bitmaps instead of pixmaps whenever possible. In Motif 1.2, when a widget uses a background pixmap, a pixmap is allocated using the available depth of the screen. In Motif 2.1, a bitmap is used if only two colors are required.

Motif 2.1 has been optimized to generate fewer mouse events from the X server when using gadgets.

9.3.4.17 UIL Extensibility and Portability

Since OSF introduced an extensibility framework, it was necessary to reflect that in the UIL language. Once a developer has developed a new widget, that new widget should be usable from UIL as well. However, they should not have to generate a new compiler for every new widget supported. It was, therefore, necessary for OSF to develop a mechanism such that the UIL compiler could dynamically incorporate new widgets to be accepted in UIL source files and generate the appropriate binary UID code.

In the past, OSF received numerous enhancements requests because the UID files in the 1.x releases were not architecture neutral. You had to compile the same UIL source file for every target platform. Moreover, only 32-bit platforms were supported.

With OSF/Motif 2.1, OSF developed a new UIL technology that reaches those three objectives simultaneously:

- The UIL compiler can be told to dynamically accept new widgets.
- The UID files generated by the compiler are architecture neutral.

Note that the architecture neutral format is still dependent on 32-bit versus 64-bit architectures. A UIL file compiled on a 32-bit architecture can be read on any other 32-bit architecture but not on a 64-bit architecture. Similarly, a UIL file compiled on a 64-bit architecture can be read on any other 64-bit architecture but not on a 32-bit architecture. This design decision was made because the memory used for 64-bit architectures is about twice the size of that used on 32-bit, and 32-bit platforms should not have to bear a 100 percent penalty.

9.3.5 Compatibility with Motif 1.2 and 2.0

As stated at the beginning of this section, compatibility with CDE/Motif 1.2 was given greater emphasis than compatibility with OSF/Motif 2.0. The following issues should be noted concerning compatibility with Motif 1.2:

- If an application subclasses a widget in the group of DrawingArea, Label, List, Manager, or Primitive, there is a potential for binary incompatibility. The Class Records for these widgets changed from Motif 1.2 to Motif 2.0. The only resolution in this instance is to recompile, unless the application writer used XmResolvePartOffsets (now known as XmeResolvePartOffsets in the *Motif 2.1 Widget Writer's Guide*).
- The XmString definition has changed in Motif 2.1. It is now a union instead of a typedef of char *. Existing Motif applications running on Motif 2.1 may have problems depending upon their usage of XmString.

Some OSF/Motif 2.0 applications may experience problems because of the following changes:

- The XmCSText widget has been withdrawn, as have those APIs added to OSF/Motif 2.0 solely to support it.
- Mrm support for word size-independent .uid files has been removed. Existing .uid files compiled with Motif 2.0 UIL may not be readable. As in OSF/Motif 1.2, .uid files are portable only between machines with the same word size.

- New XmComboBox XmNpositionMode and XmSpinBox XmNpositionType resources default to incompatible index values and should be forced by all applications using these widgets. XmONE _BASED is recommended for XmComboBox widgets because it lets applications distinguish between new values entered in the text field and the first item in the list.
- XmStringCreateLocalized now handles new lines and tabs.
- The _XmStrings array has, on some machines, been split into multiple subarrays with the same techniques used by libXt. This preserves compatibility with Motif 1.2 and permits future expansion.
- Labels for automatically-created subwidgets, like the buttons in a file selection box, are now unconditionally localized and may not be set or overridden by the user.
- The XmDisplay XmNenableThinThickness resource now has wider effect than it did in Motif 2.0.
- The XmDisplay XmNenableToggleVisual resource now changes the way XmNindicatorOn and XmNindicatorType values are rendered, instead of simply changing their default values. Motif 2.0 applications that called XtSetValues() for these resources may notice a change. New constants have been added to obtain the old behavior.
- In Motif 2.0, there were two distinct XmREPLACE constants with different values. The XmMergeMode constant has been renamed XmMERGE_REPLACE. This is a source compatibility issue; binary compatibility is unaffected.
- XmDisplay XmNdragReceiverProtocolStyle default value has been reverted to XmDRAG_PREFER_PREREGISTER. Users may find that XmDRAG_PREFER_DYNAMIC is more efficient.
- The XmNenableEtchedInMenu resource causes buttons and toggles in menus to be rendered with different colors than those in earlier releases.
- XmScrolledList and XmScrolledText scroll bar colors are computed differently. They are now derived from the scrolled window's background color, not the color of the XmList or XmText widget.
- To promote convergence with dtwm, mwm's panning, and virtual screen support has been removed, as has mwm's support for workspaces.
- The XmCxx library of C++ wrappers has been moved to the demos/lib directory.

9.4 X Virtual Frame Buffer (4.3.2)

The X Virtual Frame Buffer (XVFB) software technology was first introduced in AIX 4.3.1 and has been further enhanced in AIX 4.3.2 to support the RS/6000 SP.

The XVFB provides a virtual graphics adapter that allows the X Windows Server to start and operate on a server machine that has no physical graphics adapter. In addition, the OpenGL 3D Graphics rendering library can be used with the XVFB technology.

This capability was introduced primarily to provide improved support for 3D Rendering Server Applications. By removing the single shared resource (graphics adapter) from the system, and replacing it with a dynamic resource (system memory), 3D Rendering Server Applications can scale in performance as processors are added to an SMP capable system. Other advantages of XVFB include:

- Less expensive 3D Rendering Server (since no graphics adapter or display is required)
- Better security (since the 3D Rendering Server does not have to be left logged in, XVFB runs in the background)
- Better scaling with SMP machines (since each process gets a private XVFB rendering area)

9.4.1 Direct Soft OpenGL

Direct Soft OpenGL (DSO) is a new IBM OpenGL rendering technology for AIX and RS/6000. Implemented to work with the XVFB, DSO was designed specifically to enhance the performance of 3D Rendering Server Applications by eliminating extraneous interprocess communication, eliminating process context switching overhead, and making rendering and image reading more direct and efficient.

DSO is a pure software implementation of OpenGL that runs as a *direct* OpenGL context. Basically this means that all of the CPU intensive OpenGL work (3D rendering) is part of the application process (not part of the X Server process). By running *direct*, all of the interprocess communications with the X Server are eliminated, making 3D rendering much more efficient. In addition, the AIX operating system is not having to context switch between the X Server and the 3D rendering applications, making system utilization more efficient.

Using the XVFB and DSO software is as simple as installing the XVFB file sets (X11.vfb and OpenGL.OpenGL_X.dev.vfb) and starting the X server with the appropriate options (for example, X -vfb -x GLX -x abx -x dbe -force).

9.4.2 CATweb Navigator and XVFB/DSO

CATweb Navigator allows end users with Java Enabled Web Browsers to view and navigate product information created with CATIA Solutions. By using the intuitive set of Java Applets that make up the CATweb Navigator Client, users can connect to a CATweb server machine, select models for viewing, and then view and navigate the models with a 3D viewer, 2D schematic viewer, or a report style viewer.

One CATweb server machine can support multiple concurrently active CATweb Navigator clients. For each CATweb client that attaches to the CATweb server, one or more CATweb processes will be started on the CATweb server to handle the requests of that particular CATweb client. One of these CATweb processes is a 3D rendering application. This application runs on the CATweb server and is responsible for:

- Loading the requested CATIA model
- Rendering the 3D model on the fly as the CATweb client requests new views
- Compressing the final rendered image
- Transferring the image to the Java CATweb client

This application uses both the X Windows and OpenGL libraries to quickly and accurately render 3D images In addition, CATweb Navigator V2.0 and above has been enhanced to effectively exploit the capabilities of XVFB and DSO.

9.5 X11R6.3 (4.3.3)

X11R6.3 is ported from X consortium to AIX4.3.3. X11R6.3 is upward compatible to X11R6 and X11R6.1 and X11R6/6.1 clients can communicate with X11R6.3 servers. Most X11R6.3 clients except those that use libICE, libXext, and libXp can also communicate with X11R6/6.1 servers. The major new function in X11R6.3 is support for World Wide Web. It addresses secure and browser oriented communication between X servers and clients through low bandwidth networks. X11R6.3 contains following:

- Low Bandwidth X (LBX)
- Remote Execution (RX)

- Security Extension
- Application Group Extension

9.5.1 Low Bandwidth X

Low Bandwidth X, formerly known as X.fast is an extension to X server. It accelerates interactions between X clients and servers by using compression and caching techniques. By utilizing LBX, you can run X applications on a remote server which are displayed on your local X server at moderate response time even if the remote server is connected through WAN or dial-up modems.

9.5.1.1 lbxproxy

Application can take the advantage of LBX by connecting to `lbxproxy`. The `lbxproxy` program, which runs on remote server, is the proxy between remote X applications and local X server. It intercepts X protocol messages from X applications as though it were a real X server and sends the messages to local X server after compressing them. Figure 97 illustrates how `lbxproxy` works:

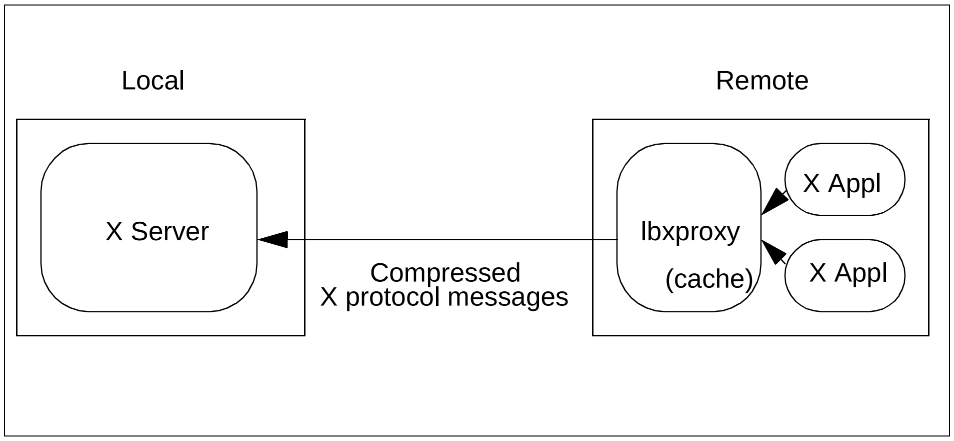


Figure 97. The lbxproxy Logic

The `lbxproxy` command has following syntax

```
lbxproxy [:<display>] [option]
```

where `display` is the display number that the `lbxproxy` program listens, that is, from X application programs the number of the display on which they should be displayed. It is the local display number such as `:1`. A typical `option` is

```
-display local:0
```

that means `lbxproxy` to route X protocol messages to local machine's display 0.

9.5.1.2 lbxproxy Example

On local machine where X applications are *displayed*, use following command to grant access to the X server from remote host:

```
$ xhost +remote
remote being added to access control list
```

or you can use `xauth` to do equivalent.

On remote machine where X applications run, start `lbxproxy` program and a X application specifying the DISPLAY environment variable that `lbxproxy` listens to:

```
$ lbxproxy :1 -display local:0
$ export DISPLAY=:1
$ xclock
```

Then `xclock` should be displayed on your local X server.

9.5.2 X Remote Execution

The remote execution (RX) service of X11R6.3 defines a MIME format for invoking X applications remotely. By specifying `application/x-rx` as MIME type, applications such as Web browsers can display an X application on its window or local X server typically using plug-ins or helper applications.

RX defines following MIME format that browser can use to run X application correctly. Such information includes:

ACTION	Specifies the URL to initiate remote execution. This field typically contains the name of a CGI script.
REQUIRED-SERVICES	Specifies the list of services the application requires. They fall into following two categories.
UI	Specifies the list of user interface protocols the applications supports.
PRINT	specifies the list of printing protocol the application supports.
WIDTH	Specifies the width of display area.
HEIGHT	Specifies the height of display area.

EMBEDDED	Specifies whether the application is to be embedded in the browser or not.
AUTO-START	Specifies whether the application should be started immediately upon retrieval of the RX document or on user demand.
APP-GROUP	Specifies the logical application group to which the application is related.

After receiving these information specified in RX documents, a Web browser returns corresponding return information. The information indicates whether requested information is available on the Web browser's side or not.

9.5.2.1 RX Example

A typical RX sequence is as follows:

- A Web browser gets an HTML document contains a tag that specifies an RX document. The tag looks like the following example:

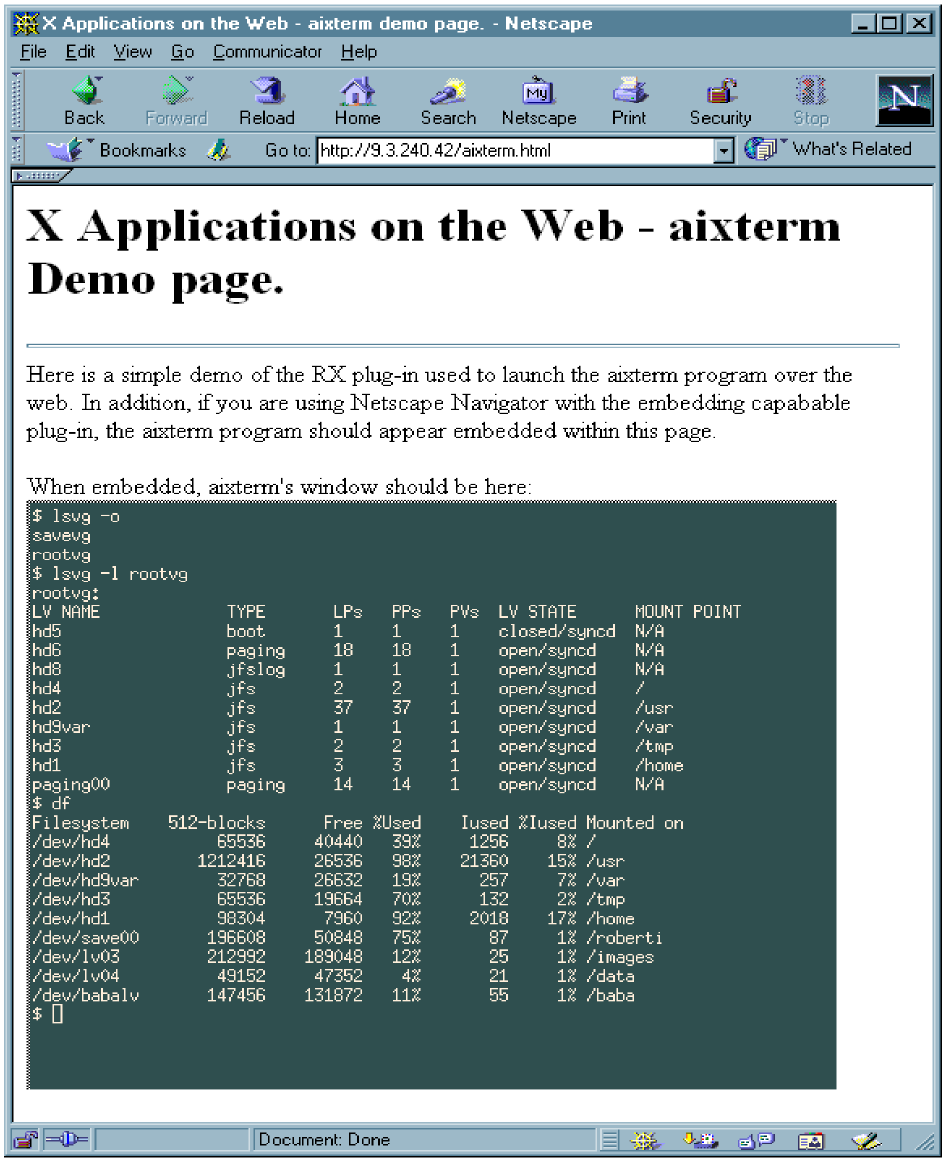
```
<EMBED version=1.0 type=application/x-rx src="aixterm.rx" width=200 height=200>
```

- The Web browser loads the RX document.

```
<PARAM Name=VERSION Value=1.0>
<PARAM Name=REQUIRED-SERVICES Value=UI>
<PARAM Name=UI Value=X>
<PARAM Name=ACTION Value=http://9.3.240.42/aixterm.pl>
<PARAM Name=WIDTH Value=640>
<PARAM Name=HEIGHT Value=480>
<PARAM Name=EMBEDDED Value=YES>
<PARAM Name=X-UI-INPUT-METHOD VALUE=YES;foo>
<PARAM Name=X-UI-LBX Value=YES>
<PARAM Name=X-AUTH VALUE=MIT-MAGIC-COOKIE-1>
```

- Upon receiving the RX document associated with `application/x-rx` MIME type, a plug-in or helper application is activated.
- The plug-in or the helper application registers the application to X server.
- The plug-in or the helper application returns information that corresponds to information specified in the RX document.
- The Web server initiates the CGI program that runs a X application as specified in the ACTION filed.
- The X application connects to X server.

Figure 98 shows an example RX application. The Netscape browser embeds the aixterm screen image in its window.


X Applications on the Web - aixterm demo page. - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Go to: http://9.3.240.42/aixterm.html What's Related
Document: Done

X Applications on the Web - aixterm Demo page.

Here is a simple demo of the RX plug-in used to launch the aixterm program over the web. In addition, if you are using Netscape Navigator with the embedding capabable plug-in, the aixterm program should appear embedded within this page.

When embedded, aixterm's window should be here:

```
$ lsvg -o
savevg
rootvg
$ lsvg -l rootvg
rootvg:
LV NAME             TYPE       LPs   PPs   PVs  LV STATE      MOUNT POINT
hd5                 boot       1     1     1    closed/syncd  N/A
hd6                 paging     18    18    1    open/syncd    N/A
hd8                 jfslog     1     1     1    open/syncd    N/A
hd4                 jfs        2     2     1    open/syncd    /
hd2                 jfs        37    37    1    open/syncd    /usr
hd9var              jfs        1     1     1    open/syncd    /var
hd3                 jfs        2     2     1    open/syncd    /tmp
hd1                 jfs        3     3     1    open/syncd    /home
paging00            paging     14    14    1    open/syncd    N/A
$ df
Filesystem    512-blocks      Free %Used    Iused %Iused Mounted on
/dev/hd4           65536     40440   39%     1256     8% /
/dev/hd2         1212416     26536   98%    21360    15% /usr
/dev/hd9var        32768     26632   19%      257     7% /var
/dev/hd3           65536     19664   70%      132     2% /tmp
/dev/hd1           98304      7960   92%     2018    17% /home
/dev/save00       196608     50848   75%       87     1% /roberti
/dev/lv03         212992    189048   12%       25     1% /images
/dev/lv04          49152     47352    4%       21     1% /data
/dev/babalv       147456    131872   11%       55     1% /baba
$ 
```

Figure 98. RX Example - aixterm

Note that a typical remote execution also utilizes LBX, security extension, and application group extension introduced on X11R6.3.

9.5.3 Security Extension

X11R6.3 provides security extension that contains new protocols to enhance X server security. Clients issue SecurityGenerateAuthorization request to X servers and if the request is successfully accepted by the servers, the clients are *trusted*, otherwise they are *untrusted*. The restrictions applied to untrusted clients include:

- Untrusted clients cannot steal keyboard input of trusted clients or interfere keyboard input of trusted clients.
- Untrusted clients cannot retrieve the image contents of a trusted clients' window.
- A certain degree of restrictions to property requests (ChangeProperty, GetProperty, and so on).
- Untrusted Clients cannot use ChangeHosts, ListHosts, and SetAccessControl.

9.5.4 Application Group Extension

By using application group extension, applications other than window managers can manage other applications' windows. The primary purpose of this extension is that such applications as Web browsers to embed or insert other applications' windows into their windows.

A special application called Application Group Leader issues AppGroupCreate request to X server followed by SecurityGenerateAuthorization request. Any applications that connect using that authorization will automatically become part of the Application Group.

When a child application (a member of the application group) creates and maps a window as a child of root window, the MapRequest and Configure request are sent to Application Group Leader application instead of sent to the window manager. The Application Group Leader application may display the window inside its window or let window manager to manage the window by issuing the map request again.

9.6 OpenGL Enhancements

The following section describes the enhancements to OpenGL.

9.6.1 OpenGL 64-bit Indirect Rendering (4.3.1)

In AIX 4.3.1, OpenGL has added 64-bit support for indirect rendering. To use this function, a new level of the xlc++ compiler is required. See /usr/lpp/OpenGL/README for more information.

9.6.2 OpenGL Performance Enhancements (4.3.2)

IBM OpenGL Versions 1.1 and 1.2 are enhanced to improve performance in several areas. Users of uniprocessor systems will note faster drawing of primitives under most conditions. All users should see improvements in:

- Throughput and cache utilization
- Latency when lighting is enabled
- Overall performance on any primitives using the new MultiDrawArray extension

9.6.3 OpenGL Version 1.2 and ZAPdb (4.3.2)

OpenGL Version 1.2, released on March 16, 1998, is the second revision since the original Version 1.0. OpenGL within AIX Version 4.3.2 is enhanced to support OpenGL Version 1.2. Several additions were made to the graphics library (GL), especially to the texture mapping capabilities and the pixel processing pipeline. Following are brief descriptions of each addition:

- Three-Dimensional-Texturing

  Three/dimensional textures can be defined and used. In-memory formats for the three-dimensional images, and pixel storage modes to support them, are also defined. One important application of three-dimensional texture is rendering volumes of image data.

- BGRA Pixel Formats

  BGRA extends the list of host-memory color formats. Specifically, it provides a component order matching file and framebuffer formats common on Windows platforms.

- Packet Pixel Formats

  Packed pixels in host memory are represented entirely by one unsigned byte, one unsigned short, or one unsigned integer. The fields with the packed pixels are not proper machine types, but the pixel as a whole is. Thus the pixel storage modes and their unpacking counterparts all work correctly with packed pixels.

- Normal Rescaling

  Normals may be rescaled by a constant factor derived from the modelview matrix. Rescaling can operate faster than renormalization in many cases, while resulting in the same unit normals.

- Separate Specular Color

  Lighting calculations are modified to produce a primary color consisting of emissive, ambient, and diffuse terms of the usual GL lighting equation, and a secondary color consisting of specular terms. Only the primary color is modified by the texture environment; the secondary color is added to the result of texturing to produce a single post-texturing color. This allows highlights whose color is based on the light source creating them, rather than the surface properties.

- Texture Coordinate Edge Clamping

  GL normally clamps such that the texture coordinates are limited to exactly the range [0,1]. When a texture coordinate is clamped using this algorithm, the texture sampling filter straddles the edge of the texture image, taking half its sample values from within the texture image, and the other half from the texture border. It is sometimes desirable to clamp the texture without requiring a border, and without using the constant border color.

  A new texture clamping algorithm, CLAMP_TO_EDGE, clamps texture coordinates at all mipmap levels such that the texture filter never samples a border texel. The color returned when clamping is derived only from the edge of the texture image.

- Texture Level of Detail Control

  Two constraints related to the texture level of detail parameter λ (lambda) are added. One constraint clamps λ (lambda) to a specified floating point range. The other limits the selection of mipmap image arrays to a subset of the arrays that would otherwise be considered.

  Together, these constraints allow a large texture to be loaded and used initially at low resolution and to have its resolution raised gradually as more resolution is desired or available. Image array specification is necessarily integral rather than continuous. By providing separate, continuous clamping of the λ (lambda) parameter, it is possible to avoid popping artifacts when higher resolution images are provided.

- Vertex Array Draw Element Range

  A new form of DrawElements that provides explicit information on the range of vertices referred to by the index set is added. Implementations can take advantage of this additional information to process vertex data without having to scan the index data to determine which vertices are referenced.

Because OpenGL 1.2 is a superset of OpenGL 1.1, all programs written for OpenGL 1.1 run on OpenGL 1.2 without modification, recompiling, or relinking. OpenGL 1.2 support is available in AIX 4.3.2. Users of other graphics adapters are limited to functions contained in OpenGL 1.1 and OpenGL 1.1 extensions. The optional Imaging Extension subset is not supported by the IBM OpenGL 1.2 implementation (at this time).

The ZAPdb interactive OpenGL debugger is enhanced to support all new features provided in OpenGL Version 1.2.

9.6.4 New OpenGL Extensions (4.3.2)

Three new extensions to OpenGL are available in AIX Version 4.3.2. For details on implementation, consult the user documentation. Brief descriptions of the new extensions follow as such:

- MultiDrawArray Extension

  Enables users to group together multiple primitives and send them to the adapter with one call. This is supported on all OpenGL-capable adapters except the GXT1000.

- Texture Mirrored Repeat Extension

  Gives users the capability of specifying texture maps without discontinuities at the edges. This is supported only on the GXT3000P.

- Color Blend Extension

  Gives users more options in creating blended or translucent colors without having to use an alpha buffer. This is supported only on the GXT3000P.

9.6.5 GLX Version 1.3 (4.3.3)

GLX1.3 provides three features that were not available in GLX1.2

- Separate read and draw drawables

  This feature allows applications to directly copy window contents for GXT2000P and GXT3000P.

- Pbuffer rendering

  A Pbuffer is an off-screen HW window. It is superior to pixmap rendering in that HW can render to it and direct rendering supported. This feature is supported for GXT2000P only.

- RGB rendering to a pseudo-color window

This feature is not supported.

9.6.6 Thread Enablement for Magician (4.3.3)

Magician is a set of OpenGL bindings for Java produced by Arcane Technologies Lts. Since Java is threaded, the underlying OpenGL implementations must also support threads. Instead of requiring thread-safe OpenGL libraries, Magician has serialized OpenGL access within the OpenGL Java class libraries. When magician detects a thread access change a glXMakeCurrent request is made. This is supported both for GXT2000P and GXT3000P.

9.6.7 Pixmap Rendering for GXT3000P and GXT2000P (4.3.3)

GXT3000P and GXT2000P had not supported OpenGL pixmap rendering because the soft rasterizer does not use the 1.7 (OpenGL 1.2) rasterizer I/F. Pixmap rendering has been enabled on these adapters by switching the entire pipeline and rasterizer from device specific version to the *soft* version when processing on a pixmap drawable. This implementation has following limitations that applications should be aware of:

- OpenGL 1.2 function is not supported when rendering to a pixmap.
- The list of supported extensions may not be the same in pixmap as in windows, so that you must query for extensions in pixmap mode separately from the queries in window mode. This limitation has always existed in pixmap rendering on other adapters.
- Context is not kept synchronized between pixmap drawables and window drawable, so the application is strongly encouraged to use separate contexts for two drawables.
- Pixmaps are only supported in indirect contexts, never in direct contexts.

9.6.8 Performance Enhancements (4.3.3)

Two generic software performance enhancements are provides:

- Perform all lighting in object space
- Optimized slow lighting

9.6.9 New OpenGL Extensions (4.3.3)

Several performance extensions have been added to OpenGL. These include:

- Vertex Array Lists

An IBM extension that enables providing a list of vertex attribute arrays with a single OpenGL function call.

- Multimode Vertex Arrays

  An IBM extension enable requesting multiple vertex arrays of varying modes using a single function all.

- Clip Volume Hint

  A multivendor extension that enables applications to notify IBM's OpenGL implementation whenever view frustum clipping is not required.

9.7 graPHIGS Enhancements

IBM graPHIGS has been enhanced in AIX Version 4.3.2 to support the new high performance GXT3000P PCI Graphics Accelerator that attaches to the RS/6000 43P 7043 Models 150 and 260. Further performance improvements were made, and support for the new Euro symbol support has been added.

9.7.1 Performance Enhancements (4.3.2)

IBM graPHIGS within AIX Version 4.3.2 has been altered to improve the performance of polygon rendering under most conditions. Throughput to the graphics adapter has also been enhanced, often resulting in performance improvements for the rendering of all primitives.

9.7.2 Performance Enhancements (4.3.3)

The following graPHIGS performance enhancements are provided with AIX4.3.3.

- Implement HW clipping on GXT2000P and GXT3000P

  Establish a bounding box which reflects the maximum window that the HW can handle rather than restrict to a view boundary.

- General performance analysis and tuning

  Tune for maximum performance for the GXT2000P and GXT3000P adapters on RS/6000 model 43P 150 and 260.

9.7.3 Euro Symbol Support (4.3.2)

The graPHIGS API offers several facilities for the display of text information:

- Geometric text and annotation text that allow an application to specify a character string that is to be displayed on a workstation.

- Echo of string device input that allows a user to see the input being entered on a string device (for example, a keyboard).

Your application can specify two attributes that affect the interpretation and display of text information: the character set identifier (CSID) and the font identifier. When the application specifies a character to be displayed, the CSID and font identifier together determine the symbol that is displayed to represent the character.

The application can specify a character string to be displayed in geometric text. The graPHIGS API, in processing that character string, accesses a character set file and a symbol file to get the information necessary to interpret and display the text:

- The character set file contains information about a particular character set. The contents of the character set file tell, among other things, the available code points of the character set and the index that is used to locate the symbol in a symbol file. There is only one character set file for a particular character set identifier.
- The symbol file contains the drawing controls for each symbol that is displayed to represent the character. There is a symbol file for each available font within a particular character set.

IBM graPHIGS will use the same locales as AIX to support the Euro symbol. For details about the Euro symbol support through AIX locales refer to 11.5, “Euro Symbol Support for AIX (4.3.2)” on page 603. graPHIGS selects the appropriate code set and font support based on the value of the LANG environment variable:

- LANG=xx_XX

  The effective character set is given by CSID 10 - ISO 8859-1 (Latin 1) with the Euro in the x80 character position. The font related symbol file is afm0a01.sym, thus the graPHIGS default FONT 1 is used.
- LANG=xx_XX.IBM-1252

  graPHIGS does not refer to the explicitly stated code set in the name of the locale. That means that the code set designation IBM-1252 is ignored and the same conditions as LANG=xx_XX apply. The code set IBM-1252 is of particular interest for the Euro symbol support in conjunction with the single-byte migration option. This option provides customers with the opportunity to get Euro symbol support through a single-byte character set instead of using the Unicode locales. For further details, refer to section 11.5.6, “Euro SBCS Migration Option - IBM-1252 Locale” on page 626.

- LANG=XX_XX

  The effective character set is given by CSID 131 - Unicode with the Euro glyph in the standard U+20AC character position. The font related symbol file is afm8301.sym, thus the graPHIGS default FONT 1 is used.

- LANG=Xx_XX

  The effective character set is PC850, and no Euro symbol support is available.

Note: xx_XX, Xx_XX, and XX_XX represent the language and territory designation with respect to a given code set. For example, the national language support for German allows you to select between the de_DE, the De_DE, and the DE_DE locale.

The graPHIGS default font files for CSID 10 have added the Euro character at codepoint 0x80. If you have used these font files as a basis for your user defined font files, you will need to evaluate your use of the codepoint 0x80.

Chapter 10. Online Documentation

There have been several changes made to the way online assistance is provided in AIX Version 4.3. The main change is the move from InfoExplorer, that is essentially proprietary in nature, to a more generalized HTML based system.

AIX Version 4.3.2 included the Documentation Search Service, but this chapter has been updated to described the new Documentation Library Service included as part of AIX 4.3.3. You also no longer need to be the root user to perform many of the tasks; you just need to belong to the imnadm group.

10.1 Unified Documentation Library Services (4.3.3)

AIX Version 4.3.3 has extended the Documentation Library Service to integrate the navigation, reading, and search of online documents. You can use these functions with a new documentation library Graphical User Interface (GUI). This new GUI offers easier access to online documentation with a single integrated graphical user interface.

The AIX operating system documentation can be accessed through this library service. Additionally, you can register locally written HTML documents into the library so that you can go to a single library GUI to access a wide range of documents. You can have a unified presentation of documents to users; that is they will only need to use one library application to find any HTML documentation that is stored on the system.

For example, as well as AIX documentation, other documents could include online documentation for customer applications and also company policies and procedures. The library services can be made available locally, or through use of a Web server, and the documents can be used remotely by AIX or any other appropriately configured network computer clients.

10.1.1 Advice to Help You Create Your Library

If you wish to create a documentation library, you should refer to the latest AIX documentation which accompanies the new Documentation Library Service Facility. Some documents you should refer to include:

- The AIX Version 4.3.3 Release notes

- The publications *AIX Version 4.3 Installation Guide*, SC23-4112 and *General Programming Concepts: Writing and Debugging Programs*, SC23-4128, currently available in two formats for you
    - An online version in the library available from the AIX home page on the Internet, which is located at http://www.ibm.com/servers/aix/library/
    - A printed version of the book that you can order from IBM.

  The *General Programming Concepts: Writing and Debugging Programs*, SC23-4128 book helps you create, index and register your own HTML documents for the library, as described in 10.1.2.4, “Add Your Own HTML Documents to the Library” on page 575. Refer to chapter 21, *Documentation Library Service.*

  Chapter 7, Installing and Configuring Documentation Library Service and AIX Online Documentation, in the book *AIX Version 4.3 Installation Guide*, SC23-4112 provides information concerning the:

    - Installation and configuration of the Documentation Library Service in AIX Version 4.3.3.
    - Installation and configuration of the Documentation Search Service in AIX Version 4.3.0 to AIX Version 4.3.2.
    - Creation and configuration of a documentation server and client system.
    - Installation and configuration of the AIX online documentation contained on the AIX Version 4.3 Base Documentation CD and the AIX Version 4.3 Extended Documentation CD.

  The most relevant new article you should start to review is *Installing the Documentation Library Service for AIX Version 4.3.3*, which includes advice to help you:

    - Test the Search Service
    - Install the Documentation Library Service with either of
        - Configuration Assistant
        - A manual installation
    - Install the AIX Documentation

10.1.2 Installing the Documentation Library Service

This section describes what you need, and then it guides through a simple example of the creation of a library that contains your own documents, which you can access as shown in the section “Using the Documentation Library” on page 581.

10.1.2.1 Requirements

You can store the library documents on a hard disk, or, if you do not have much free disk space, you can access the documents directly from the slower CD-ROM interface. You need to determine whether your environment will be:

- Standalone: Web server and browser on same machine

  You can use the Lite NetQuestion Web server which was installed with the base operating system. This server can serve documents to local users on the server computer, but cannot be used to server documents to remote users. The exception to this is if these users have access to an X server on which they can use a Web browser from the Lite NetQuestion Web server. You can check that you have installed this Web server from the AIX Installation CD-ROMs by executing:

  ```
  # lslpp -l|grep -i http
    IMNSearch.rte.httpdlite    2.0.0.1  COMMITTED  NetQuestion Web Server
  ```

- Remote: Web server and browser on different machines.

  You can set up one of your AIX systems in your organization to be the documentation server and all other systems as documentation clients. This will allow documentation to be installed only on one system and all other systems can access this system without needing the documentation installed locally.

You need the following products and components installed for a complete set of services in the previously mentioned environments:

- For the client:

  2. A Web browser. The browser must be a forms-capable browser. Your Web browser choices include:

     - Netscape
     - Internet Explorer
     - Lynx

  3. The bos.docsearch.client.* filesets (for AIX integration)

     You can invoke the Documentation Library Service without installing the docsearch client component. In fact, you do not even need to invoke the Documentation Library Service from an AIX machine. You can do this by first invoking the browser and enter the following URL:

     `http://<server_name>[:<port_number>]/cgi-bin/ds_form`

     This URL points to a global search form on the document server where the name of the remote server given in `server_name`. The `port_number` only needs to be entered if the port is different from 80.

- For the documentation server (which may also act as a client)
  1. The entire bos.docsearch package
  2. The documentation libraries (including your own HTML files)
  3. A Web browser. The browser must be a forms-capable browser. Your Web browser choices include:
     - Netscape
     - Internet Explorer
     - Lynx
  4. A Web server. The Web server must be CGI-compliant, such as:
     - The IMNSearch Lite Server
     - IBM HTTP Server software (Based on Apache). You can use the IBM HTTP Server software from one of the AIX Version 4.3 Bonus Pack CD-ROMs.
     - Apache Server
     - Lotus Go Server
     - Netscape FastTrack Server

With the exception of the search engine, the components of the Documentation Library Service are installed along with the BOS (Base Operating System). The search engine must be installed from the second CD containing the AIX operating system. After installation, you may need to configure it to ensure proper operation.

10.1.2.2 Installation

Read the documents referred to in “Advice to Help You Create Your Library” on page 565 to help ensure you fully understand and benefit from this section.

In particular, read the article *Installing the Documentation Library Service for AIX Version 4.3.3*. The following installation example follows the relevant parts of this article, starting at the section with the title *Manually Installing the Documentation Service*. This example is for a stand-alone environment, so only the server part of this environment is setup. Only some important parts of this process are described in detail here in this example. Also note that the end of the article refers you to the associated *Installing the AIX Documentation* article. There are a variety of ways to install the documentation, Web server, and Document Library Service. You can use the

Configuration Assistant TaskGuide, Web-Based System Management, or SMIT; the manual method uses SMIT.

The easiest way for a nontechnical user to install and configure Documentation Search Services, is to use the Configuration Assistant TaskGuide. To run the Configuration Assistant TaskGuide, use the `configassist` command.

The manual installation steps from the *Manually Installing the Documentation Service* article are:

1. Install the server

   You need to go through all the steps in this section for this example which is about the creation of a documentation server.

   1. Install the Web server software

      Both Web browser and Web server software are required. You can check that you have installed these products using commands similar to:

      ```
      # lslpp -l|grep -i http
        IMNSearch.rte.httpdlite    2.0.0.1  COMMITTED  NetQuestion Web Server
        http_server.admin          1.3.6.0  COMMITTED  IBM HTTP Server Administration
        http_server.base.rte       1.3.6.0  COMMITTED  IBM HTTP Server Base Run-Time
        http_server.base.source    1.3.6.0  COMMITTED  IBM HTTP Server Source Code
        http_server.frca           1.3.6.0  COMMITTED  IBM HTTP Server Fast Response
        http_server.html           1.3.6.0  COMMITTED  IBM HTTP Server Documentation
        http_server.modules.ldap   1.3.6.0  COMMITTED  IBM HTTP Server LDAP Module
        http_server.modules.mt     1.3.6.0  COMMITTED  IBM HTTP Server MT Module
        http_server.modules.snmp   1.3.6.0  COMMITTED  IBM HTTP Server SNMP Module
        http_server.msg.en_US.admin
                                   1.3.6.0  COMMITTED  IBM HTTP Server Admin Messages
        http_server.msg.en_US.html
                                   1.3.6.0  COMMITTED  IBM HTTP Server Documentation
        internet_server.base.httpd
        internet_server.msg.en_US.httpd
        IMNSearch.rte.httpdlite    2.0.0.1  COMMITTED  NetQuestion Web Server
        internet_server.base.httpd
      # lslpp -l|grep -i scape
        Netscape.msg.en_US.nav.rte
                                   4.0.7.0  COMMITTED  Netscape Navigator Runtime
        Netscape.nav.rte           4.0.7.0  COMMITTED  Netscape Navigator Runtime
        Netscape.nav.rte           4.0.7.0  COMMITTED  Netscape Navigator Runtime
      ```

      If you installed the Netscape browser from other sources or you installed other Web browsers, follow the installation instructions that come with the software. Note that there will not be any records in the ODM if your product source is not in `installp` format.

   2. Configure and start your Web server software

      The easiest setup is to use the Lite NetQuestion Web server, so you can now move to the section *Install the Documentation Search Engine*.

3. Install the Documentation Search Engine

Installation Note

Ensure that /home has enough free space to accommodate the creation of the imnadm user ID, since the install preview only checks / and /usr.

You can verify that you have the required three filesets (IMNSearch.rte, IMNSearch.bld and bos.docsearch using commands similar to:

```
# lslpp -l|grep -i imns
  IMNSearch.bld.DBCS         1.2.3.0  COMMITTED  NetQuestion DBCS Buildtime
  IMNSearch.bld.SBCS         1.2.3.0  COMMITTED  NetQuestion SBCS Buildtime
  IMNSearch.rte.DBCS         1.2.3.1  COMMITTED  NetQuestion DBCS Search Engine
  IMNSearch.rte.SBCS         1.2.3.1  COMMITTED  NetQuestion SBCS Search Engine
  IMNSearch.rte.httpdlite    2.0.0.1  COMMITTED  NetQuestion Web Server
  IMNSearch.rte.DBCS         1.2.3.1  COMMITTED  NetQuestion DBCS Search Engine
  IMNSearch.rte.SBCS         1.2.3.1  COMMITTED  NetQuestion SBCS Search Engine
  IMNSearch.rte.httpdlite    2.0.0.1  COMMITTED  NetQuestion Web Server
# lslpp -l|grep -i docsearch
  bos.docsearch.client.Dt    4.3.3.0  COMMITTED  DocSearch Client CDE
  bos.docsearch.client.com   4.3.3.0  COMMITTED  DocSearch Client Common Files
  bos.docsearch.rte          4.3.3.0  COMMITTED  DocSearch Runtime
  bos.msg.en_US.docsearch.client.Dt
                             4.3.2.0  COMMITTED  DocSearch CDE Action - U.S.
  bos.msg.en_US.docsearch.client.com
                             4.3.3.0  COMMITTED  DocSearch Common Messages -
bos.docsearch.client.Dt    4.3.3.0  COMMITTED  DocSearch Client CDE
  bos.msg.en_US.docsearch.client.Dt
                             4.3.2.0  COMMITTED  DocSearch CDE Action - U.S.
```

4. Install messages

   This example uses the English language. The previous output shows that the requirement in the section *Install the Documentation Library Service Messages for Additional Languages* is satisfied since the bos.msg.en_US.docsearch.client.com and the bos.msg.en_US.docsearch.client.Dt filesets are installed.

5. Configure the Documentation Library Service

   This is described in 10.1.2.3, “Configuration” on page 571

6. Install or register your documentation

   This is described in 10.1.2.4, “Add Your Own HTML Documents to the Library” on page 575

7. Completed Server Installation

   You can use the library as shown in 10.1.3, “Using the Documentation Library” on page 581. You do not need to follow the *Installing the Client* section advice since the client and server is the same machine in this example.

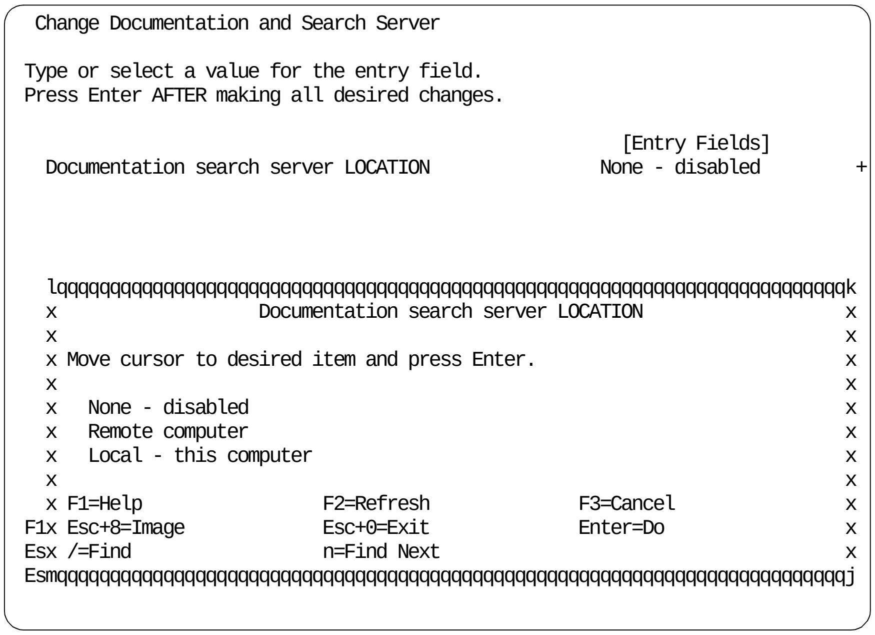
10.1.2.3 Configuration

This example uses the SMIT based method. The steps are:

1. Use the fast path `smitty Web_configure` and then:
    1. Select the option **Change / Show Default Browser** to obtain the browser selection screen which has the following field:

       ```
       * Default browser LAUNCH COMMAND                    [netscape]
       ```

    2. Press **Enter** to verify that the /etc/environment file variable is `DEFAULT_BROWSER = netscape`
    3. Press the **F3** key to return to the screen with the title Internet and Documentation Services from where you can select the option **Change Documentation and Search Server** to obtain the server selection screen.
    4. Press the **F4** key to display the screen:



```
 Change Documentation and Search Server

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

                                                        [Entry Fields]
  Documentation search server LOCATION                None - disabled        +




  lqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqk
  x                  Documentation search server LOCATION                   x
  x                                                                         x
  x Move cursor to desired item and press Enter.                            x
  x                                                                         x
  x   None - disabled                                                       x
  x   Remote computer                                                       x
  x   Local - this computer                                                 x
  x                                                                         x
  x F1=Help                F2=Refresh              F3=Cancel                x
F1x Esc+8=Image            Esc+0=Exit              Enter=Do                 x
Esx /=Find                 n=Find Next                                      x
Esmqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqj
```

You can define the documentation search server location to be:

- None - disabled
- Remote computer

  Type the remote documentation server name. The default TCP/IP port address is 80. Change it to the port address used by the documentation server. This is not applicable for this example where the server and client are the same physical system.

- Local - this computer

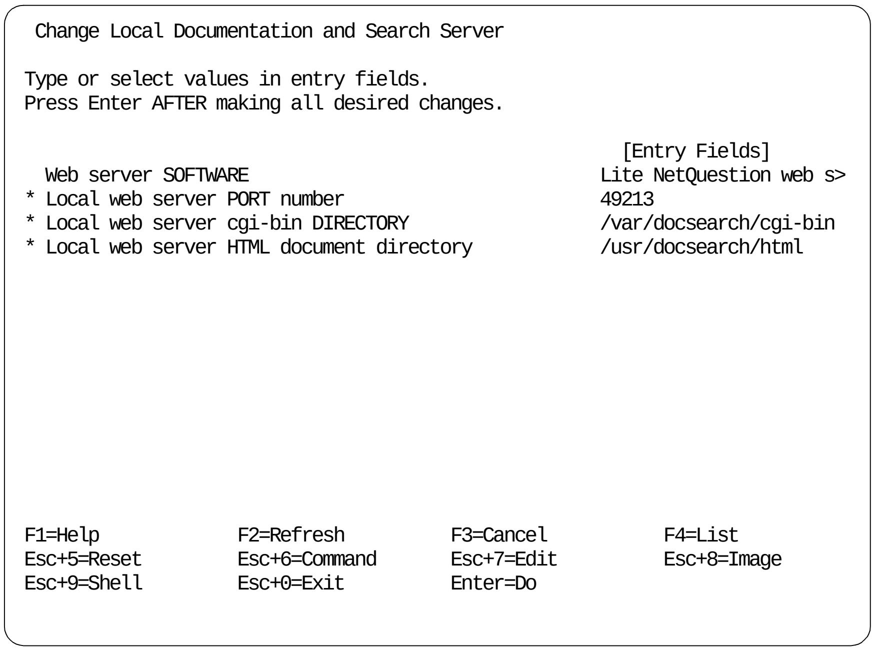
5. Use the down arrow key to move to the `Local - this computer` option. Select this option and the subsequent screen displays the server field with a default entry, which is:

```
Web server SOFTWARE                                          Lite NetQuestion web s> +
```

The Web servers you installed will determine what options appear if you select the **F4** list key. In this example, the choices are:

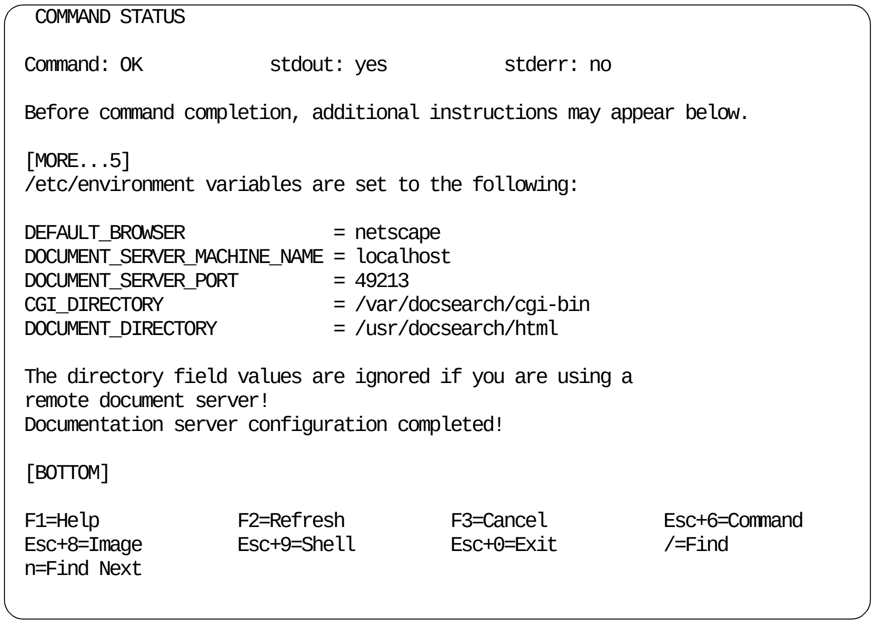
- Lite NetQuestion Web server
- IBM HTTP Server Web server in default location
- Lotus Domino Go Webserver in default location
- IBM Internet Connection Server (IICS) in default location
- Other local server or one of the above in non-default location

6. Press the **Enter** key to use the default `Lite NetQuestion web server` so that you obtain the following screen:



```
 Change Local Documentation and Search Server

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

                                                     [Entry Fields]
  Web server SOFTWARE                              Lite NetQuestion web s>
* Local web server PORT number                     49213
* Local web server cgi-bin DIRECTORY               /var/docsearch/cgi-bin
* Local web server HTML document directory         /usr/docsearch/html









F1=Help            F2=Refresh         F3=Cancel          F4=List
Esc+5=Reset        Esc+6=Command      Esc+7=Edit         Esc+8=Image
Esc+9=Shell        Esc+0=Exit         Enter=Do
```

7. Press the **Enter** key to obtain the screen that indicates the command has completed successfully, and to check the server variables:



```
 COMMAND STATUS

Command: OK            stdout: yes           stderr: no

Before command completion, additional instructions may appear below.

[MORE...5]
/etc/environment variables are set to the following:

DEFAULT_BROWSER              = netscape
DOCUMENT_SERVER_MACHINE_NAME = localhost
DOCUMENT_SERVER_PORT         = 49213
CGI_DIRECTORY                = /var/docsearch/cgi-bin
DOCUMENT_DIRECTORY           = /usr/docsearch/html

The directory field values are ignored if you are using a
remote document server!
Documentation server configuration completed!

[BOTTOM]

F1=Help             F2=Refresh          F3=Cancel           Esc+6=Command
Esc+8=Image         Esc+9=Shell         Esc+0=Exit          /=Find
n=Find Next
```

8. Exit SMIT with the **F10** function key.

2. You should verify that your Web server daemon is currently active. For this example, you can use the command:

```
# ps -ef|grep httpdlite
    root 24500 26862   2 18:26:39  pts/7  0:00 grep httpdlite
```

- If its not active, you need to start it. The inittab file may contain the daemon activation command. Execute it manually or reboot your system. You can search the inittab file with the command:

```
# grep htt /etc/inittab
ihshttpd:2:wait:/usr/HTTPServer/bin/httpd > /dev/console 2>&1 # Start HTTP daemo
n
rchttpd:2:wait:/etc/rc.httpd > /dev/console 2>&1 # Start HTTP daemon
httpdlite:2:once:/usr/IMNSearch/httpdlite/httpdlite -r /etc/IMNSearch/httpdlite/
httpdlite.conf & >/dev/console 2>&1
#
```

- If necessary, execute the following command using nohup and & to ensure it executes correctly in the background, as a daemon should.

```
# nohup /usr/IMNSearch/httpdlite/httpdlite -r /etc/IMNSearch/httpdlite/httpdlite.conf
```

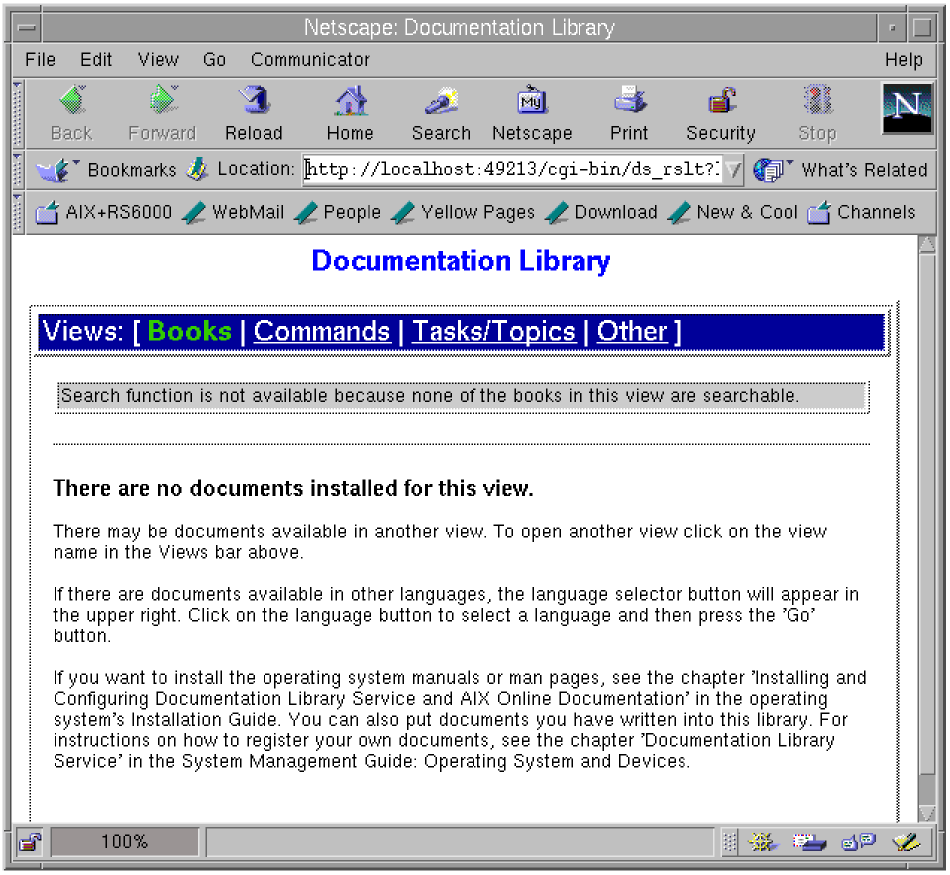
The Documentation Library is now ready to accept documents.

For the documentation clients, you need only a Web browser. Installation of the bos.docsearch.client fileset will give you the CDE desktop icon and the `docsearch` command.

If you have your own HTML document, you can follow this section’s example as it is continued in 10.1.2.4, “Add Your Own HTML Documents to the Library” on page 575.

Alternatively, if you want to immediately verify that your library search service is installed and configured, you can follow the advice in the section *Testing the Documentation Library Service*. This requests you to test the `docsearch` command, and provides advice for subsequent error messages. The `docsearch` command should behave as shown in 10.1.3, “Using the Documentation Library” on page 581.

Even if no documents are installed, it should display a window similar to that in 99 on page 575.


Netscape: Documentation Library
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://localhost:49213/cgi-bin/ds_rslt?
What's Related
AIX+RS6000 WebMail People Yellow Pages Download New & Cool Channels
100%

Documentation Library

Views: [Books | Commands | Tasks/Topics | Other]

Search function is not available because none of the books in this view are searchable.

There are no documents installed for this view.

There may be documents available in another view. To open another view click on the view name in the Views bar above.

If there are documents available in other languages, the language selector button will appear in the upper right. Click on the language button to select a language and then press the 'Go' button.

If you want to install the operating system manuals or man pages, see the chapter 'Installing and Configuring Documentation Library Service and AIX Online Documentation' in the operating system's Installation Guide. You can also put documents you have written into this library. For instructions on how to register your own documents, see the chapter 'Documentation Library Service' in the System Management Guide: Operating System and Devices.

Figure 99. A Newly Installed Documentation Library with No Documents

10.1.2.4 Add Your Own HTML Documents to the Library

This section follows the advice in Chapter 21, *Documentation Library Service,* of the publication *General Programming Concepts: Writing and Debugging Programs*, SC23-4128, to enable you to add your own HTML documents to the Documentation Library. It steps through the addition of some redbook related files, in HTML format, into the library.

HTML Documents

Currently, the Documentation Library Service supports searching HTML documents that are written using the languages and codesets listed in the Language Support Table. All documents in a single index must be written using the same language and codeset.

Specifically, you can use the following steps based on the steps in the chapter's topic *4.Creating Indexes of your Documentation*. You should read all of the steps before trying this to do this procedure

1. Choose a unique index name

   This example will index 4 HTML files in the directory path /docbuild. The documents are in English, so an eight character unique index name can be *999ab1en*.

2. Create a new directory

   The build directory is /docbuild. The HTML documents intended for the library need to go under this directory, with the common top directory as specified by the index name. So the directory /docbuild/999ab1en contains:

   ```
   # ls /docbuild/999ab1en
   index.htm      navigate.htm   sg242014.htm   start.htm
   ```

   The `index.htm` file happens to be the name of a file to be added to the library; it is not created by the document library index generation process.

3. Create an ASCII file

   One way to create the document list file for the 999ab1en index is to execute:

   ```
   # cd /docbuild
   # find /docbuild/999ab1en -print > ./doclist.file
   ```

   This method ensures that the file `doclist.file` contains full path names, but the file still needs to be edited to

   - Remove the directory /docbuild/999ab1en
   - Replace each / after docbuild with @ so that the file contains:

     ```
     # cat doclist.file
     /docbuild@999ab1en/index.htm
     /docbuild@999ab1en/navigate.htm
     /docbuild@999ab1en/sg242014.htm
     /docbuild@999ab1en/start.htm
     #
     ```

4. Choose a title for your index

   For this example, the title of the index is My Favorite Redbooks

5. Create an empty index

   If required, add your user ID to the imnadm group. Then set the ownership of the indexes directory as follows:

   ```
   chown  imnadm:imnadm  /usr/docsearch/indexes
   ```

English is a single-byte language, so you can create the index by typing, as the AIX documentation suggests, the following three command parts, all on one command line with a space between each part (Include the WT at the end of the third part.):

```
/usr/IMNSearch/cli/imnixcrt 999ab1en
/usr/docsearch/indexes/999ab1en/data
/usr/docsearch/indexes/999ab1en/work WT
```

Not all the command fits on one line, so you may prefer to use the backslash (\) command continuation character. This allows you to enter the command in three lines, as shown in the following example:

```
/usr/IMNSearch/cli/imnixcrt 999ab1en \
/usr/docsearch/indexes/999ab1en/data \
/usr/docsearch/indexes/999ab1en/work WT
Index creation successfully done.

#
```

You can confirm that your index is listed with the Documentation Library Service by executing (You may see other indexes on your system.):

```
# /usr/IMNSearch/cli/imnixlst
Index list fetch successful!

List of indexes:
999AB1EN
```

6. Add your documents to the update list

Now use the ASCII file that contains the list of all HTLM documents as a parameter to the `imdoadd` command, so that these files can be added to the list of files to be indexed. Execute:

```
# /usr/IMNSearch/cli/imndoadd 999ab1en /docbuild/doclist.file
Document scheduling ended
```

Verify this step by executing:

```
# /usr/IMNSearch/cli/imnixsta 999ab1en
Function status:
  Search:   available
  Update:   available
  Merge :   available
Active processes:
  Update: No
  Merge : No

Documents in index:
  primary  : 0
  secondary: 0
  scheduled: 4
```

This shows that 4 HTML documents are scheduled to be indexed.

7. Start the index updating process to build your index. Execute:

```
# /usr/IMNSearch/cli/imnixupd 999ab1en
Document Indexing process started!
```

This command works in the background and you must wait until it finishes. As suggested by the AIX documentation, execute the following command regularly until you can see that there are no more documents in the `scheduled` status:

```
# /usr/IMNSearch/cli/imnixsta 999ab1en
Function status:
  Search:   available
  Update:   available
  Merge :   available
Active processes:
  Update: No
     Last start: 1999/08/26 14:32:19  ended: 1999/08/26 14:32:29
  Merge : No

Documents in index:
  primary  : 4
  secondary: 0
  scheduled: 0
```

This confirms that four HTML documents have been indexed, but note the following advice about this command’s possible failure:

Index Creation Failure

If this command does not show four documents in the `primary` part of the index, then before repeating all the previous steps, you should check your system; for example:

- Ensure they contain the @ character to split the file name into two parts.
- Ensure the files to be indexed and the /usr/docsearch/indexes directory have read access for the imnadm user.
- Check your file systems are not full.

If you want to delete an index named abs206en, you can execute:

```
# /usr/IMNSearch/cli/imnixdel abs206en
Index deleted successfully!
# /usr/IMNSearch/cli/imnixlst
No indices available.

/usr/IMNSearch/cli/imnixlst: List is empty.
#
#  ls /usr/doc*/ind*
abs206en
# rm -r /usr/docsearch/indexes/abs206en
```

You can now return to the previous steps. You may also want to change the index name to have a clean start to this procedure, but then you will need a new doclist.file file and thus start from the beginning of this procedure. Otherwise, return to 5 on page 576.

8. Update the registration table

   The command to register this index in the registration table is perhaps the most difficult part of this procedure. As the AIX documentation explains, the doc_link part of the command is related to your Web server's home directory. For the Lite NetQuestion Web server used in this example, which has DOCUMENT_DIRECTORY = /usr/docsearch/html, doc_link is the link to /usr/share/man/info, as can be seen from:

```
# ls -l /usr/docsearch/html/doc*
lrwxrwxrwx   1 root     system       19 Aug 26 13:15
/usr/docsearch/html/doc_link -> /usr/share/man/info
```

   The registration command uses doc_link, not the associated file names.

AIX Documentation Update

Under *Building the indexes*, in the section titled *Update the registration table* of the AIX documentation that you are using (see 10.1.1, “Advice to Help You Create Your Library” on page 565), the single-byte syntax should read:

```
/usr/IMNSearch/cli/imndomap
     /var/docsearch/indexes  -c  index_name
     /doc_link/locale/application_name/ index_title
```

The double-byte syntax should read:

```
/usr/IMNSearch/cli/imqdomap
     /var/docsearch/indexes  -c  index_name
     /doc_link/locale/application_name/ index_title
```

Note the space between the last slash after `application_name`, and the `index_title`; this space may not be present in your version of the documentation. Each of the three lines in these syntax guides should be entered as one command, with a space character between each line. Alternatively, you can use the backslash (\) line continuation character.

The explanation of the example in the AIX documentation may not be clear to you. The line before the command text currently reads as:

(Type the following three command parts, all on one command line with a space between each part):

It is true that the example that follows contains three parts which need to be entered as one command. The third part starts on the third printed line with the text /doc_link, and there is a space after the character sequence calculator/ since the application_name is calculator. The final variable, index_title, is all the HTML definition in the character sequence from "<A to A>". The full correct example from the AIX documentation is:

```
/usr/IMNSearch/cli/imndomap
/var/docsearch/indexes  -c  cal413en
/doc_link/en_US/calculator/
"<A HREF='/doc_link/en_US/calculator/cal413en/user/doc1.htm'> Calculator
```

For this example, execute the following `imndomap` command. It is listed in four lines, and the backslash character joins the lines to form one command.

The output of the successful command is `NetQ function completed.` Note that the command part that starts with /doc_link does not imply that there is a real directory named /doc_link. /doc_link is not a variable part of this command's syntax.

```
# /usr/IMNSearch/cli/imndomap \
/var/docsearch/indexes -c 999ab1en /doc_link/en_US/redbooks/ \
"<A HREF='/doc_link/en_US/redbooks/999ab1en/index.htm'>My \
Favorite Redbooks</A>"
NetQ function completed.
```

You may get an informational message similar to:

```
NetQ warning: file not found or empty.
NetQ function completed.
```

if this is the first index that you are registering. Now copy the new index registration table over the backup copy of the index registration table with the command:

```
# cp  /var/docsearch/indexes/imnmap.dat  /usr/docsearch/indexes
```

9. Copy your HTML documents from the build directory into the documentation directory. Execute the commands:

```
# mkdir /usr/share/man/info/en_US/redbooks
# mkdir /usr/share/man/info/en_US/redbooks/999ab1en
# cp /docbuild/* /usr/share/man/info/en_US/redbooks/999ab1en
# ls /usr/share/man/info/en_US/redbooks/999ab1en
index.htm      navigate.htm   sg242014.htm   start.htm
```

10. Test your index. The test process for this example is described in 10.1.3, "Using the Documentation Library" on page 581.
11. The final step in the AIX documentation is not required for this example since the development computer is also the documentation server.

10.1.3 Using the Documentation Library

You can navigate, read, and search registered HTML documents using your Web browser and the library application. This application presents two type of GUIs to users; a global GUI and an application GUI. The AIX global GUI shows users all HTML documents on the document server that are registered with the global GUI. The global views may contain documents from many different applications. You can access the global library application GUI by either of:

- Typing `docsearch` on the command line
- Clicking the **Documentation Library** icon in the **Help** sub panel under the **CDE Desktop** front.

The global library application presents documents in a expandable-tree format that can be navigated by clicking on button controls in the tree. For example, if you followed the steps in 10.1.2, “Installing the Documentation Library Service” on page 566 to index your own HTML documents but had no AIX documentation installed, then the library tree is shown in Figure 100.

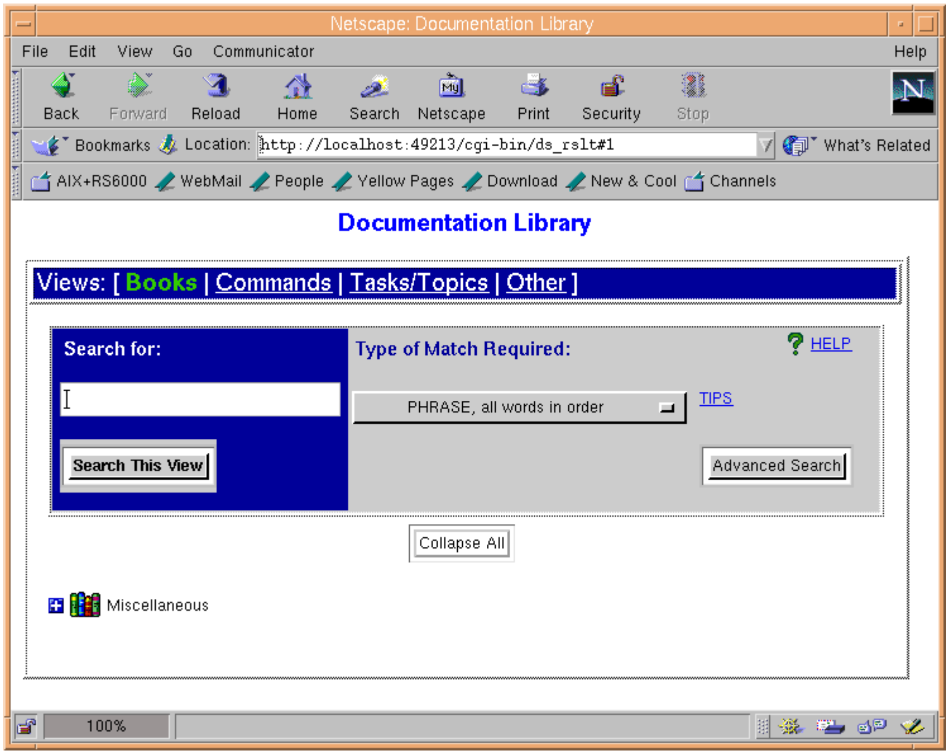

Netscape: Documentation Library
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://localhost:49213/cgi-bin/ds_rslt#1 What's Related
AIX+RS6000 WebMail People Yellow Pages Download New & Cool Channels
Documentation Library
Views: [Books | Commands | Tasks/Topics | Other]
Search for:
Type of Match Required:
HELP
PHRASE, all words in order
TIPS
Search This View
Advanced Search
Collapse All
Miscellaneous
100%

Figure 100. Closed Library Tree

You can expand the tree by clicking on the **plus sign in the blue box** next to the books in Figure 100. This results in the open tree in Figure 101 on page 583.

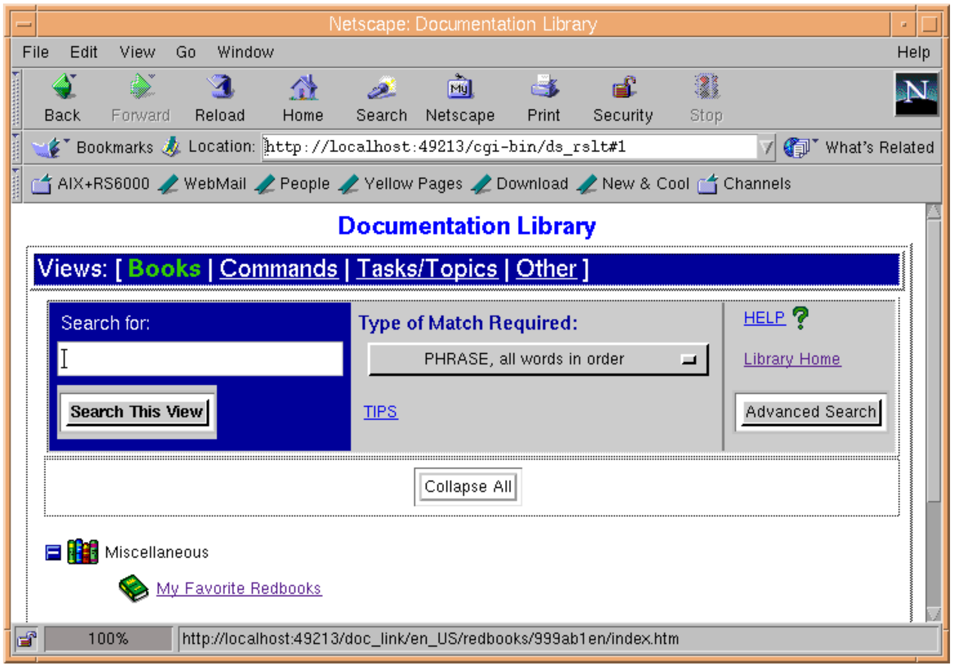

Netscape: Documentation Library
File Edit View Go Window Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://localhost:49213/cgi-bin/ds_rslt#1 What's Related
AIX+RS6000 WebMail People Yellow Pages Download New & Cool Channels
Documentation Library
Views: [Books | Commands | Tasks/Topics | Other]
Search for:
Search This View
Type of Match Required:
PHRASE, all words in order
TIPS
HELP
Library Home
Advanced Search
Collapse All
Miscellaneous
My Favorite Redbooks
100%
http://localhost:49213/doc_link/en_US/redbooks/999ab1en/index.htm

Figure 101. Open Library Tree

This expansion of the tree opens a list of HTML documents. In Figure 101, you could click on the title My Favorite Redbooks to go directly to the HTML document it is linked to, the file index.htm. You can see this link displayed at the bottom of the browser window in Figure 101, since this image was saved with the mouse pointer over the hypertext link title My Favorite Redbooks. This is the title that was chosen in 4 on page 576.

The global library GUI also allows you to search using keywords in a search form present in the application. Figure 102 on page 584 shows the closed library GUI with word `differences` in the white Search for: field.

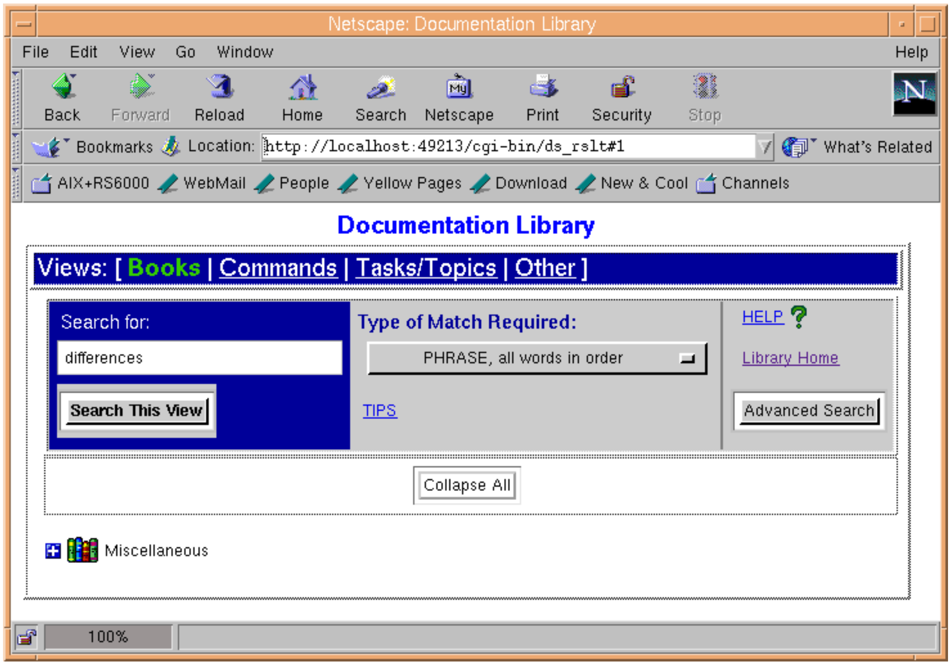
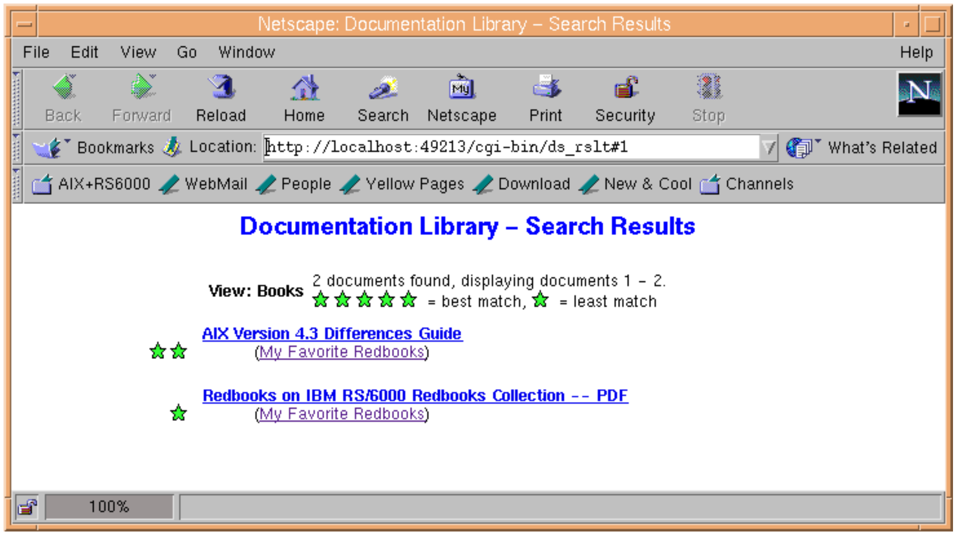

Netscape: Documentation Library
File Edit View Go Window Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://localhost:49213/cgi-bin/ds_rslt#1 What's Related
AIX+RS6000 WebMail People Yellow Pages Download New & Cool Channels
Documentation Library
Views: [Books | Commands | Tasks/Topics | Other]
Search for:
differences
Search This View
Type of Match Required:
PHRASE, all words in order
TIPS
HELP
Library Home
Advanced Search
Collapse All
Miscellaneous
100%

Figure 102. How to Search the Library

The Documentation Library Service searches for the keywords, and then presents a list of results that are linked to information contained in the online documentation files. You can then click on **Search This View** to obtain the results shown in Figure 103 on page 585.


Netscape: Documentation Library – Search Results
File Edit View Go Window Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://localhost:49213/cgi-bin/ds_rslt#1
What's Related
AIX+RS6000 WebMail People Yellow Pages Download New & Cool Channels
100%

Documentation Library – Search Results

View: Books 2 documents found, displaying documents 1 - 2.
★★★★★ = best match, ★ = least match

★★ AIX Version 4.3 Differences Guide
(My Favorite Redbooks)

★ Redbooks on IBM RS/6000 Redbooks Collection -- PDF
(My Favorite Redbooks)

Figure 103. Documentation Library Search Results

These are two of the HTML documents that were indexed in the procedure in 10.1.2.4, “Add Your Own HTML Documents to the Library” on page 575.

As well as the global library GUI that you obtained from the docsearch command, there is also a second type of new GUI, know as the application GUI. The application GUIs may be launched from links inside the menus or document pages of an application. The library pages displayed contain information relating to the application. For example, the Search link in the Web-Based System Manager Help menu calls a library page that only displays the documentation for Web-Based System Manager.

10.2 Installing Online Manuals

You can either install the documentation information onto the hard disk or mount the documentation CD in the CD-ROM drive. Mounting the CD will save some amount of hard disk space, but it requires the CD to be kept in the CD-ROM drive at all times. Also, searching the documentation from the CD-ROM drive can be significantly slower (in some cases up to 10 times slower) than searching the information if it is installed on a hard disk. In addition, there are two documentation CDs:

- The AIX Version 4.3 Base Documentation CD
- The AIX Version 4.3 Extended Documentation CD

Use `smit install_latest` to install the online manuals onto the hard disk. The fileset bos.docregister is a prerequisite for all online manuals. It will be automatically installed the first time you install any online manuals even if you have not selected this fileset.

Note

The installation images located on the AIX Version 4.3 Base Documentation and Extended Documentation CDs do not contain the HTML files. These files exist separately on the CD to allow access from non-AIX platforms. Installing the images from the CD will work correctly; however copying the installation images by themselves to another location is not enough for a proper install.

10.3 Help for Library Problems

If you have not run Netscape previously, a lot of informational messages and windows will be shown while Netscape is setting up the environment in your home directory. This is standard behavior for the first execution of Netscape. The messages will not be shown again the next time you start Netscape.

You must logout and login again after the Documentation Library Service has been configured so that you will pick up the environment variables set up during the configuration.

If you have a problem starting the Documentation Library Service, check the following environment variables. These environment variables may be set, displayed, and changed using SMIT. Start SMIT, select **System Environments**, then select **Internet and Documentation Services**.

- On the client machine
  1. Invoke the Web browser manually and enter the URL `http://<server_name>[:<port_number>]/cgi-bin/ds_form` to ensure that the server is up and running.
  2. Ensure the DEFAULT_BROWSER variable is set to the command for starting your Web browser.

     Use the command `echo $DEFAULT_BROWSER` to obtain the command used in starting the browser. Test whether that command can bring up the browser by manually entering it on the command line.
  3. Ensure the DOCUMENT_SERVER_MACHINE_NAME variable is set to the document server’s host name or IP address.

4. Ensure the DOCUMENT_SERVER_PORT variable is set to the port address used by the document server's port address.

- On the server machine

1. Ensure the DEFAULT_BROWSER variable is set to the command for starting your Web browser.

   Use the command `echo $DEFAULT_BROWSER` to obtain the command used in starting the browser. Test whether that command can bring up the browser by manually entering it on the command line.
2. Ensure the DOCUMENT_SERVER_MACHINE_NAME variable is set to the local hostname.
3. Ensure the DOCUMENT_SERVER_PORT variable is set to the port address used by the local Web server.
4. Ensure that the CGI_DIRECTORY variable is set to the correct cgi-bin directory used by the local Web server.
5. Ensure that the DOCUMENT_DIRECTORY is set to the directory where the symbolic links `doc_link` and `ds_images` reside. If you have not changed the default, it should be in /usr/lpp/internet/server_root/pub for both IBM Internet Connection Server and Lotus Domino Go Web Server.
6. If you are not using the default directory, ensure that you have defined the necessary directory mapping in your Web server configuration file such that the directory can be resolved.

10.4 Internationalization

At the time of writing, the Documentation Library Service has language support for several languages and codesets. For a list of supported languages, codesets, and locales, see the language support table section in *AIX Version 4.3 General Programming Concepts: Writing and Debugging Programs*, SG23-2533.

Web browsers also have a set of supported languages and codesets. You should be aware that differences between the client and server's language and codeset may cause some documents to become difficult to read.

10.5 Man Page Changes

Prior to AIX Version 4.3, the `man` command extracted its information from the InfoExplorer database. In AIX Version 4.3, the traditional InfoExplorer LPP is no longer part of the base. The replacement of InfoExplorer with HTML-based documentation has brought about a change in the way the `man` command now operates.

Previously, the `man` command had to search the InfoExplorer database to locate the information that it required. Now under AIX Version 4.3, the information is stored in HTML files. The HTML files are stored in the directory structure by `COLLECTION_ID` and `BOOK_ID`. The `man` command uses this structure to locate the required information. This new structure provides the user with performance benefits, because the search criteria is more narrowly defined.

In terms of overall function, the `man` command remains unchanged, performing the way it always has.

10.6 SMIT Documentation

With the introduction of HTML-based product libraries, the SMIT documentation and all of the SMIT helps have been converted to HTML. SMIT is updated to access the HTML versions of the help files. The naming convention of the filesets has also been changed to reflect the introduction of Web-Based System Management.

Chapter 11. National Language Support

The AIX operating system has no built-in dependencies on code set, character classification, character comparison rules, character collation order, monetary formatting, numeric punctuation, date and time formatting, or the text of messages. The national language support (NLS) environment is defined by a combination of language and geographic or cultural requirements. These conventions consist of four basic components:

- Translated language of the screens, panels, and messages
- Language convention of the geographical area and culture
- Language of the keyboard
- Language of the documentation

Customers are free to mix and match the above components for each user. The NLS environment allows AIX to be tailored to the individual user's language and cultural expectations.

To support this design, applications use standard APIs to display messages and handle characters in code set independent fashion. Additionally, libraries hide all code set-independent processing and do not alter the locale set by applications.

11.1 National Language Character Handling

The NLS feature of AIX Version 4.3 allows input and output of national language (NL) characters. NL-specific cultural conventions can be set by the user from the command line or by an application on a per-process basis. These cultural conventions include the territory-unique ways to represent date, time, monetary values, numbers, and collating sequences. By setting the appropriate environment variables, users can define their own NL behavior. Individual users may even operate using different locales, keyboards, and language text on the same system.

11.2 Levels of NLS Enablement

AIX provides the following levels of national language support enablement. To determine the level of NL support provided by any particular IBM licensed product, consult the program product announcement material. To assist in classifying the extent to which a product provides national language support, the levels are listed following in descending order of national language capability:

Universal Language Support (ULS)
Support for multiple languages based on universal character set ISO 10646. See Section 11.3, “Unicode” on page 590 for more information.

Full International Language Support (ILS)
Support for all locales in the underlying operating system.

Multi-Byte Character Set Support (MBCS)
Support for all locales based on multi-byte and single-byte code sets in the underlying operating system. Bidirectional code set support is limited.

Single-Byte Character Set Support (SBCS)
Support is limited to single-byte locales in the underlying operating system. Bidirectional code set support is limited.

PASSTHRU
Support is limited to the ability of the product to pass data through the program without processing. The information is handled in such a manner that all data, control, and graphics characters flow unaltered through the program directly to its output.

The following products and commands do not support full ILS:

- Adobe Acrobat Reader supports PASSTHRU only graphics, tplot, graph, and spline commands.
- Netscape Navigator supports PASSTHRU only.
- NFS supports PASSTHRU only.
- NCS supports PASSTHRU only.
- TCP/IP `telnet` command does not support NLS.

11.3 Unicode

The Universal Coded Character Set (UCS), or Unicode (UCS-2), is a character code designed to encode text for display and storage in computer-based files. The UCS global character encoding was developed jointly by the computer industry and the International Organization for Standardization (ISO) and defines the state of the art for character handling.

The design of the Unicode standard is based on the simplicity and consistency of today's prevalent character code set, ASCII (and Latin-1, an extended version of the ASCII code set), but goes beyond ASCII's limited ability to encode only the Latin alphabet. The Unicode encoding provides the capability to encode all of the characters used by all the principle written languages throughout the world.

To accommodate the many thousands of characters used in international text, the Universal Coded Character Set was developed and implemented in two variations:

UCS-2	A 16-bit code
UCS-4	A 32-bit code

However, UCS-2, the 16-bit code (Unicode), has already been established as the predominant code, while the 32-bit code is of limited practical interest. The Unicode implementation keeps character coding simple and efficient since the Unicode standard assigns each character a unique 16-bit value. It does not require complex modes or escape codes to specify modified characters or special cases.

The development of UCS on AIX Version 4.3 is based on Unicode code set Version 2.0, which is a widely accepted standard for encoding international character data. The AIX implementation of Unicode (UCS-2) will allow the user to process data from any of the supported Unicode scripts and to mix and match characters from differing language scripts within the same session.

Note: The terms UCS-2 and Unicode will be used interchangeably throughout this documentation.

The Unicode 2.0 standard (ISBN 0-201-48345-9) is the basis for the AIX implementation and is used as the primary reference.

Further information is also available at the Unicode URL:
http://www.unicode.org

11.3.1 UTF-8

For file systems to be able to work with Unicode, a UCS Transformation Format was developed by the X/Open Internationalization (also known as I18N, because there are 18 letters between the I and the N) working group. This transformation, called UTF-8, provides a way of transforming Unicode into an ASCII representation.

The UTF-8 transformation is important when considering that the majority of file systems are ACSII-based. When using the UTF-8 Transformation Format, you are assured of a file system-safe way of using Unicode to store files.

The UTF-8 transformation format is a multi-byte code set capable of encoding the same set of characters of UCS-2 in 1 to 3 bytes or UCS-4 in 1 to 6 bytes. It has the following characteristics:

- ASCII (7-bit code) is a proper subset.
- It preserves the semantics of the portable character set.
- It preserves the semantics of a null octet for the C programming language.

Note: It is expected that UTF-8 will be a dominant transformation method where the UCS is not practical.

11.3.2 ULS

ULS, or Universal Language Support, refers to a toolkit of functions that enable applications to work with UCS. The ULS is designed to be operating system-independent, so the functions can be ported to any system.

The Universal Language Support specification describes a set of functions and utilities for addressing a wide range of national language support (NLS) problems. In summary, it provides:

- A universal coded character set capable of encoding most of today's languages and scripts.
- A set of universal language support functions for the processing of input and output of supported characters.
- A set of universal locale objects that describe processing of text on a global scope.
- A set of universal layout objects that describe processing of text on a global scope.
- A set of universal conversion objects for import/export of traditional data into the ULS environment.

This specification identifies the level of support expected for the Universal Coded Character Sets in the AIX operating system and all future releases of it. Without abandoning its standards-based internationalization support, ULS will add support for a set of UCS locales as an extension to the existing internationalization language support.

11.3.3 Universal Locale

The Universal Locale is a method of using Unicode as a wchar_t or wide character encoding. As mentioned earlier, Unicode is based on a 16-bit encoding, whereas an ASCII character is based on an a 7-bit encoding.

For AIX to use Unicode across the entire system, it uses the UTF-8 encoding for files and uses Unicode as the process code. Process code is the actual code that is being executed in memory.

Note: All supported locales will function correctly in both their current code as well as UTF-8.

11.3.3.1 Locale Definitions

The following can be said about locale definitions:

- Unicode-based locales must be compiled based on a UTF-8 charmap containing all characters currently defined in Unicode 2.0. The names for such characters must be the exact character names as outlined in the standard, with any spaces imbedded in the character name changed to underscores. For example, the correct symbol name for the Unicode value U+00A1 would be <INVERTED_EXCLAMATION_MARK>.
- Character symbol names that are currently required to be defined by the `localedef` command must be defined with both the Unicode and POSIX symbol names in the charmap. For example, the value U+0041 would be defined as both <A> and <LATIN_CAPITAL_LETTER_A>.
- To ensure that multiple locales can be built from a given locale, all existing charmaps must be modified to use the Unicode names. Each charmap will define only those actually contained within the code set.

Many of the locale definitions have similar characteristics across multiple locales, especially in the LC_CTYPE section. From a maintenance perspective, it is highly desirable to be able to maintain a common source for these and allow them to have an include facility. The C preprocessor can be used for this purpose, provided that the localedef comment character and escape character (line continuation) be changed to values that do not conflict with the C preprocessor.

Currently, the default comment character for locales is "`#`", and the default continuation character is "`\`". By changing these to "`*`" and "`/`", respectively, you can use C preprocessor directives in the context of a localedef source file, such as `#include` and `#ifdef` statements to tailor the locale definition based on the characters available in the codeset. Thus, before running `localedef`, each locale source is run through the C preprocessor with the code set name to be used specified as an option to the C preprocessor through `-D` on the command line. For example, suppose your locale definition was to be compiled with four different code sets, ISO8859-1 (Latin), ISO8859-5 (Cyrillic), ISO8859-7 (Greek), and UTF-8 (Unicode). Moreover, suppose that in the LC_CTYPE section you wanted to declare the following three characters as uppercase:

```
<LATIN_CAPITAL_LETTER_A>

<GREEK_CAPITAL_LETTER_ALPHA>

<CYRILLIC_CAPITAL_LETTER_A>
```

This presents a bit of a problem since <LATIN_CAPITAL_LETTER_A> is defined in all four codesets, <GREEK_CAPITAL_LETTER_ALPHA> is only defined in ISO8859-7 and UTF-8, and <CYRILLIC_CAPITAL_LETTER_A> is defined only in ISO8859-5 and UTF-8. Such a situation could be coded as follows with the preprocessor directives:

```
upper /
<LATIN_CAPITAL_LETTER_A>/
#if defined(ISO88597) || defined(UTF8)
;<GREEK_CAPITAL_LETTER_ALPHA>/
#endif
#if defined(ISO88595) || defined(UTF8)
;<CYRILLIC_CAPITAL_LETTER_A>
#endif
```

Using such techniques, it is possible to create a locale definition that tailors itself to the code set being compiled against while not requiring you to define dummy codepoints in the charmap. This technique also fulfills the requirement that all locales for a given country be generated from the same locale source definition, so that consistency is achieved within a given language/territory combination.

To support Unicode symbol names in charmaps and localedef source files, the `localedef` command has been modified to accept symbol names of at least 85 characters instead of the current limit of 32. Currently, the longest symbol name from the Unicode standard is 83 characters long, not including the required < and > delimiters. This is a simple modification to symtab.h and has been done without any impact on performance, as the symbol names are only used during locale compilation, not at run time.

11.3.3.2 Locale Methods

The current set of locale methods, as provided in libc, was implemented based on the locales supported in AIX Version 3.2. Since that time, many new locales have been added to the system, mostly with corresponding locale-specific methods. It is a goal of this design to minimize the number of locale-specific method objects required and also to provide a framework under which future AIX locales can be easily added.

Locale-specific methods that are no longer used will remain in the libc.a library in order to insure binary compatibility. The following is the set of locale-specific methods that are provided with AIX Version 4.3. These locale methods have been placed into libi18n.a, instead of libc.a, to minimize incompatibilities with previous releases.

mblen()

Determine the length (in bytes) of a multi-byte character. This is the most difficult method to generalize. The following locale-specific variants are supported:

__mblen_sb() - For single-byte code sets, always returns 1.

__mblen_utf() - multi-byte determination rules for UTF-8-based locales.

__mblen_std() - For all other code sets, use the __mbtowc_std() method to determine the number of bytes in the character by calling the appropriate iconv() converter and seeing how many bytes were able to be converted.

mbstopcs() Convert multi-byte string to process code string. This function is somewhat obsolete, as mbstowcs() is the preferred alternative. There is one standard locale method, __mbstopcs_std(), which loops through the multi-byte string calling mbtopc() to convert each multi-byte character to the appropriate process code.

mbstowcs() Convert multi-byte string to wide character string. There is one standard locale method, __mbstowcs_std(), which loops through the multi-byte string calling mbtowc() to convert each multi-byte character to the appropriate wide character.

mbtopc() Convert multi-byte character to process code. This function is somewhat obsolete, as its function is duplicated by the mbtowc() function. There is one standard locale method, __mbtopc_std(), which calls mbtowc() to perform the intended function of converting the multi-byte character to process code (wide character).

mbtowc() Convert multi-byte character to wide character. All of the provided mbtowc() methods convert the multi-byte character to a Unicode value. There are three different methods provided in order to optimize performance of this operation based on the nature of the multi-byte codeset:

__mbtowc_iso1() - Since ISO8859-1 is a proper subset of Unicode. Its data can be converted to Unicode by simply casting its 8-bit value to a 16-bit value. This is the fastest method for converting ISO8859-1 to Unicode.

__mbtowc_utf() - UTF-8-based data can be converted to Unicode by using a simple bit-shifting algorithm. This method should be used for all UTF-8 based locales since it is faster than the __mbtowc_std() method listed below.

__mbtowc_std() - For all other codesets, the __mbtowc_std() method converts the multi-byte data to Unicode using the iconv() interface. The conversion descriptor is defined as a static pointer so that multiple calls to __mbtowc_std() will not incur the overhead of multiple iconv_open() calls. Because this locale method is dependent on libiconv, it has been placed into a common locale method object in /usr/lib/nls/loc/methods/stdmeth.o instead of in the libc library, to avoid creating an unnecessary dependency between libc and libiconv.

pcstombs() This method is obsolete since its function has been replaced by the method wcstombs(). Currently, this function just returns `-1`.

pctomb() This method is obsolete since its function has been replaced by the method wctomb(). Currently, this function just returns `-1`.

wcstombs() Convert wide character string to multi-byte string. There is one standard locale method __wcstombs_std(), which loops through the wide character string calling wctomb() to convert each wide character to the appropriate multi-byte string.

wctomb() Convert wide character to multi-byte character. All of the provided wctomb() methods convert a Unicode value to the appropriate multi-byte string. There are three different methods provided to optimize performance of this operation based on the nature of the multi-byte codeset:

__wctomb_iso1() - Since ISO8859-1 is a proper subset of Unicode, its data can be converted from Unicode by simply casting its 16-bit value to an 8-bit value. This is the fastest method for converting Unicode to ISO8859-1.

__wctomb_utf() - UTF-8-based data can be converted from Unicode by using a simple bit-shifting algorithm. This method should be used for all UTF-8-based locales since it is faster than the __wctomb_std() method listed below.

__wctomb_std() - For all other codesets, the __wctomb_std() method converts the multi-byte data from Unicode using the iconv() interface. The conversion descriptor is defined as a static pointer so that multiple calls to __wctomb_std() will not occur. Because this locale method is dependent on libiconv, it has been placed into a common locale method object in /usr/lib/nls/loc/methods/stdmeth.o instead of the libc library to avoid creating an unnecessary dependency between libc and libiconv.

wcswidth() Determine the display width of a wide character string. There is one standard locale method, __wcswidth_std(), that loops through the wide character string calling wcwidth() to determine the display width of each character.

wcwidth() Determine the display width of a wide character. Since all wide characters can be assumed to have the Unicode encoding, each character’s display width can be determined with a single locale method.

__wcwidth_std() - Determine the display width of a character based on its Unicode value. Most characters that can be displayed have a display width of 1; CJK ideographs and Hangul syllables will have a display width of 2, and combining characters have a width of 0. The assumption that the wide character encoding is Unicode allows for a single wcwidth() method that is appropriate for all locales.

Localedef Command Impacts
The `localedef` command has a global method table that is defined internally to the command and is used if the `localedef` command user does not explicitly state the methods desired using the -m flag. These tables will need to

be modified and expanded to reflect the newly-available methods.

Table 58. Internal Locale Methods Called for Each Locale

Locale Method	Unicode Locales	ISO8859-1-Based Locales	Single Byte Non-ISO1-Based Locales	Multi-Byte Locales
mblen()	__mblen_utf()	__mblen_sb()	__mblen_sb()	__mblen_std()
mbstopcs()	__mbstopcs_std()	__mbstopcs_std()	__mbstopcs_std()	__mbstopcs_std()
mbstowcs()	__mbstowcs_std()	__mbstowcs_std()	__mbstowcs_std()	__mbstowcs_std()
mbtopc()	__mbtopc_std()	__mbtopc_std()	__mbtopc_std()	__mbtopc_std()
mbtowc()	__mbtowc_utf()	__mbtowc_isol()	__mbtowc_std()	__mdtowc_std()
pcstombs()	__pcstombs_std()	__pcstombs_std()	__pcstombs_std()	__pcstombs_std()
pctomb()	__pctomb_std()	__pctomb_std()	__pctomb_std()	__pctomb_std()
wcstombs()	__wcstombs_std()	__wcstombs_std()	__wcstombs_std()	__wcstombs_std()
wcswidth()	__wcswidth_std()	__wcswidth_std()	__wcswidth_std()	__wcswidth_std()
wctomb()	__wctomb_utf()	__wctomb_isol()	__wctomb_std()	__wctomb_std()
wcwidth()	__wcwidth_std()	__wcwidth_std()	__wcwidth_std()	__wcwidth_std()

11.3.3.3 Input Methods

The universal input method that is currently used in the UNIVERSAL locale is the basis for input methods in Unicode-based locales. ISO/IEC DIS 14755 basic operation support has been added, which provides the ability to enter a Unicode character by holding down <Ctrl>,<Shift> and entering the appropriate hexadecimal representation. The character set lists align with the script designations in the Unicode 2.0 standard. Input method lists contain simply the xx_XX designation for the input method to be selected. These attributes are configurable through the .imcfg file for the universal locale.

The default input method selected should match that of the corresponding locale. For example, bringing up the JA_JP (Unicode) locale should default to the Japanese input method.

11.3.3.4 Fonts and X11 Locales

AIX currently provides bitmap fonts for UTF-8-based versions of the Baltic locales in the fileset X11.fnt.ucs.com. However, these fonts currently contain only those characters necessary for Baltic support. Similar types and sizes for these fonts already exist for a full complement of characters from

ISO8859-1 to ISO8859-9, IBM-1046 and IBM-850. These existing bdf fonts have been combined to provide a nearly-complete set of single wide bitmap fonts.

In the same way, 19 point and 27 point CJK fonts can be generated to provide a complete set of UCS-based fonts in two sizes that cover the entire CJK Ideograph Range (4E00 - 9FFF) and the Hangul Syllables Range (AC00 - D7FF). These fonts are shipped with the fileset X11.fnt.ucs.cjk.

Dt font aliases have been created to provide the appropriate font sets, so that any Unicode-based locale should operate with a similar set of compatible fonts.

An alternate X11 locale has been made available for use in those installations where CJK font support is not desired and where loading of complete CJK fonts would be a severe hindrance to proper performance.

11.3.3.5 Layout Services

A single layout object, /usr/lib/nls/loc/methods/uni_layout.o, has been provided that will format Unicode characters according to the following sets of rules:

- For Hebrew and Arabic, characters are presented according to the bidirectional behavior rules as presented in the Unicode 2.0 standard, section 3.11.
- For Vietnamese, combining sequences that correspond to characters in the Unicode range U+1EA0 through U+1EF9 have been changed to their precombined forms for presentation purposes.
- Simple combining sequences for characters in the Basic Latin and Latin A extensions shall be honored and altered to the precomposed forms for presentation (that is, characters less than U+01FF).
- Where fonts exist to do so, combining sequences other than those mentioned above are presented as *show hidden* sequences.
- An optional layout engine that formats Korean according to the rules in Section 3.10 of the Unicode Standard.

11.3.4 Installation and Packaging

The following is true regarding installation and packaging:

- Unicode-based locales are selectable for install from SMIT and from the BOS install menus.

- Any locale support that is common to all UTF-8 Unicode based platforms is packaged and shipped in the fileset bos.loc.com.utf.
- Each Unicode based locale is packaged as bos.loc.XX_XX, where XX_XX is the language designation (that is, EN_US, FR_FR, and so on).
- If locale-specific X11 resources are necessary, they are packaged as X11.loc.XX_XX, as appropriate.
- Translated message filesets have been created for all Unicode-based locales that directly correspond with a translatable AIX language. For example, bos.msg.KO_KR was created for Korean messages in the Unicode-based locale.
- See 11.3.5, “List of Supported Unicode Locales” on page 600 for a list of supported Unicode locales.

11.3.5 List of Supported Unicode Locales

Table 59 provides the currently supported Unicode locales:

Table 59. Supported Unicode Locales

Unicode Locale	Language Designation
Albanian	SQ_AL
Arabic	AR_AA
Bulgarian	BG_BG
Catalan	CA_ES
Chinese (Simplified)	ZH_CN
Chinese (Traditional)	ZH_TW
Croatian	HR_HR
Czech	CS_CZ
Danish	DA_DK
Dutch	NL_NL
Dutch (Belgium)	NL_BE
English (Great Britain)	EN_GB
English (United States)	EN_US
Estonian	ET_EE
Finnish	FI_FI

Unicode Locale	Language Designation
French	FR_FR
French (Belgium)	FR_BE
French (Canada)	FR_CA
French (Switzerland)	FR_CH
German	DE_DE
German (Switzerland)	DE_CH
Greek	EL_GR
Hebrew	IW_IL
Hungarian	HU_HU
Icelandic	IS_IS
Italian	IT_IT
Japanese	JA_JP
Korean	KO_KR
Latvian	LV_LV
Lithuanian	LT_LT
Macedonia	MK_MK
Norwegian	NO_NO
Polish	PL_PL
Portuguese	PT_PT
Portuguese (Brazil)	PT_BR
Romanian	RO_RO
Russian	RU_RU
Serbian Cyrillic	SR_SP
Serbian Latin	SH_SP
Slovak	SK_SK
Slovene	SL_SI
Spanish	ES_ES

Unicode Locale	Language Designation
Swedish	SV_SE
Thai	TH_TH
Turkish	TR_TR
Vietnamese	VI_VN

11.4 Java NLS Support

Prior to AIX Version 4.3, Java Version 1.02 was shipped with AIX. In AIX Version 4.3.0, Java Version 1.11 is shipped as part of the base operating system.

The previous version of Java (1.02) had virtually no national language support (Java 1.02 only really has support for the English language). However, Java 1.1 is NLS enabled for the G7 countries: France, the United States, U.K., Germany, Japan, Italy and Canada. Currently, there is no support within Java 1.1 for bidirectional languages. All of the locale support within Java is based on Unicode.

Below is a summary of the Java 1.1 NLS support:

- Locale support
  - Character types based on Unicode 2.0
  - Date, time, number, and currency formatting
  - Collation
- External character sets
  - Font handling
  - Use OS-provided input methods
  - Message handling
  - Printing support
  - Text line-break

11.5 Euro Symbol Support for AIX (4.3.2)

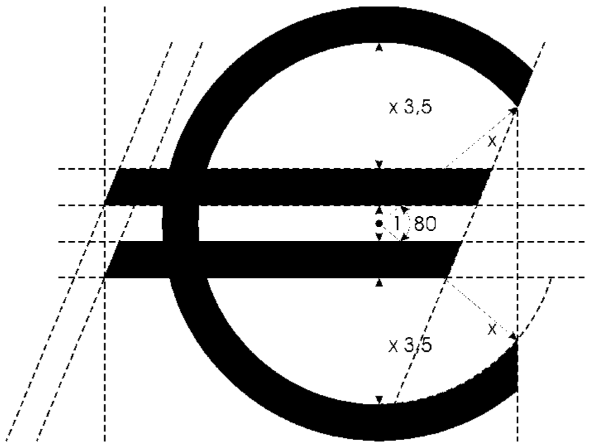


Figure 104. Euro Symbol (http://europa.eu.int/euro/html/entry.html)

This section will provide you with the necessary information required to introduce the Euro symbol as a valid graphical and printable character to your AIX system.

The primary means of code set support for the Euro symbol is achieved by use of the UTF-8 multi-byte encoded locales for each country. A detailed outline of the locale definitions for the UTF-8 code set, the keyboard definitions, the input methods, and the codeset conversion tables is given.

For those customers that have applications that will not support multi-byte encoding, such as UTF-8, a Euro single-byte migration option based on the IBM-1252 code set is provided. This topic is addressed in 11.5.6, “Euro SBCS Migration Option - IBM-1252 Locale” on page 626.

11.5.7, “Packaging” on page 628, and 11.5.8, “Installation of Euro Symbol Support” on page 629 cover the Euro symbol support in more practical terms. You may want to skip over to these sections first and install the Euro symbol related locales by following the documented step-by-step instructions, gain some experience in the new environment, then come back to the more theoretical oriented sections at a later time.

11.5.1 Overview

The Euro is being introduced by the European Monetary Union (EMU) as a common currency to be adopted by all EMU member countries. Initial use of the Euro in banking and industry is planned beginning in January of 1999. During the first three years after introduction, the Euro currency and the existing national currencies will both be used, and a fixed exchange rate will be established. The goal is to completely replace the existing national currencies by the year 2002.

The primary means of achieving code set support for the Euro symbol is to make use of the UTF-8 locales for each country. UTF-8 refers to the X/Open file system safe UCS transformation format (FSS-UTF). It is a multi-byte code set suitable to encode plain text on traditional byte-oriented systems, such as AIX, and is directly related to the universal code character set (UCS). The Unicode standard has adopted code point position U+20AC as <EURO_SIGN>. This is not to be confused with the old European currency symbol located at U+20A0.

The new locales must be able to effectively support the dual currency situation that will exist between the years 1999 - 2002. During this time period, the locale definition will still use the country’s national currency definition as the default, but a mechanism is provided to switch the LC_MONETARY definition so that the Euro currency formatting is used instead of the country’s national currency formatting rules.

For those customers that have applications that will not support multi-byte encodings such as UTF-8, a Euro single-byte migration option is provided as well. This option is based on the Windows 1252 placement of the Euro at 0x80 to provide the best compatibility with Windows 98/NT clients.

11.5.2 Local Definitions for the UTF-8 Code Set

A locale is made up of the language, territory, and code set combination used to identify a set of language conventions. The language specific information is accessed through the locale database that is compiled by the `localedef` command. The `localedef` command takes three different files as input:

- One file describes the local methods to be overridden in respect to the defaults when constructing a locale.
- The second file contains a mapping from the character symbols and collating element symbols to actual character encodings.
- Finally, the language conventions are grouped in six categories to include information about collation, case conversion, and character classification, the language of message catalogs, date-and-time representation, the monetary symbol, and numeric representation.

The local category source definitions are given in the third input file, the local definition source file.

The following categories can be defined for a given local:

LC_COLLATE	Determines character-collation or string-collation information.
LC_CTYPE	Determines character classification, case conversion, and other character attributes.
LC_MESSAGES	Determines the format for affirmative and negative responses.
LC_MONETARY	Determines rules and symbols for formatting monetary numeric information.
LC_NUMERIC	Determines rules and symbols for formatting and non-monetary numeric information.
LC_TIME	Determines a list of rules and symbols for formatting time and date information.

NLS uses the environment variables LC_COLLATE**,** LC_CTYPE**,** LC_MESSAGES**,** LC_MONETARY**,** LC_NUMERIC and LC_TIME to define the current values for their respective categories and to influence the selection of locales. In addition to the previously mentioned medium priority class environment variables, the LANG low priority class environment variable specifies the installation default locale. You can change the LANG variable at any time. Furthermore, the high priority class variable LC_ALL is provided that takes precedence above all the other NLS environment variables mentioned before.

Any UTF-8 based locale installed on an AIX system will provide the desired support for the Euro symbol. For those countries that may have to actively use the Euro (Table 61 on page 608), the locales will deliver the input methods and the keyboard maps required to enter the Euro symbol through the keyboard. For the same group of countries, an additional LC_MONETARY locale is available to enable the Euro currency formatting. This locale is identified by the suffix *@euro.* For example, if you are using the UTF-8 locale DE_DE, which specifies German language and territory, and your system is configured for Euro currency formatting, then the `locale` command will return the following output:

```
# locale
LANG=DE_DE
LC_COLLATE="DE_DE"
LC_CTYPE="DE_DE"
```

```
LC_MONETARY="DE_DE@euro"
LC_NUMERIC="DE_DE"
LC_TIME="DE_DE"
LC_MESSAGES="DE_DE"
LC_ALL=
```

The related locale definition source files and the locale databases can be found in the /usr/lib/nls/loc directory:

```
# cd /usr/lib/nls/loc
# ls -l DE_DE* | egrep  -v "\.[il]" | cut -c55-
DE_DE -> /usr/lib/nls/loc/DE_DE.UTF-8
DE_DE.UTF-8
DE_DE.UTF-8.src
DE_DE.UTF-8@euro
DE_DE.UTF-8@euro.src
DE_DE.UTF-8@euro__64
DE_DE.UTF-8__64
DE_DE@euro -> /usr/lib/nls/loc/DE_DE.UTF-8@euro
DE_DE@euro__64 -> /usr/lib/nls/loc/DE_DE.UTF-8@euro__64
DE_DE__64 -> /usr/lib/nls/loc/DE_DE.UTF-8__64
```

The locale database for the DE_DE locale is actually an alias for DE_DE.UTF-8, and likewise, the database for the DE_DE@euro locale is linked to DE_DE.UTF-8@euro. Note, you should not use the referenced locale databases DE_DE.UTF-8 and DE_DE.UTF-8@euro explicitly as parameter to the `chlang` command or as variable value for LANG. The names of the locales are aligned to the traditional AIX locale naming convention.

Since a locale is a loadable object module, a different object is required when running in the 64-bit environment. Consequently, you find one 64-bit enabled locale database for any 32-bit locale database. The 64-bit databases are readily identified by the *__64* suffix. In the 64-bit environment, the application will automatically append *__64* to the name of the locale when searching for the proper NLS.

The locale definition source files hold the suffix *.src* and belong to the separately installable bos.loc.adt.locale fileset.

11.5.2.1 Euro Sign Character Classification

For each UTF-8 based locale, the Euro sign was added as a valid graphical and printable character. The Euro sign is not classified as an alphabetic, upper or lower case letter, nor a numeric symbol.

For example, if you display the locale definition source file /usr/lib/nls/loc/DE_DE.UTF-8.src for the German UTF-8 locale, you will find one entry for each of the keywords `graph` and `print` in the LC_CTYPE category statement:

```
*************
LC_CTYPE
*************
...
graph /
...
          ;<EURO-CURRENCY_SIGN>/
...
          ;<EURO_SIGN>/
...
print /
...
          ;<EURO-CURRENCY_SIGN>/
...
          ;<EURO_SIGN>/
...
END LC_CTYPE
```

(<EURO-CURRENCY_SIGN> refers to the old European currency symbol.)

The locale definition source files belong to the separately installable bos.loc.adt.locale fileset. Note that the Euro sign is added for all UTF-8 based locales not only for the locales of the countries which are member of the European Monetary Union. Indeed, this is one of the advantages of the Universal Coded Character Set (UCS).

11.5.2.2 Euro Sign Encoding

For each UTF-8 based locale, the UTF-8 character set description source file /usr/lib/nls/charmap/UTF-8 maps the old European currency symbol to U+20A0 and the new Euro sign to U+20AC. Each character symbol definition consists of a symbol name and the character encoding. In our case, the code value is given as a set of three hexadecimal constants:

```
...
# Currency Symbols : U20A0 - U20CF
#

<EURO-CURRENCY_SIGN>                                    \xe2\x82\xa0
...
<EURO_SIGN>                                             \xe2\x82\xac
...
```

Table 60 provides an overview of the different encoding for the <EURO-CURRENCY_SIGN> and the <EURO_SIGN>:

Table 60. Encoding for the European Currency Symbol and Euro Sign

Encoding	<EURO-CURRENCY_SIGN>	<EURO_SIGN>
UCS-2: Hexadecimal Representation	20A0	20AC
UTF-8 multi-byte: Byte Sequence in Hexadecimal Representation	e2 82 a0	e2 82 ac
UTF-8 multi-byte: Byte Sequence in Binary	11100010 10000010 10100000	11100010 10000010 10101100

For more details about character encoding refer to Code Set Overview in *AIX Version 4.3 General Programming Concepts: Writing and Debugging Programs*.

The character set description source files belong to the separately installable fileset bos.loc.adt.locale.

11.5.2.3 LC_MONETARY Formatting Information

For each Western European locale that may actively use the Euro symbol, an additional locale that contains the LC_MONETARY information for the Euro currency is provided. A list of all affected locales is given in Table 61:

Table 61. List of Locales for Euro Specific LC_MONETARY Locale

Language/Territory	Identifier
Catalan	CA_ES
Dutch (Belgium)	NL_BE
Dutch	NL_NL
English (Great Britain)	EN_GB
Finnish	FI_FI
French (Belgium)	FR_BE
French (Switzerland)	FR_CH
French	FR_FR

Language/Territory	**Identifier**
German (Switzerland)	DE_CH
German	DE_DE
Italian	IT_IT
Portuguese	PT_PT
Spanish	ES_ES

To obtain the traditional local currency definition, the standard locale is used as before. Whenever the Euro currency formatting is desired, the LC_MONETARY category should be set by the application (using the `setlocale()` subroutine) or by the user (using the LC_MONETARY environment variable) to "*XX_XX@euro*", where *XX_XX* represents the language and territory designation for any of the locales listed in the previous table.

The LC_MONETARY information for each @euro variant is defined by the related locale definition source file. For example, you will find in the DE_DE@euro locale definition source file /usr/lib/nls/loc/DE_DE.UTF-8@euro.src the following entries for the LC_MONETARY category:

```
*************
LC_MONETARY
*************

int_curr_symbol   "<E><U><R><SPACE>"
currency_symbol   "<EURO_SIGN>"
mon_decimal_point <COMMA>
mon_thousands_sep "<FULL_STOP>"
mon_grouping      3
positive_sign     ""
negative_sign     "<HYPHEN-MINUS>"
int_frac_digits   2
frac_digits       2
p_cs_precedes     0
p_sep_by_space    1
n_cs_precedes     0
n_sep_by_space    1
p_sign_posn       1
n_sign_posn       1

END LC_MONETARY
```

The XPG monetary formatting subroutine *strfmon()* uses the information provided to format monetary quantities according to the settings for the keywords, as provided in Table 62.

Table 62. LC_MONETARY Keywords for the Euro Locale

Keyword	Description
int_curr_symbol	Specifies the string used for the international currency symbol.
currency_symbol	Specifies the string used for the local currency symbol.
mon_decimal_point	Specifies the string used for the decimal delimiter.
mon_thousands_sep	Specifies the character separator used for grouping digits to the left of the decimal delimiter.
mon_grouping	Specifies a string that defines the size of each group of digits.
positive_sign negative_sign	Specifies the string used to indicate a nonnegative / negative-valued formatted monetary quantity.
int_frac_digits frac_digits	Specifies an integer value representing the number of fractional digits (those after the decimal delimiter) to be displayed in a formatted monetary quantity using the int_curr_symbol / currency_symbol value.
p_cs_precedes n_cs_precedes	Specifies an integer value indicating whether the int_curr_symbol or currency_symbol string, precedes (1) or follows (0) the value for a nonnegative / negative formatted monetary quantity
p_sep_by_space n_sep_by_space	Specifies an integer value indicating whether the int_curr_symbol or currency_symbol string is separated (1) or not separated (0) by a space from a nonnegative / negative formatted monetary quantity
p_sign_posn n_sign_posn	Specifies an integer value indicating the positioning of the positive_sign/negative_sign string for a nonnegative/negative formatted monetary quantity. 1 Indicates that the positive_sign/negative string precedes the quantity and the int_curr_symbol or currency_symbol string. 2 Indicates that the positive_sign/negative string follows the quantity and the int_curr_symbol or currency_symbol string. 4 Indicates that the positive_sign string immediately follows the int_curr_symbol or currency_symbol string.

For a detailed description of the LC_MONETARY category keywords refer to LC_MONETARY Category for the Locale Definition Source File Format in *AIX Version 4 Files Reference*. The locale definition source files are part of the separately installable bos.loc.adt.locale fileset.

Generally, the LC_MONETARY information for each @euro variant of the locale contains the same formatting information as the country's historical currency formatting with the following modifications, as necessary:

- The alphabetic international currency symbol as defined by the int_curr_symbol keyword is set to "<E><U><R><space>".
- The string to use for the locale currency symbol as defined by *currency_symbol* keyword is set to <EURO_SIGN> (the Euro symbol).
- The number of decimal places for Euro symbol formatting, as defined by the keywords *int_frac_digits* and *frac_digits* is set to two decimal places in all cases.
- Other formatting conventions, such as decimal point and thousands separator, can be uniquely specified by each country. The relevant keywords where minor deviations occur are *mon_decimal_point, mon_thousands_sep, p_cs_precedes, n_cs_precedes, p_sign_posn, n_sign_posn.* *Table 63* summarizes the differences between the Euro LC_MONETARY locales

Table 63. Locale Specific Deviations in the LC_MONETARY Category

Locale	**mon_decimal_point**	**mon_thousands_sep**	**p_cs_precedes n_cs_precedes**	**p_sign_posn**	**n_sign_posn**
CA_ES@euro	<COMMA>	<FULL_STOP>	1	1	1
DE_DE@euro	<COMMA>	<FULL_STOP>	0	1	1
ES_ES@euro	<COMMA>	<FULL_STOP>	1	1	1
FI_FI@euro	<COMMA>	<SPACE>	0	1	1
FR_BE@euro	<COMMA>	<FULL_STOP>	0	1	1
FR_FR@euro	<COMMA>	<SPACE>	0	1	1
IT_IT@euro	<COMMA>	<FULL_STOP>	1	4	4
NL_BE@euro	<COMMA>	<FULL_STOP>	0	1	1
NL_NL@euro	<COMMA>	<FULL_STOP>	1	1	2
PT_PT@euro	<DOLLAR_SIGN>	<FULL_STOP>	0	1	1

Note: the <FULL_STOP> character is similar in appearance to a period (.).

11.5.2.4 Collating Sequence for Euro Locales

For each UTF-8 based locale, the collation sequence for that locale was modified to contain the Euro symbol. The UTF-8 locales adhere to a multiple pass collation sequence, and the Euro sign will be collated in the fourth pass between the dollar sign and the sterling sign (British Pound). The appropriate entry for this behavior, in the associated locale definition source file, appear as follows:

```
...
LC_COLLATE

...
<DOLLAR_SIGN>       IGNORE;IGNORE;IGNORE;<DOLLAR_SIGN>
<EURO_SIGN>         IGNORE;IGNORE;IGNORE;<EURO_SIGN>
<POUND_SIGN>        IGNORE;IGNORE;IGNORE;<POUND_SIGN>
...

END LC_COLLATE
...
```

If the LC_MONETARY locale is set to activate the Euro symbol formatting, the Euro sign collates in the first pass between the dollar sign and the percent sign for all UTF-8 based locales listed in Table 63 on page 611. Any of the relevant locale definition source files /usr/lib/nls/loc/XX_XX.UTF-8@euro.src will have the following entries:

```
...
**************
LC_COLLATE
**************

oder_start

...
<DOLLAR_SIGN>
<EURO_SIGN>
<PERCENT_SIGN>
...

order_end

END LC_COLLATE
...
```

11.5.3 Keyboard Definitions

IBM follows the recommendation of the European Commission (EC) regarding placement of the Euro symbol on keyboards. This recommendation, along with other information from the EC, can be found at IT impact of the Euro*, (EC Information Society Project Office (ISPO))* `http://www.ispo.cec.be/y2keuro/euroit.htm`. The European Commission’s recommendation is to place the Euro symbol at the position AltGr+e on all European keyboards, except on those keyboard layouts where the key combination AltGr+e is already assigned to produce a different character. In those cases, a combination of AltGr+4 or AltGr+5 will be assigned, depending on the particular keyboard layout. The AltGr (Alt Graphics) key allows you to enter additional characters through the keyboard and is located to the right of the space bar on keyboards with this feature.

Table 64 summarizes the AIX keyboards that will be modified to incorporate the Euro symbol and specifies the placement of the Euro symbol on each European keyboard. As of the writing of this document, these keyboard placements match the current recommendation of the EC regarding placement of the Euro symbol on keyboards.

Table 64. Keyboard Definitions to Incorporate the Euro Symbol

Language/Territory	AIX Keyboard Name	Keyboard ID	Euro Placement
Catalan/Spain	CA_ES	172	AltGr+e
Danish/Denmark	DA_DK	159	AltGr+5
Dutch/Belgium	NL_BE	120	AltGr+e
Dutch/Netherlands	NL_NL	143	AltGr+e
English/UK	EN_GB	166	AltGr+4
English/UK	EN_GB@alt	168	AltGr+e
Finnish/Finland	FI_FI	153	AltGr+5
Finnish/Finland	FI_FI@alt	285	AltGr+5
French/Belgium	FR_BE	120	AltGr+e
French/France	FR_FR	189	AltGr+e
French/France	FR_FR@alt	251	AltGr+e
French/Switzerland	FR_CH	150F	AltGr+e
German/Germany	DE_DE	129	AltGr+e

Language/Territory	AIX Keyboard Name	Keyboard ID	Euro Placement
German/Switzerland	DE_CH	150G	AltGr+e
Icelandic/Iceland	IS_IS	197	AltGr+5
Italian/Italy	IT_IT	142	AltGr+5
Italian/Italy	IT_IT@alt	293	AltGr+5
Norwegian/Norway	NO_NO	155	AltGr+5
Portuguese/Portugal	PT_PT	163	AltGr+5
Spanish/Spain	ES_ES	172	AltGr+e
Swedish/Sweden	SV_SE	153	AltGr+5
Swedish/Sweden	SV_SE@alt	285	AltGr+5

AIX supports two different types of keyboards: low function terminal (LFT) and X server keyboards. Although these two keyboard maps appear to be the same, they are separate and distinct.

Low-function terminals (LFTs) support single-byte codeset languages using key maps. An LFT key map translates a key stroke into a character string in the code set of the given locale. LFT does not support languages that require multi-byte code sets. Hence, you will not have any Euro support on behalf of the lft0 pseudo device driver. However, if you configure your system to use UTF-8 locales to gain Euro symbol support, you will find by the use of the `lskbd` command, that the name of the LFT software keyboard map suggests a multi-byte keyboard map. For example, in a German UTF-8 locale environment, you would get the following response by the `lskbd` command:

```
# lskbd
The current software keyboard map = /usr/lib/nls/loc/DE_DE.lftkeyboard
```

But a closer examination of the relevant keyboard map files reveals that they are symbolic links to the regular C locale LFT keyboard map: `/usr/lib/nls/loc/C.lftkeymap`.

For a full graphical Euro symbol enablement, you have to be in an X environment. The associated X server has an attached keyboard, and the server uses mapping tables to manage the mapping of keyboard events. The mapping of an X server keyboard can be changed by using the `xmodmap` command. This command converts the keyboard so that it returns the key symbol supported by the system.

At startup of the X server, a query to the ODM returns the locale that determined the keyboard map for the LFT pseudo device driver; that is, the `swkb_path` attribute of lft0 is examined for the currently used locale for the keyboard map. In the next step, the `xmodmap` command defines the proper keyboard mapping to the server according to the found locale.

When characters are typed in on the keyboard reach the server, the characters are in the form of key codes. These key codes are converted into keysyms (key symbols), or if applicable, into a pair of keysym/modifier as defined by the table provided in the client.(A modifier indicates the state of the keyboard as determined by the modifier keys: Shift, Lock, Ctrl, Alt and Alt Graphic. The keyboard table was defined to the sever by the `xmodmap` command and contains mappings for each of the keycodes into a predefined set of codes called keysyms or keysym/modifier.

The file containing the xmodmap command expressions to be run by the `xmodmap` command is in the directory /usr/lpp/X11/defaults/xmodmap/XX_XX, where *XX_XX* represents the language/territory designation for the UTF-8 locale. The following example shows the Euro specific entry for the German UTF-8 locale in /usr/lpp/X11/defaults/xmodmap/DE_DE/keyboard:

```
...
! Row 2        Base           Shift          Alt-Gr (Mod2)
! -----        ----           -----          -------------
!
!keycode 24 =  Tab            NoSymbol       NoSymbol
 keycode 25 =  q              Q              at
 keycode 26 =  w              W              NoSymbol
 keycode 27 =  e              E              EuroSign
 keycode 28 =  r              R              NoSymbol
...
```

If you press the AltGr+e key combination on a German keyboard, the first step of the input processing is completed when the keycode to keysym/modifier conversion is done. In this specific case, the modifier is masked. For example, the AltGr modifier and the character "e" are combined to map to the EuroSign key symbol. But the keysym still must be mapped to the related character string in the code set specified by your locale before an application can process the input. This conversion is governed by the input method.

11.5.4 Input Methods for the Euro Symbol

For an application to run in the international environment, for which National Language Support provides a base, input methods are needed. An input method is a set of functions that translates key strokes, or more precisely, key symbol/modifier pairs into character strings in the code set specified by your locale. Each type of input method has the following features:

Keymaps	Set of input method keymaps (imkeymaps) that work with the input method and determine the supported locales.
Keysyms	Set of key symbols (keysyms) that the input method can handle.
Modifiers	Set of modifiers, or states, each having a mask value, that the input method supports.

Your locale determines which input method should be loaded, how the input method runs, and which devices are used. The /usr/lib/nls/loc directory contains the input methods installed on your system. Input method file names have the format *XX_XX.im* where *XX_XX* represents the language and territory designation for any of the locales. For a given input method, you may find an associated configuration file identified by the suffix *.imcfg* .The input method provides support for user-defined imkeymaps, allowing you to customize input method mapping. The input methods support imkeymaps for each locale. The file name for imkeymaps is similar to that of input methods, except that the suffix for imkeymap files is *.imkeymap* instead of *.im*. The imkeymaps are generated by the keycomp command. The keycomp command compiles a keyboard mapping file into an input method keymap file. The locale specific keyboard mapping files are recognized by the suffix *.imkeymap.src* .

For a system set up for German UTF-8 Euro symbol support (DE_DE), you would find all the files mentioned above in the /usr/lib/nls/loc directory:

```
# ch /usr/lib/nls/loc
# ls -l DE_DE*im* | cut -c55-
DE_DE.UTF-8.im -> /usr/lib/nls/loc/sbcs.im
DE_DE.UTF-8.imcfg -> /usr/lib/nls/loc/UNIVERSAL.imcfg
DE_DE.UTF-8.imkeymap
DE_DE.UTF-8.imkeymap.src
DE_DE.im -> /usr/lib/nls/loc/UNIVERSAL.im
DE_DE.imcfg -> /usr/lib/nls/loc/UNIVERSAL.imcfg
DE_DE.imkeymap -> /usr/lib/nls/loc/DE_DE.UTF-8.imkeymap
```

Note that actually two input methods are present, and indeed, both are needed for your UTF-8 Euro environment. DE_DE.im is linked to the standard UNIVERSAL input method and DE_DE.UTF-8.im is an alias for the traditional single-byte input method commonly used in conjunction with the ISO8859 and the IBM-850 PC code sets. Both input methods are related by the use of the same UNIVERSAL input method configuration file UNIVERSAL.imcfg.

11.5.4.1 Single-Byte Character Set Input Method

The Single-Byte Character Set Input Method (SIM) is the standard that supports most of the locales. It is a mapping function that supports simple composition defined on workstation keyboards associated with single-byte locales.

SIM supports any keyboard, code set, and language that the `keycomp` command can describe. You can customize SIM using imkeymaps. The coded strings returned by the input method depend on the imkeymap. Due to this feature, you can extend the capability of the SIM to support multi-byte UTF-8 locales. Adding just one new key combination (AltGr+e, AltGr+4 or AltGr+5) to make the input of the Euro sign from the keyboard feasible does not require an entire input method to be rewritten.

Similar to the German DE_DE alias for the DE_DE.UTF-8 locale contains the following entry in the keycomp source file /usr/lib/nls/loc/DE_DE.UTF-8.imkeymap.src:

```
...
XK_EuroSign                                         \
                    "\xe2\x82\xac"                  \
                    XK_EuroSign                     \
                    U                               \
                    U                               \
                    U                               \
                    U                               \
                    U
...
XK_e                                                \
                    ’e’                             \
                    XK_E                            \
                    XK_E                            \
                    XK_e                            \
                    ’\x05’                          \
                    U                               \
                    XK_EuroSign
...
```

The XK_e maps to the XK_EuroSign keysym when the modifier key AltGr is used while entering the character e on the keyboard. The XK_EuroSign keysym, in turn, is mapped to the code point \xe2\x82\xac.

11.5.4.2 UNIVERSAL Input Method

If you configured your system to use a UTF-8 locale for Euro symbol support, you have access to the entire range of UTF-8 encoded characters. In order to allow for character input from the keyboard for several thousand different characters, the single-byte input method is not sufficient, and consequently, a UNIVERSAL input method is provided. Note, if your system is configured for a UTF-8 locale that does not support the character input of the Euro sign through a key combination, the UNIVERSAL input method must be used.

The UNIVERSAL input method supports two general modes of character input:

- Switching between installed AIX locale input methods
- Selection of characters from character lists

The UNIVERSAL input method is configured by the default UNIVERSAL.imcfg file in the /usr/lib/nls directory. This file defines the set of available AIX locale input methods and the lists of characters for list-based input. It also defines the key combinations used to invoke the input method selection menus.

The default key combinations recognized by the UNIVERSAL input method are:

Ctrl+Alt+i	Invokes the input method selection menu
Ctrl+Alt+l	Invokes the menu of character lists
Ctrl+Alt+c	Invokes the current list of characters

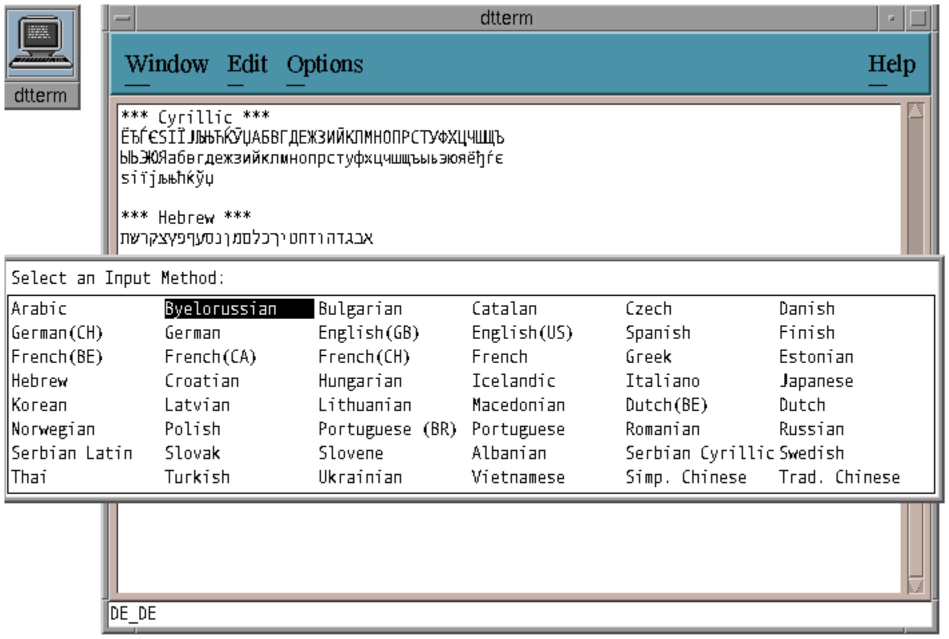
A selection menu can be closed by pressing the Enter key.

After an input method is selected from the input method selection menu (see Figure 105 on page 619), it can be used in the same way as in its associated locale. The label at the left edge of the input method status bar describes the locale for the currently selected input method. For example, if the Simplified Chinese input method is selected from the selection menu, the CN_ZH label is displayed in the status area. Any status information specific to the current input method is displayed immediately to the right of this label. Of course, if you want to switch the input method, the related locale must be installed on the system.

Most input methods require a locale-specific keyboard mapping to support certain input sequences. In the X environment, the keyboard can be remapped by invoking the `xmodmap` command with the keyboard files under the /usr/lpp/X11/defaults/xmodmap directory. For example, the following command remaps the keyboard so as to support the Chinese input method:

```
# xmodmap /usr/lpp/X11/defaults/xmodmap/CN_ZH/keyboard
```

Note that the keyboard mappings assume an associated physical keyboard.



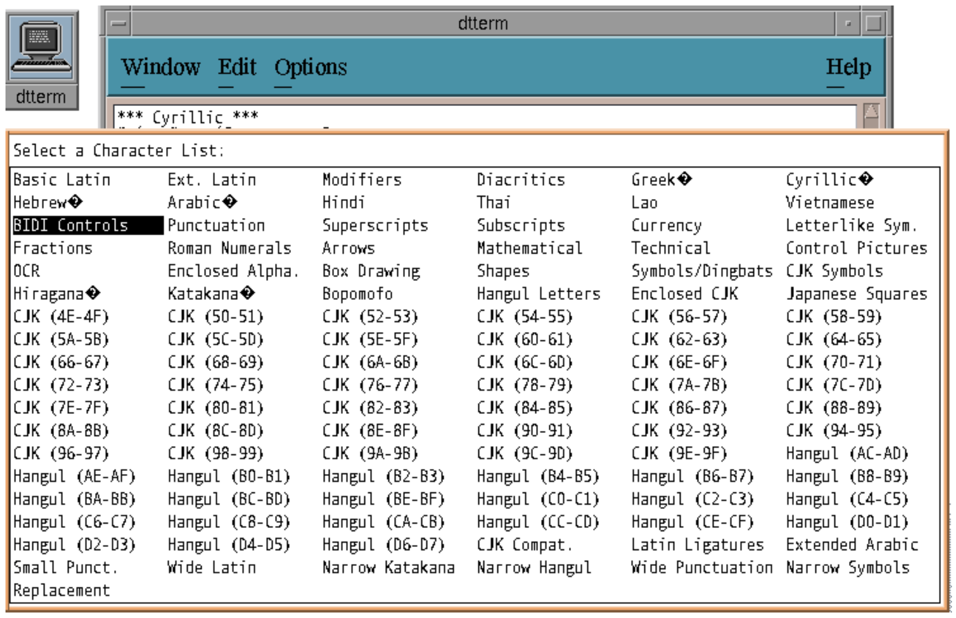
```
*** Cyrillic ***
ЁЂЃЄЅІЇЈЉЊЋЌЎЏАБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪ
ЫЬЭЮЯабвгдежзийклмнопрстуфхцчшщъыьэюяёђѓє
ѕіїјљњћќўџ

*** Hebrew ***
אבגדהוזחטיךכלםמןנסעףפץצקרשת
```

```
Select an Input Method:
Arabic        Byelorussian  Bulgarian     Catalan       Czech         Danish
German(CH)    German        English(GB)   English(US)   Spanish       Finish
French(BE)    French(CA)    French(CH)    French        Greek         Estonian
Hebrew        Croatian      Hungarian     Icelandic     Italiano      Japanese
Korean        Latvian       Lithuanian    Macedonian    Dutch(BE)     Dutch
Norwegian     Polish        Portuguese (BR) Portuguese  Romanian      Russian
Serbian Latin Slovak        Slovene       Albanian      Serbian Cyrillic Swedish
Thai          Turkish       Ukrainian     Vietnamese    Simp. Chinese Trad. Chinese
```

```
DE_DE
```

Figure 105. UNIVERSAL Input Method: Switching

By default, the Ctrl+Alt+l key combination invokes the menu of character lists (Figure 106 on page 620). After a list is selected from this menu, characters can be input by selecting them from the list with the mouse. The characters will initially be inserted in the input method pre-edit area. The characters can be *committed* by pressing the Enter key.



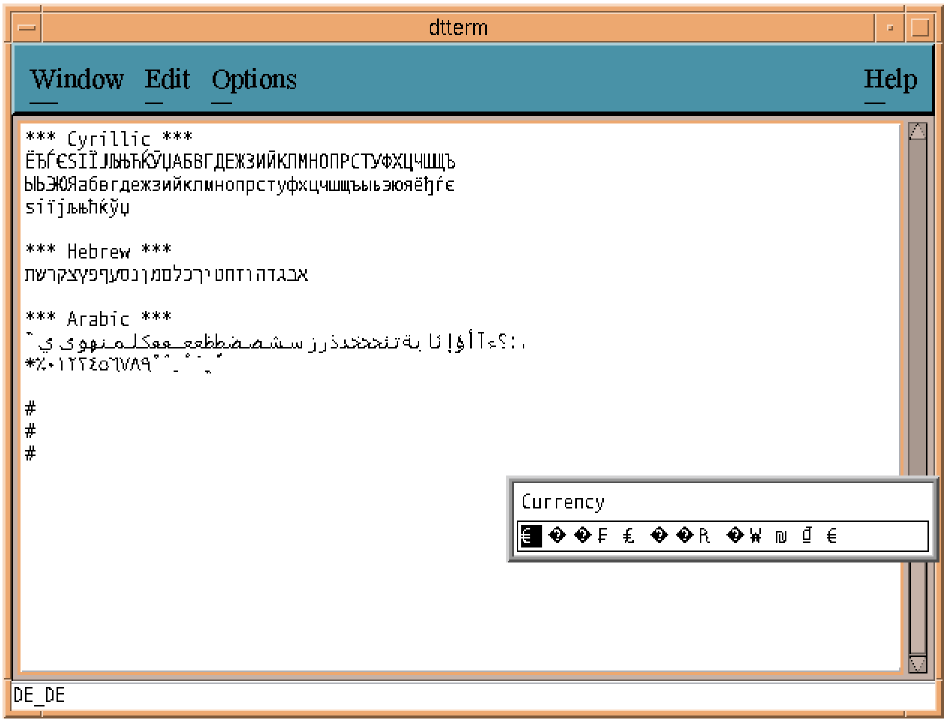
Select a Character List:

Basic Latin	Ext. Latin	Modifiers	Diacritics	Greek�	Cyrillic�
Hebrew�	Arabic�	Hindi	Thai	Lao	Vietnamese
BIDI Controls	Punctuation	Superscripts	Subscripts	Currency	Letterlike Sym.
Fractions	Roman Numerals	Arrows	Mathematical	Technical	Control Pictures
OCR	Enclosed Alpha.	Box Drawing	Shapes	Symbols/Dingbats	CJK Symbols
Hiragana�	Katakana�	Bopomofo	Hangul Letters	Enclosed CJK	Japanese Squares
CJK (4E-4F)	CJK (50-51)	CJK (52-53)	CJK (54-55)	CJK (56-57)	CJK (58-59)
CJK (5A-5B)	CJK (5C-5D)	CJK (5E-5F)	CJK (60-61)	CJK (62-63)	CJK (64-65)
CJK (66-67)	CJK (68-69)	CJK (6A-6B)	CJK (6C-6D)	CJK (6E-6F)	CJK (70-71)
CJK (72-73)	CJK (74-75)	CJK (76-77)	CJK (78-79)	CJK (7A-7B)	CJK (7C-7D)
CJK (7E-7F)	CJK (80-81)	CJK (82-83)	CJK (84-85)	CJK (86-87)	CJK (88-89)
CJK (8A-8B)	CJK (8C-8D)	CJK (8E-8F)	CJK (90-91)	CJK (92-93)	CJK (94-95)
CJK (96-97)	CJK (98-99)	CJK (9A-9B)	CJK (9C-9D)	CJK (9E-9F)	Hangul (AC-AD)
Hangul (AE-AF)	Hangul (B0-B1)	Hangul (B2-B3)	Hangul (B4-B5)	Hangul (B6-B7)	Hangul (B8-B9)
Hangul (BA-BB)	Hangul (BC-BD)	Hangul (BE-BF)	Hangul (C0-C1)	Hangul (C2-C3)	Hangul (C4-C5)
Hangul (C6-C7)	Hangul (C8-C9)	Hangul (CA-CB)	Hangul (CC-CD)	Hangul (CE-CF)	Hangul (D0-D1)
Hangul (D2-D3)	Hangul (D4-D5)	Hangul (D6-D7)	CJK Compat.	Latin Ligatures	Extended Arabic
Small Punct.	Wide Latin	Narrow Katakana	Narrow Hangul	Wide Punctuation	Narrow Symbols
Replacement					

Figure 106. UNIVERSAL Input Method: Character List Selection

When a character list is selected from the character list menu, it becomes the current character list. It can then be invoked by simply entering the default Ctrl+Alt+c key combination (Figure 107 on page 621).

To enter the Euro sign, select the Currency character list and select the Euro sign from the list.



```
*** Cyrillic ***
ЁЂЃЄЅІЇЈЉЊЋЌЎЏАБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪ
ЫЬЭЮЯабвгдежзийклмнопрстуфхцчшщъыьэюяёђѓє
ѕіїјљњћќўџ

*** Hebrew ***
אבגדהוזחטיךכלםמןנסעףפץצקרשת

*** Arabic ***
،؛؟ءآأؤإئابةتثجحخدذرزسشصضطظعغـفقكلمنهوىي ً
*٪٠١٢٣٤٥٦٧٨٩ ٌ ٍ َ ُ ِ ّ ْ

#
#
#
```

```
Currency
€ � � ₣ ₤ � � ₨ � ₩ ₪ ₫ €
```

Figure 107. UNIVERSAL Input Method: Character List

11.5.5 Codeset Conversion Tables

As more code sets are supported, it becomes important not to clutter programs with the dependency of any particular code set. This is known as code set independence. To aid in code set independence, NLS supplies converters that translate character encoding values found in different code sets. Using these converters, a system can accurately process data generated in different code set environments

In addition to Unicode, a number of new 8-bit codesets have been proposed that will contain the Euro symbol. Although AIX will not have supported locales in many of these code sets, it will provide data conversion capabilities to, and from, these code sets. The new code sets that are being introduced are as follows:

- IBM-1252 (extended) - This code set will be used by the Euro single-byte (SBCS) migration option.

- IBM-858 - This code set is identical to IBM-850 except that the dotless letter i, which is used only in Turkish, is replaced with the Euro symbol. This code set will be used in OS/2.
- IBM-114x - Ten new code sets are being introduced that serve the same function as the existing EBCDIC code sets, except that one character is replaced with the Euro. The new code sets are summarized in Table 65.

Table 65. Existing EBCIDIC Code Sets

Existing EBCDIC Code Set	New Euro Code Set	Countries
IBM-037	IBM-1140	US
IBM-273	IBM-1141	Germany
IBM-277	IBM-1142	Denmark, Norway
IBM-278	IBM-1143	Finland, Sweden
IBM-280	IBM-1144	Italy
IBM-284	IBM-1145	Spain
IBM-285	IBM-1146	Great Britain
IBM-297	IBM-1147	France
IBM-500	IBM-1148	Belgium, Canada, Switzerland
IBM-871	IBM-1149	Iceland

The new iconv converters use either the UCSTBL or the Universal_UCS_Conv conversion method.

The UCSTBL method is located in the /usr/lib/nls/loc/uconv directory and loads UCS-2 (Unicode) conversion tables created by the `uconvdef` command. In order to compile the UCS-2 (Unicode) conversion table, the `uconvdef` command reads a source file that defines a mapping between the UCS-2 and a particular multi-byte code set. The source files are in the /usr/lib/nls/uconv table directory with names that are composed of the code set name appended by the suffix .ucmap. The UCSTBL method uses the table to support UCS-2 conversions in both directions. The setup of a converter is complete if the proper symbolic links for conversions in each direction are created in the /usr/lib/nls/iconv directory. For example, the links for the conversion of the IBM-850 code set to UCS-2, and vice versa, are defined as shown below:

```
# cd /usr/lib/nls/loc/iconv
# ln -s /usr/lib/nls/loc/uconv/UCSTBL IBM-850_UCS-2
# ln -s /usr/lib/nls/loc/uconv/UCSTBL UCS-2_IBM-850
```

The Universal_UCS_Conv method is located in the /usr/lib/nls/loc/iconv directory and can be used to convert between any two code sets whose conversions to, and from, UCS-2 are defined. The conversion is instantiated by setting the proper links. Assuming that someone wants to define the conversion of the IBM-850 to, and from, the UTF-8 code set, you would have to enter the following sequence of commands:

```
# cd /usr/lib/nls/loc/iconv
# ln -s /usr/lib/nls/loc/uconv/Universal_UCS_Conv IBM-850_UTF-8
# ln -s /usr/lib/nls/loc/uconv/UCSTBL IBM-850_UCS-2
# ln-s /usr/lib/nls/loc/uconv/UCSTBL UCS-2_UTF-8
# ln -s /usr/lib/nls/loc/uconv/UCSTBL IBM-850
# ln -s /usr/lib/nls/loc/uconv/UCSTBL UTF-8
```

As you can see from the example above, UTF-8 converters are usually done by using the Universal_UCS_Conv and the /usr/lib/nls/loc/uconv/UTF-8 conversion.

It should be noted that, for some conversion pairs, a substitution character may be inserted when converting a Euro symbol from a codeset that contains the Euro to a codeset that does not (For example: Converting IBM-114x to ISO8859-1).

Table 66 summarizes the new iconv converters that are provided for Euro support.

Table 66. Converters for Euro Symbol Support

Converter	Type	Fileset
IBM-1140_IBM-858	Universal	bos.iconv.com
IBM-1140_ISO8859-1	Universal	bos.iconv.com
IBM-1140_UCS-2	UCSTBL	bos.iconv.ucs.ebcdic
IBM-1140_UTF-8	Universal	bos.iconv.ucs.ebcdic
IBM-1141_IBM-858	Universal	bos.iconv.de_DE
IBM-1141_ISO8859-1	Universal	bos.iconv.de_DE
IBM-1141_UCS-2	UCSTBL	bos.iconv.ucs.ebcdic
IBM-1142_IBM-858	Universal	bos.iconv.da_DK
IBM-1142_ISO8859-1	Universal	bos.iconv.da_DK
IBM-1142_UCS-2	UCSTBL	bos.iconv.ucs.ebcdic

Converter	Type	Fileset
IBM-1142_UTF-8	Universal	bos.iconv.ucs.ebcdic
IBM-1143_IBM-858	Universal	bos.iconv.com
IBM-1143_ISO8859-1	Universal	bos.iconv.com
IBM-1143_UCS-2	UCSTBL	bos.iconv.ucs.ebcdic
IBM-1143_UTF-8	Universal	bos.iconv.ucs.ebcdic
IBM-1144_IBM-858	Universal	bos.iconv.it_IT
IBM-1144_ISO8859-1	Universal	bos.iconv.it_IT
IBM-1144_UCS-2	UCSTBL	bos.iconv.ucs.ebcdic
IBM-1144_UTF-8	Universal	bos.iconv.ucs.ebcdic
IBM-1145_IBM-858	Universal	bos.iconv.es_ES
IBM-1145_ISO8859-1	Universal	bos.iconv.es_ES
IBM-1145_UCS-2	UCSTBL	bos.iconv.ucs.ebcdic
IBM-1145_UTF-8	Universal	bos.iconv.ucs.ebcdic
IBM-1146_IBM-858	Universal	bos.iconv.en_GB
IBM-1146_ISO8859-1	Universal	bos.iconv.en_GB
IBM-1146_UCS-2	UCSTBL	bos.iconv.ucs.ebcdic
IBM-1146_UTF-8	Universal	bos.iconv.ucs.ebcdic
IBM-1147_IBM-858	Universal	bos.iconv.fr_FR
IBM-1147_ISO8859-1	Universal	bos.iconv.fr_FR
IBM-1147_UCS-2	UCSTBL	bos.iconv.ucs.ebcdic
IBM-1147_UTF-8	Universal	bos.iconv.ucs.ebcdic
IBM-1148_IBM-858	Universal	bos.iconv.com
IBM-1148_UCS-2	UCSTBL	bos.iconv.ucs.ebcdic
IBM-1148_UTF-8	Universal	bos.iconv.ucs.ebcdic
IBM-1149_IBM-858	Universal	bos.iconv.is_IS
IBM-1149_ISO8859-1	Universal	bos.iconv.is_IS
IBM-1149_UCS-2	UCSTBL	bos.iconv.ucs.ebcdic

Converter	Type	Fileset
IBM-1149_UTF-8	Universal	bos.iconv.ucs.ebcdic
IBM-850_IBM-858	Universal	bos.iconv.ucs.com
IBM-858_IBM-1140	Universal	bos.iconv.com
IBM-858_IBM-1141	Universal	bos.iconv.de_DE
IBM-858_IBM-1142	Universal	bos.iconv.da_DK
IBM-858_IBM-1143	Universal	bos.iconv.com
IBM-858_IBM-1144	Universal	bos.iconv.it_IT
IBM-858_IBM-1145	Universal	bos.iconv.es_ES
IBM-858_IBM-1146	Universal	bos.iconv.en_GB
IBM-858_IBM-1147	Universal	bos.iconv.fr_FR
IBM-858_IBM-1148	Universal	bos.iconv.com
IBM-858_IBM-1149	Universal	bos.iconv.is_IS
IBM-858_IBM-850	Universal	bos.iconv.ucs.com
IBM-858_ISO8859-1	Universal	bos.iconv.ucs.com
IBM-858_UCS-2	UCSTBL	bos.iconv.ucs.com
IBM-858_UTF-8	Universal	bos.iconv.ucs.com
IBM_1141_UTF-8	Universal	bos.iconv.ucs.ebcdic
IBM_1148_ISO8859-1	Universal	bos.iconv.com
ISO8859-1_IBM-1140	Universal	bos.iconv.com
ISO8859-1_IBM-1141	Universal	bos.iconv.de_DE
ISO8859-1_IBM-1142	Universal	bos.iconv.da_DK
ISO8859-1_IBM-1143	Universal	bos.iconv.com
ISO8859-1_IBM-1145	Universal	bos.iconv.es_ES
ISO8859-1_IBM-1146	Universal	bos.iconv.en_GB
ISO8859-1_IBM-1147	Universal	bos.iconv.fr_FR
ISO8859-1_IBM-1148	Universal	bos.iconv.com
ISO8859-1_IBM-1149	Universal	bos.iconv.is_IS

Converter	Type	Fileset
ISO8859-1_IBM-858	Universal	bos.iconv.ucs.com
UCS-2_IBM-1140	UCSTBL	bos.iconv.ucs.ebcdic
UCS-2_IBM-1141	UCSTBL	bos.iconv.ucs.ebcdic
UCS-2_IBM-1142	UCSTBL	bos.iconv.ucs.ebcdic
UCS-2_IBM-1143	UCSTBL	bos.iconv.ucs.ebcdic
UCS-2_IBM-1144	UCSTBL	bos.iconv.ucs.ebcdic
UCS-2_IBM-1145	UCSTBL	bos.iconv.ucs.ebcdic
UCS-2_IBM-1146	UCSTBL	bos.iconv.ucs.ebcdic
UCS-2_IBM-1147	UCSTBL	bos.iconv.ucs.ebcdic
UCS-2_IBM-1148	UCSTBL	bos.iconv.ucs.ebcdic
UCS-2_IBM-1149	UCSTBL	bos.iconv.ucs.ebcdic
UCS-2_IBM-858	UCSTBL	bos.iconv.ucs.com
UTF-8_IBM-1140	Universal	bos.iconv.ucs.ebcdic
UTF-8_IBM-1142	Universal	bos.iconv.ucs.ebcdic
UTF-8_IBM-1143	Universal	bos.iconv.ucs.ebcdic
UTF-8_IBM-1144	Universal	bos.iconv.ucs.ebcdic
UTF-8_IBM-1145	Universal	bos.iconv.ucs.ebcdic
UTF-8_IBM-1146	Universal	bos.iconv.ucs.ebcdic
UTF-8_IBM-1147	Universal	bos.iconv.ucs.ebcdic
UTF-8_IBM-1148	Universal	bos.iconv.ucs.ebcdic
UTF-8_IBM-1149	Universal	bos.iconv.ucs.ebcdic
UTF-8_IBM_1141	Universal	bos.iconv.ucs.ebcdic
UTF-8_IBM_858	Universal	bos.iconv.ucs.com

11.5.6 Euro SBCS Migration Option - IBM-1252 Locale

Use of a Unicode (UTF-8) locale provides the most standardized and widely accepted way of achieving Euro support on AIX. However, not all applications or customers, are ready to migrate to the Unicode solution at this time. The Euro Single-Byte Code Set (SBCS) migration option provides a temporary solution for those customers who need Euro support immediately but are not yet ready to migrate to Unicode.

The migration option consists of an additional set of locales bound to the IBM-1252 code set (same as Windows 1252 extended) for each language/territory listed in Table 61 on page 608. ISO8859-1 is a proper subset of IBM-1252, and as such, all data currently encoded in ISO8859-1 can be used in the IBM-1252 environment without any requirement for data conversion. The IBM-1252 code set differs in relation to the industry standard code set ISO8859-1 to the extend that additional graphic characters are added in the ISO control characters range from 0x80 through 0x9F. The Euro character is defined at code point 0x80 in the IBM-1252 environment, which is identical to the Windows NT and Windows 98 value. This will provide some level of compatibility for Windows clients running AIX server applications.

While all code page 1252 graphics will be processed correctly, the AIX font set will only be extended to support the Euro symbol not the additional unique graphics in code page 1252. Customers selecting this approach are cautioned to review display requirements for 1252 graphics.

Furthermore, assigning characters to the ASCII control points in the code page 1252 can be a significant problem for customers who use ASCII terminals or have applications that use ASCII terminal emulators. Trying to use ASCII terminals with the Euro symbol will require additional customization such as fonts or INOUT transforms, or will be limited to the existing non-euro character sets. AIX provides terminal definition tables that may have to be modified depending on the usage. Since AIX locales can be changed based on an application basis, the only applications to be reviewed for potential customization are those applications that require ASCII terminal input or ASCII terminal emulation support and must handle the Euro at the same time.

Usage of the SBCS migration option will require the use of the fully qualified locale and code set name, as follows:

For example, in order to use the German locale with the Euro SBCS migration option, the user would set LANG=de_DE.IBM-1252. This would provide the proper code set environment for the Euro character but would still format monetary quantities in the traditional national currency format. In order to specify Euro currency formatting, the user would additionally set LC_MONETARY=de_DE.IBM-1252@euro.

Keyboard mappings, including the Euro and bound to IBM-1252, are provided as links to the related ISO keyboard maps.

Although SBCS migration option locales are bound to IBM-1252, full font support for the additional group of characters in IBM-1252 that is not currently defined in ISO8859-1 (other than the Euro) is not guaranteed, nor is it guaranteed that all existing ISO8859-1 fonts will be extended to include the Euro or additional characters. At a minimum, the set of fonts corresponding to the Common Desktop Environment (CDE) dt aliases for IBM-1252 (such as "-dt-interface system-*" and "-dt-interface user-*") are enhanced to include the Euro.

Documentation is provided that will instruct customers who are using IBM-850 based locales how they can achieve Euro support on AIX by creating their own customized locale using the localedef facility. Refer to AIX Version 4.3 Base Documentation and AIX Version 4.3 Extended Documentation.

11.5.7 Packaging

Each Unicode based locale listed in Table 61 on page 608, and commonly referred to by XX_XX in this paragraph, is separately installable and contains the complete set of function to provide both traditional monetary as well as Euro monetary formatting. All locale specific modules are separately installable and are grouped and distributed in two different entities:

- bos.loc.utf.XX_XX fileset
- X11.loc.XX_XX package comprised of the filesets:
    - X11.loc.XX_XX.Dt.rte
    - X11.loc.XX_XX.base.lib
    - X11.loc.XX_XX.base.rte

The scope of the files in the bos.loc.utf.XX_XX fileset is limited to provide the locale support for the Base Operating System (BOS) and the X11.loc.XX_XX package will add the locale support for the X environment. Several filesets will be automatically installed if the installation process cannot find them on the system:

- bos.loc.com.utf (coreq. to bos.loc.utf.XX_XX)
- X11.fnt.ucs.ttf (coreq. to X11.loc.base.rte)

For a complete list of dependencies, examine the output of the suitable `lslpp` command after you installed the filesets on your system. For example, let your UTF-8 locale identifier be DE_DE, so you will get for the bos.loc.utf.DE_DE fileset the following requisites:

```
# lslpp -p "bos.loc.utf.DE_DE"
```

```
  Fileset               Requisites

------------------------------------------------------------------------
--
Path: /usr/lib/objrepos
  bos.loc.utf.DE_DE 4.3.2.0
                        *coreq bos.loc.com.utf 4.3.2.0

For the X11.loc.DE_DE package, the lslpp -p command will yield:
# lslpp -p "X11.loc.DE_DE*"
  Fileset               Requisites

------------------------------------------------------------------------
--
Path: /usr/lib/objrepos
  X11.loc.DE_DE.Dt.rte 4.3.2.0
                        *instreq X11.Dt.rte 4.3.2.0
  X11.loc.DE_DE.base.lib 4.3.2.0
                        *instreq X11.base.common 4.3.2.0
  X11.loc.DE_DE.base.rte 4.3.2.0
                        *coreq X11.fnt.ucs.ttf 4.3.2.0
                        *instreq X11.base.rte 4.3.2.0

Path: /etc/objrepos
  X11.loc.DE_DE.Dt.rte 4.3.2.0
                        *instreq X11.Dt.rte 4.3.2.0
```

Locale definitions for the Euro single-byte migration option is packaged in the same filesets as their non-Euro Latin-1 counterparts; that is, bos.loc.iso.xx_XX where xx_XX represents the language and territory designation. The bos.iconv.xx_XX filesets are requisites for the bos.loc.iso.xx_XX filesets.

11.5.8 Installation of Euro Symbol Support

This section is intended to give a step-by-step instruction for the configuration changes a system administrator has to accomplish in order to make an AIX system ready to support the Euro symbol. The UTF-8 and the SBCS environment for the German language and territory designation DE_DE, or de_DE.IBM-1252, are covered respectively. All the following statements apply to any of the language/territory combinations that are listed in Table 61 on page 608, you just replace DE_DE or de_DE.IBM-1252 by the locale identifier of your choice.

An assumption is that the AIX system was installed using the following default settings for the new and complete override installation for the primary language environment (RS/6000 Hardware with German keyboard attached):

```
...
2 Primary Language Environment Settings (AFTER Install):
     Cultural Convention ................ English (United States)
     Language ........................... English (United States)
     Keyboard ........................... German
     Keyboard Type ...................... Default
...
```

Note, the UTF-8 Unicode locales are not available as optional choices for the primary language environment settings at the time of installation.

After installation the output of the `locale` command would be:

```
# locale
LANG=en_US
LC_COLLATE="en_US"
LC_CTYPE="en_US"
LC_MONETARY="en_US"
LC_NUMERIC="en_US"
LC_TIME="en_US"
LC_MESSAGES="en_US"
LC_ALL=
```

The software keyboard for the lft0 pseudo device driver would be:

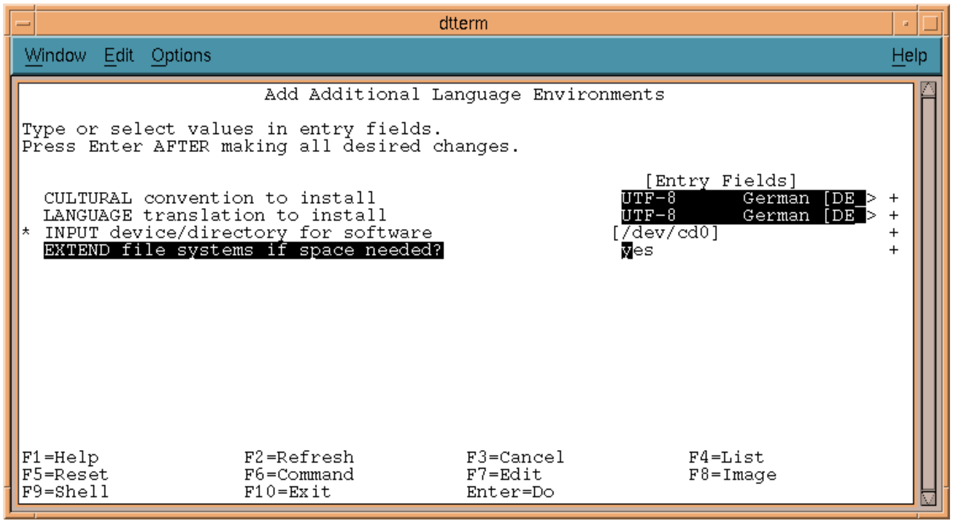
```
# lskbd
The current software keyboard map = /usr/lib/nls/loc/de_DE.lftkeymap
```

The X server, and hence the Common Desktop Environment, will use the /usr/lpp/X11/defaults/xmodmap/de_DE/keyboard file to configure the German keyboard map at startup.

There are two different ways to gain UTF-8 Euro symbol support. Either you decide to have your system default environment configured to be UTF-8 or you prefer to offer the locale just as an additional language environment on the system.

11.5.8.1 Euro UTF-8: Additional Language Environment

In order to activate Euro symbol support using the German UTF-8 locale as an additional language environment, insert one of the AIX BOS CDs in the CD-ROM drive, issue the `smitty mlang` command, select the **Add Additional Language Environments** item from the menu, and press Enter. In the Add Additional Language Environment menu, you enter, by the aide of the F4 function key, the `UTF-8 German[DE_DE]` identifier in the appropriate entry fields as shown in Figure 108 on page 631.



```
                    Add Additional Language Environments

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

                                                   [Entry Fields]
  CULTURAL convention to install                   UTF-8      German [DE > +
  LANGUAGE translation to install                  UTF-8      German [DE > +
* INPUT device/directory for software             [/dev/cd0]               +
  EXTEND file systems if space needed?             yes                      +

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

Figure 108. German UTF-8: Add Additional Language Environment

SMIT will install all necessary filesets for the locale support and you will receive the following output after SMIT has completed its task:

```
...

Installation Summary
--------------------
Name                            Level           Part        Event       Result
-------------------------------------------------------------------------------
----
bos.loc.com.utf                 4.3.2.0         USR         APPLY       SUCCESS
X11.loc.DE_DE.base.rte          4.3.2.0         USR         APPLY       SUCCESS
X11.loc.DE_DE.base.lib          4.3.2.0         USR         APPLY       SUCCESS
X11.loc.DE_DE.Dt.rte            4.3.2.0         USR         APPLY       SUCCESS
X11.loc.DE_DE.Dt.rte            4.3.2.0         ROOT        APPLY       SUCCESS
X11.fnt.ucs.ttf                 4.3.2.0         USR         APPLY       SUCCESS
bos.loc.utf.DE_DE               4.3.2.0         USR         APPLY       SUCCESS
bos.msg.DE_DE.net.tcp.cli       4.3.2.0         USR         APPLY       SUCCESS
bos.msg.DE_DE.rte               4.3.2.0         USR         APPLY       SUCCESS

---- end ----
```

The fileset bos.loc.com.utf is a requisite for the bos.loc.utf.DE_DE fileset and supplies the common locale support for UTF-8. Also, the fileset X11.fnt.ucs.ttf is pulled as a requisite by the installation process and gives you the indispensable `AIXwindows Unicode TrueType Font` support. All UTF fonts in AIX are TrueType fonts.

Now a user can begin executing in the DE_DE UTF-8 locale environment by setting the LANG environment variable to DE_DE:

```
export LANG=DE_DE
```

The `locale` command will return:

```
# locale
DE_DELANG=DE_DE
LC_COLLATE="DE_DE"
LC_CTYPE="DE_DE"
LC_MONETARY="DE_DE"
LC_NUMERIC="DE_DE"
LC_TIME="DE_DE"
LC_MESSAGES="DE_DE"
LC_ALL=
```

With this setting, all internationalized programs will execute in the German UTF-8 locale environment. If the Euro currency formatting is desired, you must change the LC_MONETARY variable in the following way:

```
export LC_MONETARY=DE_DE@euro
```

There is no need to complete the installation process and the system configuration by remapping the X server keyboard. The X server keyboard map /usr/lpp/X11/defaults/xmodmap/DE_DE/keyboard is implemented as a link to the German ISO keyboard map that is already set for your system.

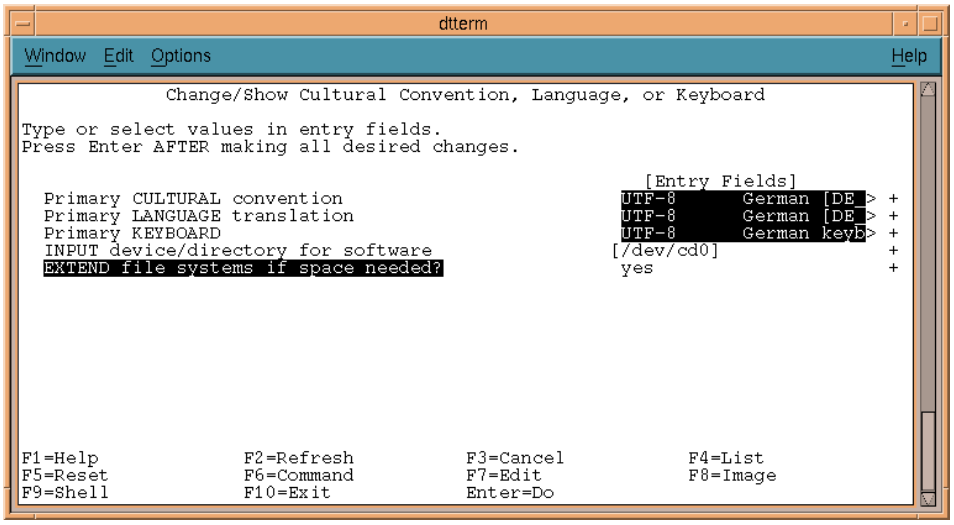
Use the AltGr+e key combination to enter the Euro symbol through your keyboard or alternatively take advantage of the UNIVERSAL Input Method, invoke the menu of the character list by Ctrl+Alt+l, and enjoy the variety of different characters that are now available to you. For more information about the new input method, refer to the Chapter 11.5.4.2, “UNIVERSAL Input Method” on page 617.

11.5.8.2 Euro UTF-8: Primary Language Environment

In order to activate Euro symbol support using the German UTF-8 locale as primary language environment, insert one of the AIX BOS CDs in the CD-ROM drive and issue:

```
# smitty chlang
```

on the command line and enter by the aide of the appropriate function keys the `UTF-8 German[DE_DE]` identifier in the relevant entry fields as shown in Figure 109.



```
              Change/Show Cultural Convention, Language, or Keyboard

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

                                                      [Entry Fields]
  Primary CULTURAL convention                        UTF-8         German [DE_> +
  Primary LANGUAGE translation                       UTF-8         German [DE_> +
  Primary KEYBOARD                                   UTF-8         German keyb> +
  INPUT device/directory for software               [/dev/cd0]                  +
  EXTEND file systems if space needed?               yes                        +

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

Figure 109. German UTF-8: Change/Show Cultural Convention, Lang., or Keyboard

SMIT uses the `chlang` command to set the environment variable LANG to DE_DE in the /etc/environment file and modifies the lft0 pseudo device driver’s predefined ODM attribute swkb_path to match the UTF-8 keymap /usr/lib/nls/loc/DE_DE.lftkeymap. If necessary the `chdev` command changes the predefined primary and secondary font path of the lft0 pseudo device driver in the ODM to be /usr/lpp/fonts/Erg22.iso1.snf and /usr/lpp/fonts/Erg11.iso1.snf respectively. Furthermore, SMIT will install all necessary filesets for the BOS locale support and you will receive the following output after SMIT completed its task:

```
...

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

  Selected Filesets
  -----------------
  bos.loc.utf.DE_DE 4.3.2.0                   # Base System Locale UTF
Code...
  bos.msg.DE_DE.net.tcp.client
  bos.msg.DE_DE.rte

  Requisites
  ----------
  (being installed automatically;  required by filesets listed above)
  bos.loc.com.utf 4.3.2.0                 # Common Locale Support - UTF-8

  << End of Success Section >>



Installation Summary
--------------------
Name                        Level           Part        Event       Result
------------------------------------------------------------------------
----
bos.loc.com.utf             4.3.2.0         USR         APPLY       SUCCESS
bos.loc.utf.DE_DE           4.3.2.0         USR         APPLY       SUCCESS
bos.msg.DE_DE.net.tcp.cli   4.3.2.0         USR         APPLY       SUCCESS
bos.msg.DE_DE.rte           4.3.2.0         USR         APPLY       SUCCESS
lft0 changed

---- end ----
The lft0 pseudo device driver does not allow for a dynamic reconfiguration,
and consequently, you must reboot the system to put the new software
keyboard map, the primary, and the secondary font paths in effect. But
prior to the reboot, it is absolutely essential to install the X11 locale
support for the new UTF-8 locale of choice on your system. The command smit
install_latest brings you to the proper SMIT menu. Fill in X11.loc.DE_DE in
the SOFTWARE to install entry filed. SMIT displays the following:
...
Installation Summary
--------------------
Name                        Level           Part        Event       Result
------------------------------------------------------------------------
-----
X11.loc.DE_DE.base.rte     4.3.2.0        USR        APPLY      SUCCESS
X11.loc.DE_DE.base.lib     4.3.2.0        USR        APPLY      SUCCESS
X11.loc.DE_DE.Dt.rte       4.3.2.0        USR        APPLY      SUCCESS
X11.loc.DE_DE.Dt.rte       4.3.2.0        ROOT       APPLY      SUCCESS
X11.fnt.ucs.ttf           4.3.2.0        USR        APPLY      SUCCESS

---- end ----
```

The fileset X11.fnt.ucs.ttf, pulled as a requisite by the installation process, gives you the AIXwindows Unicode TrueType Font support. All UTF fonts in AIX are TrueType fonts.

Optionally, you may modify the /etc/environment file to activate, on a system wide scope, the new Euro currency formatting. Add the following line:

```
LC_MONETARY=DE_DE@euro
```

Now you are ready to reboot your system. When the system comes up, and you are logged in again, you can enter the Euro symbol through the AltGr+e key combination, and by the aide of the UNIVERSAL input method, you will have access to several thousand different characters. Just use the Ctrl+Alt+l key combination, select the character list you want, and enter the characters out of the popup menu using a mouse click. If you are looking for a sample file that is actually encoded in UTF-8, install the Developers Toolkit for Unicode fileset bos.loc.adt.unicode and use the vi editor to examine the /usr/lib/nls/Unicode/samples/ut8.txt file.

11.5.8.3 IBM-1252 Code Set Euro Symbol Support

Within the framework of this test set up (LANG=en_US and German keyboard software support by swkd_path=/usr/lib/nls/loc/de_DE.lftkeymap), the IBM-1252 locale support for the BOS is already installed. The software keyboard support is contained in the same fileset as the IBM-1252 locale support: bos.loc.iso.de_DE.

You can verify that the IBM-1252 locale resides on the system. This means the locale database, the input method, the input method keymap, the LC_MONETARY locale database, and the 64-bit object modules are installed:

```
# ls -l /usr/lib/nls/loc/de_DE.IBM-1252*  | cut -c55-

/usr/lib/nls/loc/de_DE.IBM-1252
/usr/lib/nls/loc/de_DE.IBM-1252.im -> /usr/lib/nls/loc/sbcs.im
/usr/lib/nls/loc/de_DE.IBM-1252.imkeymap
/usr/lib/nls/loc/de_DE.IBM-1252@euro
/usr/lib/nls/loc/de_DE.IBM-1252@euro__64
/usr/lib/nls/loc/de_DE.IBM-1252__64
```

In case the German keyboard would not have been attached and configured you may first have to verify that the bos.loc.iso.xx_XX fileset of the desired locale is on the system. (xx_XX refers to one of the locale identifier as listed in Table 61 on page 608 modified to the extent that the first two letters are converted to lower case.) Use the `lslpp` command to accomplish this task. The lslpp command returns:

```
# lslpp -l "bos.loc.iso.*"
  Fileset                      Level  State      Description
```

```
------------------------------------------------------------------------
--
Path: /usr/lib/objrepos
  bos.loc.iso.de_DE          4.3.2.0  COMMITTED  Base System Locale ISO Code
                                                  Set - German
  bos.loc.iso.en_US          4.3.2.0  COMMITTED  Base System Locale ISO Code
                                                  Set - U.S. English
```

The bos.loc.iso.de_DE is installed, and hence, no action is required.

If for some reason the bos.loc.iso.de_DE is missing on your system, you will have to install the fileset. The command `smit install_latest` brings you to the proper SMIT menu. Fill in bos.loc.iso.de_DE in the `SOFTWARE to install` entry field and press Enter. SMIT reports the progress.

Since you verified the existence of the BOS de_DE locale support, add the X11 locale support next. The package that must be installed is named X11.loc.de_DE, and it is recommend to use SMIT in conjunction with the fastpath `install_latest` to lay the foundation for the graphical locale support. Fill in X11.loc.de_DE in the `SOFTWARE to install` entry field, let SMIT do the work, and wait until you see the following lines at the end of the output:

```
...
Installation Summary
--------------------
Name                        Level           Part        Event       Result
------------------------------------------------------------------------
----
X11.loc.de_DE.base.rte      4.3.2.0         USR         APPLY       SUCCESS
X11.loc.de_DE.base.lib      4.3.2.0         USR         APPLY       SUCCESS
X11.loc.de_DE.Dt.rte        4.3.2.0         USR         APPLY       SUCCESS
X11.loc.de_DE.Dt.rte        4.3.2.0         ROOT        APPLY       SUCCESS

---- end ----
```

Set your LANG environment variable to de_DE.IBM-1252, export LANG and open a new dtterm that is aware of your new locale by the `/usr/dt/bin/dtterm` command. Note, that you do not have to introduce a new keymap to the X server since you are already using the German ISO keymap. Now you can enter the Euro symbol by the AltGr+e key combination. The Euro currency formatting will be to your service as soon as the LC_MONETARY environment variable is set and exported:

```
# export LC_MONETARY=de_DE.IBM-1252@euro
```

It should be mentioned that there is intentionally no SMIT support through the fast paths `chlang` and `mle_add_lang`. for the SBCS migration option. This means the smit menus **Change/Show Primary Language Environment** (`smit chlang`) and **Add Additional Language Environments** (smit mle_add_lang) will not allow you to establish the Euro symbol support through the IBM-1252 code set.

11.6 National Language Enhancements

The following enhancements have been made to national language support in AIX Version 4.3, AIX Version 4.3.2 and AIX Version 4.3.3.

11.6.1 Byelorussian and Ukrainian Localization

AIX Version 4.3 has introduced support for the Byelorussian and Ukrainian localization. Both languages use the existing ISO8859-5 codeset that contains all of the characters needed to display the languages. The Localization provides the following items:

- A new input method for the input of characters to the Byelorussian and Ukrainian languages.
- Localized conventions are defined for Byelorussian and Ukrainian imkeymaps and lftkeymaps.
- Localized resource files for X clients.
- Additions to X locale database (I18N).
- Localized X keyboard mapping.

11.6.2 Thai Language Support

Previous to AIX 4.3, Thai language support was based on a solution from IBM Thailand. AIX Version 4.3 formally implements support for the Thai language.

The features implemented in AIX Version 4.3 include:

Locale The Thai locale is based on either TIS620-1 or Unicode Version 2. The wide character processing for UTF-8 Thai locales is based on Unicode.

Input Method

The Thai input method provides users with a facility for composing characters by combining several key strokes to produce a single Thai character.

cmdpios	AIX Version 4.3 provides a printer filter for Thai/Vietnamese that supports downloadable UCS-2 fonts and can print data for both Thai and UTF-8 locales.
cmdiconv	New iconv converters are present for the following Thai language codesets:
TIS620-1	
Unicode	
charmap	New character map for Thai code set definition.
cmdtty	Thai csmap is included in AIX Version 4.3.
xmodmap	Localized X keysyms and server keyboard maps are present for Thai.
Aixterm	Enhanced to support the Thai language.
imkeymap	AIX Input Method keymaps (imkeymaps) for the Thai language have been added.
Motif	Motif contains enhancements to support of Thai language.
Unicode	Unicode support for the Thai language has been added.

11.6.2.1 Systems Management

The Thai locales are configurable just like any other locale in the system using the SMIT panel Manage Language Environment.

The user can install the following items:

- Thai Cultural Conventions (TIS620-1)
- Thai Cultural Conventions (Unicode)

11.6.2.2 Standards Compliance of Thai Language Support

The UTF-8 support for the Thai locale consists of the entire Base Multilingual Plane of ISO 10646. It complies with the following standards:

- X/Open Portability Guide, issue 3 and issue 4
- POSIX.2
- X Window System Version 11 Release 5
- Unicode Version 2.0 for display and fonts

Note: The localization components for this locale will only support the Thai and ISO8859-1 characters.

11.6.2.3 Application Binary Interface (ABI)

The Thai locale, input method, and conversions comply with the AIX NLS subsystem loadable interfaces.

11.6.2.4 Application Programming Interfaces (API)

AIX Input Method defines the API that is used by other programs to support multi-byte input. The Thai input method complies with the AIX Input Method APIs.

The Universal input method treats the Thai input method as another option in the list of input methods.

11.6.3 Vietnamese Language Support

Previous to AIX 4.3, Vietnamese language support was based on a solution from IBM Vietnam. AIX Version 4.3 formally implements support for the Vietnamese language.

The features implemented in AIX Version 4.3 include:

Locale The Vietnamese locale is based on either IBM-1129 or Unicode Version 2. The wide character processing for UTF-8 Vietnamese locales is based on Unicode.

Input method

In Vietnam, it is not possible to assign each character to a key on the keyboard. The input method provides a character composing mechanism for the user. In other words, several key combinations can be used to build or compose a character. The Vietnamese input methods are based on IBM 461 A/B on a 101 key US keyboard where some of the Vietnamese and French characters are mapped to the first and second rows.

cmdiconv

New iconv converters are present for the following Vietnamese Language code sets:

- IBM-1129
- Unicode
- EBCDIC Code (IBM-1130)
- PC Code (IBM-1258, which is the same as Windows Code page)
- TCVN2 (current standard code set used in Vietnam)

charmap	New character map for Vietnamese code set definition.
cmdpios	AIX Version 4.3 provides a printer filter for Vietnamese that supports downloadable UCS-2 fonts and can print data for both IBM-1129 and UTF-8 locales.
libmeth	Locale-specific methods for wide character processing (based on Unicode).
cmdtty	Vietnamese csmap.
i18n	The Vietnamese language has been added to the X locale database.
xmodmap	Localized X keysyms and server keyboard maps are present for Vietnamese.
CTL layout engine	Vietnamese makes use of combining zero width characters that require the use of layout services as defined by the CTL APIs.

11.6.3.1 Systems Management

The Vietnamese locales are configurable through the SMIT panel Manage Language Environment.

The user can install the following items:

- Vietnamese Cultural Conventions (IBM-1129)
- Vietnamese Cultural Conventions (Unicode)

11.6.3.2 Standards Compliance of Vietnamese Language Support

Vietnamese Language Support follows the standards that the AIX NLS Architecture follows, especially those standards listed below. The UTF-8 support for the Vietnamese locale consists of the portion of the Base Multilingual Plane of ISO 10646 that is supported by other locales on AIX. The relevant standards and resources are:

- X/Open Portability Guide, issue 3 and issue 4
- POSIX.2
- X Window System Version 11 Release 6
- IBM 1129 for internal Code Page
- IBM 461 A/B for Keyboard layout
- IBM 1130 for EBCDIC code Convert
- Unicode Version 2.0 for display and fonts

11.6.3.3 Migration

Current users of the country solution for Vietnam are expected to migrate to AIX Version 4.3 with the Vietnamese localization support. The new Vietnamese Language support is binary-compatible with the previous country solution for Vietnam.

11.6.4 Japanese Code Page 943 (AIX 4.3.2)

IBM-943 is a compatible code set for the Japanese Microsoft Windows environment. This code set is known as ’83 ordered Shift JIS.

In Japan, most of operating systems on PC clients support JIS 83 ordered Shift JIS code set (IBM-943) to implement Japanese. AIX has supported JIS 78 ordered Shift JIS code (IBM-932) since AIX Version 3. This was one of the advantages in enabling a UNIX-PC mixed environment. Recently, most Japanese PCs use JIS 83 ordered Shift JIS code set as the default, and this causes a data incompatibility between PC and AIX environments.

AIX 4.3.2 supports the IBM-943 code set that is needed for PC interoperability. The code page 943 supports interoperability with Microsoft Windows clients in Japan and makes AIX Version 4.3.2 the preferred release for Japan.

The difference between IBM-932 in previous AIX versions and IBM-943 is as follows:

- New JIS sequence ('83 ordered).
- NEC selected characters (83) are added.
- NEC's IBM selected characters (374) are added.
- Two new characters are added (defined by JIS90).

Ja_JP is used for a short locale name of the IBM-943 locale. The short locale name of IBM-932 is no longer supported.

If you want to use IBM-932 rather than IBM-943, you can use the IBM-932 by specifying the long locale name Ja_JP.IBM-932.

11.6.4.1 Installation and Packaging

Following list describes the installation and packaging of IBM-943 locale:

- The IBM-943 locale is selectable for installation from SMIT.
- Any locale support that is common to all Japanese locales are packaged and shipped in the existing fileset bos.loc.com.JP.

- The IBM-943 locale is packaged in bos.loc.pc.Ja_JP. This fileset contains both IBM-932 and IBM-943 locales.
- The IBM-943 locale-specific X11 resource is shared with the IBM-932 X11 resource.

11.6.5 Korean TrueType Font (AIX 4.3.2)

TrueType is the scalable font technology built into Windows and Macintosh workstations. AIX 4.3.2 now supports Korean TrueType fonts in the AIX X environment.

A TrueType rasterizer is available in the AIXwindows environment in AIX 4.3. All scalable font systems need a rasterizer to convert their glyph outlines into bitmaps suitable for direct copying to the screen.

The TrueType rasterizer in AIX 4.3.2 supports TrueType 1.0 Font Files and most of the character sets currently defined in the Unicode 2.0 specification.

There are three kinds of extensions for the font file:

TTF	The recommended file extension for TrueType font files.
TTC	TrueType Collection file. A scheme where multiple TrueType fonts can be stored in a single file, typically used when only a subset of glyphs changes among different designs. They are used in Japanese fonts, where the Kana glyphs change but the Kanji remain the same.
OTF	The recommended file extension for the Type 1 font (not for TrueType).

Changes to existing TTF rasterizer is described as follows:

- Korean support added
- Font types added: Batang, Dotum, and Sammul
- Font sizes supported: 9, 10, 12, 18, 24, 32, 40, 48, and 72
- Application support: X Window application, Netscape, and dtterm

AIX has two locales related with Korean: ko_KR (IBM-eucKR) and KO_KR (Unicode). In AIX 4.3.2, Korean TrueType font rasterizer works only on ko_KR locale based on ksc5601.

The TrueType rasterizer is available in the AIXwindows font library, which is shipped with both the AIXwindows Font Server and X server.

The Korean TrueType font file supplied in AIX 4.3.2 is a TrueType collection file that has two TrueType Korean font sets. One is proportional, and the other is monospaced. The font file name is hmfmm.ttc. This font file is packaged in the X11.fnt.ksc5601.ttf fileset. It is installed in the /usr/lib/X11/fonts/TrueType directory.

11.6.5.1 Standards

The AIXwindows Font server supports the X window System Font Protocol Version 2.0 and X Version 1.1 Release 6.1.

The TrueType rasterizer supports TrueType 1.0 font files.

The name of the AIX windows Font server in AIX 4.3 is changed from fs to xfs. Also, the font server's port number is changed from 7500 to 7100. These changes between X11R5 and X11R6 were made by the X consortium, suppliers of the X Window System technology.

11.6.5.2 Installation and Packaging

The TrueType rasterizer is automatically installed as part of the AIX window X server and font server.

The following is some information about Korean True Type font packaging:

X11.fnt.fontServer
The TrueType rasterizer is shipped in the AIXwindows font server, which can serve fonts to different X servers across the network.

X11.base.rte
This fileset ships the AIXwindows X server. The pre-req.s for this fileset are not be updated since the X server serves mostly local clients. If remote clients require additional fonts, then the X server should be configured to use an AIXwindow font server configured with the proper converters.

X11.samples.fnt.util
This fileset includes the ttfinfo sample utility, which displays the header information of a TrueType font.

11.7 Additional Languages (4.3.3)

AIX 4.3.3 introduces the following new localizations:

- Italian for Switzerland (it_CH and IT_CH)
- English for Australia (en_AU and EN_AU)

- English for Belgium (en_BE and EN_BE)
- English for South Africa (en_ZA and EN_ZA)

Each new localization is shipped in its own fileset and provides both a single byte version based on the ISO8859-15 codeset and a Unicode (UTF-8) version.

11.7.1 64-bit Localized Objects (4.3.3)

On AIX 4.3.3, there are several enhancements that enables developing 64-bit internationalized applications. They include:

- 64-bit support for layout objects

  Layout objects include functions that are used when characters are input bidirectional in such languages as Arabic and Hebrew. AIX 4.3.3 supports 64-bit applications that use such languages.

- 64-bit input methods support

  64-bit input methods are supported on AIX 4.3.3. 64-bit applications can be developed using these input methods.

64-bit locale and iconv have been already supported on AIX 4.3.2.

11.7.2 Traditional Chinese Unicode Input Methods(4.3.3)

AIX 4.3.3 supports traditional Chinese Unicode input methods. There are six different input methods:

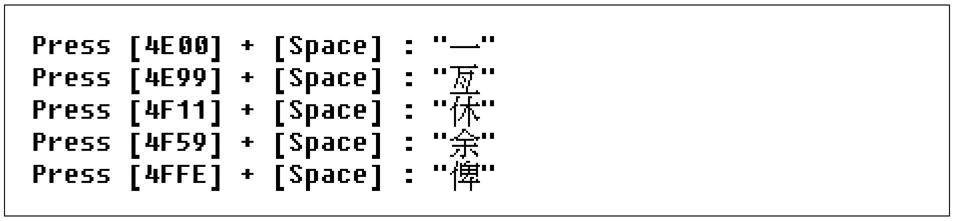
- Internal Code Input Method
- Phonetic Input Method
- Dayi Input Method
- Array Input Method
- Tsangjye Input Method
- Boshiami Input Method

11.7.2.1 Internal Code Input Method

Users can input Chinese characters using this input method by inputting four-digit hexadecimal UCS-2 codes.

[A-F], [0-9], and [Space] keys are used for internal code input method.

Example



```
Press [4E00] + [Space] : "一"
Press [4E99] + [Space] : "亙"
Press [4F11] + [Space] : "休"
Press [4F59] + [Space] : "余"
Press [4FFE] + [Space] : "俾"
```

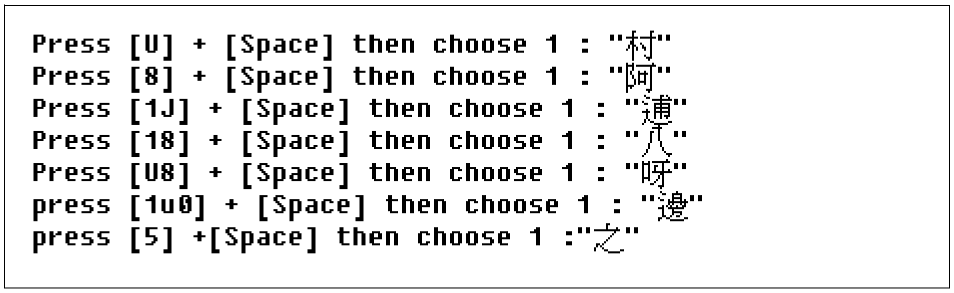
Figure 110. Internal Code Input Method

11.7.2.2 Phonetic Input Method

This input method allows user to input Chinese characters based on phonetic sound of each character. Each sound of every character is formed a combination of a consonant, a mid-sound, a vowel and a respective tone symbol. In some cases each pronunciation will have many corresponding Chinese characters. Users will then select the character based on its meaning.

[A-Z], [0-9], [,], [.], [/], [?], and [Space] keys are used for Phonetic input method. Each character has corresponding phonetic symbol and tone symbol.

Example



```
Press [U] + [Space] then choose 1 : "村"
Press [8] + [Space] then choose 1 : "阿"
Press [1J] + [Space] then choose 1 : "逋"
Press [18] + [Space] then choose 1 : "八"
Press [U8] + [Space] then choose 1 : "呀"
press [1u0] + [Space] then choose 1 : "邊"
press [5] +[Space] then choose 1 :"之"
```

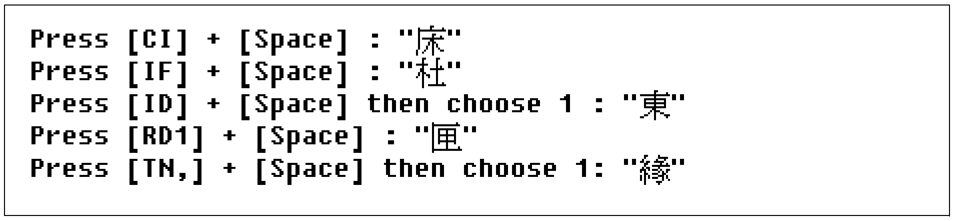
Figure 111. Phonetic Input Method

11.7.2.3 Dayi Input Method

The Dayi input method is according to the Dayi rules of inputting Dayi defined radicals. When there are more than one character in each input radical order, users have to choose the one they desire from the candidates.

[A-Z], [,], [;], [.], [/], [0-9], and [Space] keys are used for Dayi input method. Each key corresponds to a Dayi radical. The radical keys are categorized to seven categories, which are people, animals, agriculture, nature, craft work, the five elements and others.

Example



```
Press [CI] + [Space] : "床"
Press [IF] + [Space] : "杜"
Press [ID] + [Space] then choose 1 : "東"
Press [RD1] + [Space] : "匣"
Press [TN,] + [Space] then choose 1: "緣"
```

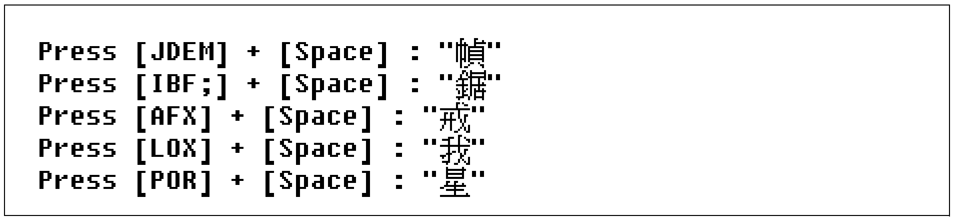
Figure 112. Dayi Input Method

11.7.2.4 Array Input Method

The array input method is according to the Array rules of inputting Array defined radicals. When there are more than one character in each input radical order, users have to choose the one they desire from candidates.

[A-Z],[,],[.],[/] and [Space] keys are used for Array input method. Each key except [W] owns 10 first grade simple codes (290 in total). When the first radical key is pressed, 10 first grade simple codes are shown immediately. By pressing a number key, user can input a corresponding character from candidates. If [Space] is pressed, the process of main conversion table lookup is taken effect to search matched characters. In other words, the input is determined. User can input radical keys at most five times in row before pressing [Space] to determine the Chinese character. [W] is used to input symbol characters.

Example



```
Press [JDEM] + [Space] : "幀"
Press [IBF;] + [Space] : "鋸"
Press [AFX] + [Space] : "戒"
Press [LOX] + [Space] : "我"
Press [POR] + [Space] : "星"
```

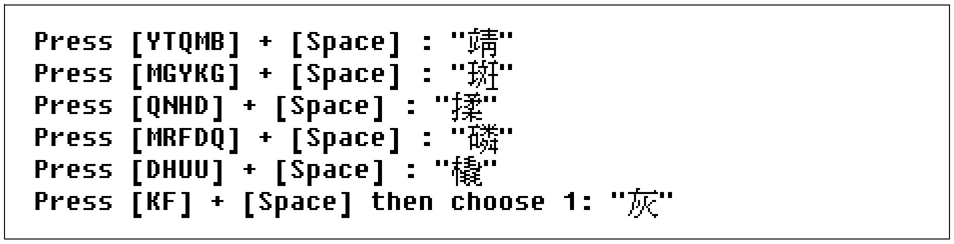
Figure 113. Array Input Method

11.7.2.5 Tsangjye Input Method

The Tsangjye input method is according to the Tsangjye rule to input Tsangjye defined radicals. When there are more than one characters in each input radical order, users have to choose the one they desire from candidates.

[A-Z] and [Space] keys are used for Tsangjye input method.

Example



```
Press [YTQMB] + [Space] : "靖"
Press [MGYKG] + [Space] : "斑"
Press [QNHD] + [Space] : "揉"
Press [MRFDQ] + [Space] : "磷"
Press [DHUU] + [Space] : "檮"
Press [KF] + [Space] then choose 1: "灰"
```

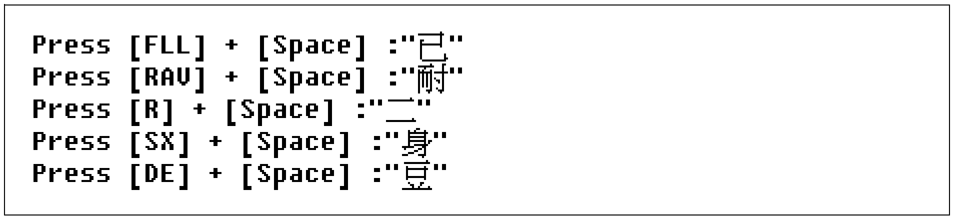
Figure 114. Tsangjye Input Method

11.7.2.6 Boshiami Input Method

The Boshiami input method is according to Boshiami rule based on shape, pronunciation and meaning of each character to input radicals. The radicals in Boshiami are English alphabetic letters. When there are more than one character in each input radical order, users have to choose the one they desire from the candidates.

[A-Z] and [Space] keys are used to input Chinese characters and [,.?[]] keys are used to input symbols.

Example



```
Press [FLL] + [Space] :"已"
Press [RAV] + [Space] :"耐"
Press [R] + [Space] :"二"
Press [SX] + [Space] :"身"
Press [DE] + [Space] :"豆"
```

Figure 115. Boshiami Input Method

11.7.3 Korean Input Method Enhancements (4.3.3)

AIX 4.3.3 supports Korean Unicode input method. As a part of Unicode support function in AIX 4.3.3, Korean Unicode input method provides the capability of input all Korean characters defined in KSC 5700.1995 which is based on UCS 2.0. The Hangul code set includes Hangul JAMO, Hangul and Hanja characters.

One Hangul character can comprise several consonants and vowels which are called Hangul JAMO. Each Hanja character has its own meaning and is thus more specific than Hangul.

Users can input 11,172 Hangul characters, 7,744 Hanja characters, and JAMO characters with this input method.

Korean Unicode input method provides following capabilities:

- Compound consonants and compound vowel can be input as one character.
- Half-width and full-width character input supports ASCII characters in both single-byte and multibyte modes.
- Special characters can be input by conversion from JAMO or code input function.
- Over-the-spot pre-editing drawing area. Allows intermediate characters in reverse video area that temporarily covers the text line.
- The complete character is sent to the editor by pressing the conversion key.

11.7.4 Internationalized Classes for Unicode (4.3.3)

AIX 4.3.3 provides internationalized classes for Unicode (ICU), a set of API that is being provided on all IBM platforms. The ICU packages is based on the Java/Unicode work originally done by Taligent. The ICU provides a wide variety of APIs that allow C or C++ language programmers to handle Unicode data and perform manipulation and formatting tasks in a platform independent fashion.

By using ICU, you can develop and maintain one application that supports a wide variety of national languages. When a an application is internationalized by ICU, all the text shown to the user is in the native language, and user expectations are met for dates, times, and other locale conventions.

11.7.4.1 ICU Function

The ICU has following function:

- UnicodeString support the Unicode 2.0.14 standard
- Resource bundles

A resource bundle provides a general mechanism that allows you to access strings and other objects according to locale conventions. It provides a mapping from <key,locale> to <value>. It also supplies inheritance among locale resources that allows you to minimize duplication across countries.

- Number formatters

  Number formatters provide capabilities for converting binary numbers and currencies into text strings for the meaningful display for the locales.

- Date and time formatters

  Data and time formatters provide capabilities for converting internal time data into text strings for the meaningful display for the locales.

- Message formatters

  Message formatters provide capabilities for putting together sequences of strings, number dates, and other format to create messages.

- Text collation support

  ICU supports language sensitive comparison of strings.

- Text boundary analysis

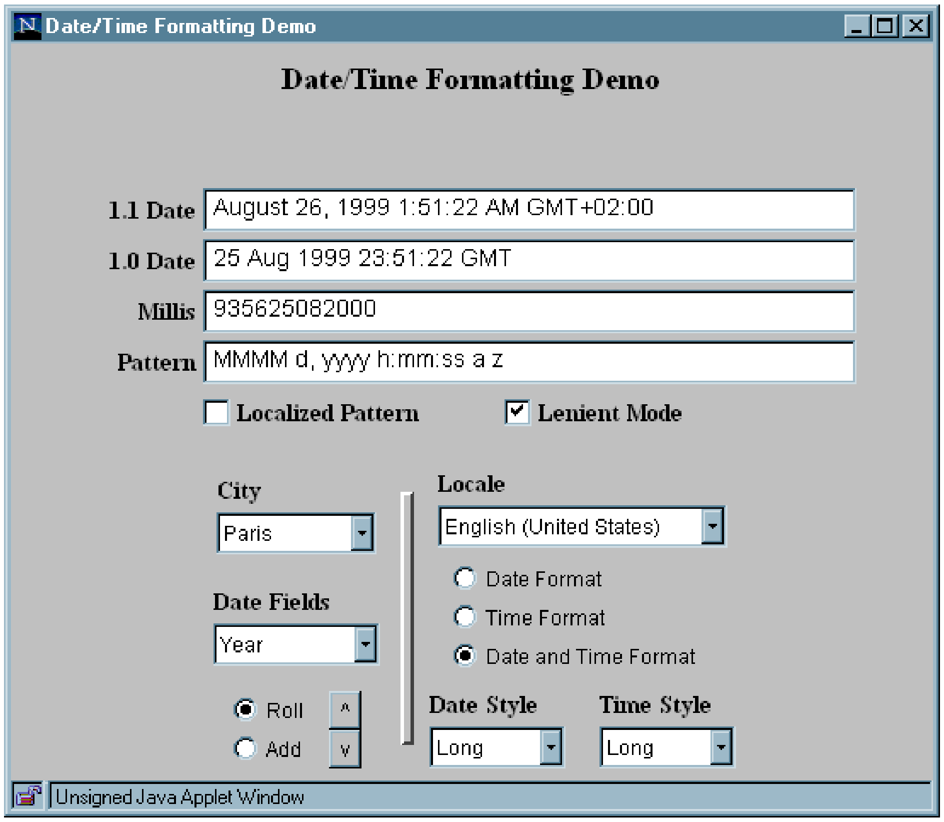
  ICU provides capabilities for finding characters, word, and sentence boundaries. For example, you can find a word even though the word is not bounded by spaces.

The following locales are supported:

ar, be, bg, ca, cs, da, de, de_AT, de_CH, el, en, en_AU, en_CA, en_GB, en_IE, en_NZ, en_ZA, es, es_AR, es_BO, es_CL, es_CO, es_CR, es_DO, es_EC, es_GT, es_HN, es_MX, es_NI, es_PA, es_PE, es_PR, es_PY, es_SV, es_UY, es_VE, et, fi, fr, fr_BE, fr_CA, fr_CH, hr, hu, is, it, it_CH, iw, ja, ko, lt, lv, mk, nl, nl_BE, no, no_NO_B, no_NO_NY, pl, pt, pt_BR, ro, ru, sh, sk, sl, sq, sr, sv, tr, uk, zh, zh_TW.

11.7.4.2 Sample Application

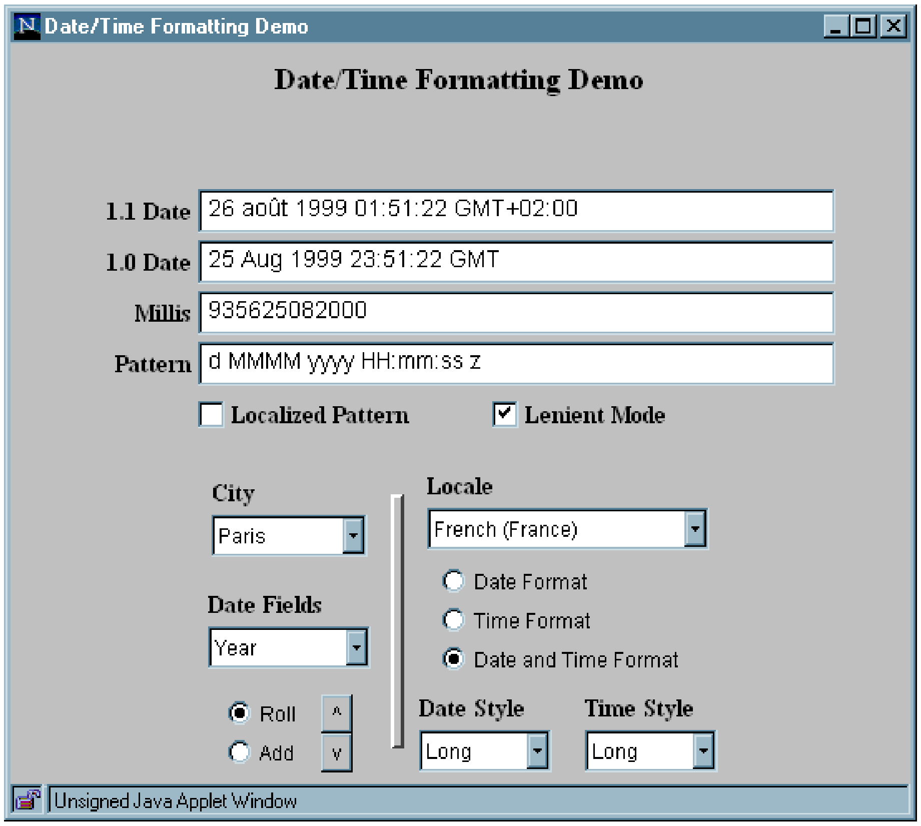
Figure 116 and Figure 117 shows the ICU capabilities to handle multiple locales in one application. The applications is written using the Java version of ICU.



1.1 Date: August 26, 1999 1:51:22 AM GMT+02:00
1.0 Date: 25 Aug 1999 23:51:22 GMT
Millis: 935625082000
Pattern: MMMM d, yyyy h:mm:ss a z
[ ] Localized Pattern [✓] Lenient Mode
City: Paris
Date Fields: Year
[● Roll] Add
Locale: English (United States)
Date Format / Time Format / [● Date and Time Format]
Date Style: Long
Time Style: Long
Unsigned Java Applet Window

Figure 116. ICU English Locale Example

In Figure 116, the date in first column is displayed in English conforming to U.S. conventional date and time representation order.


Date/Time Formatting Demo

Date/Time Formatting Demo

1.1 Date: 26 août 1999 01:51:22 GMT+02:00

1.0 Date: 25 Aug 1999 23:51:22 GMT

Millis: 935625082000

Pattern: d MMMM yyyy HH:mm:ss z

☐ Localized Pattern ☑ Lenient Mode

City: Paris

Locale: French (France)

○ Date Format
○ Time Format
● Date and Time Format

Date Fields: Year

● Roll
○ Add

Date Style: Long

Time Style: Long

Unsigned Java Applet Window

Figure 117. ICU French Locale Example

By changing the locale to French (Figure 117), the date is displayed in French conforming to French conventional date and time representation order.

11.7.4.3 AIX Integration

The ICU consists of shared libraries, header files, data files, and a utility program. They are installed to /usr/intlwork directory. The fileset bos.loca.adt.icu contains the ICU package. Applications using the ICU can be both 32-bit and 64-bit applications.

11.8 Documentation Search Service: DBCS HTML Search (4.3.2)

The AIX Documentation Search Service is extended to add the capability to search specified Double Byte Character Set (DBCS) codesets in Japanese, Korean, Simplified Chinese, and Traditional Chinese.

The doublebyte languages and code sets supported by Documentation Search Service (docsearch) in AIX 4.3.2 are listed in the Table 67. For a complete list of supported languages, codesets, and locales, see the language support table section in *AIX Version 4.3 General Programming Concepts: Writing and Debugging Programs*, SG23-2533.

Table 67. Additional Doublebyte Support in Docsearch

Language	**CCSID**	**Codeset**	**Locale**
Japanese	932 *	IBM-932 *	Ja_JP *
Korean	970	IBM-eucKR	ko_KR
Simplified Chinese	1383	IBM-eucCN	zh_CN
Traditional Chinese	950	big5	Zh_TW
* Note: The Ja_JP locale uses the 943 codeset and CCSID, however the Documentation Search Service currently supports 932.			

The Document Search Service is browser and Web-server based. It allows you to search registered HTML documents using a search form that appears in the Web browser. When you type words into the search form, the service searches for those words and then presents a search results page containing links that lead to the documents containing the target words.

The browser of a desired language code set is both used for display of text and the form based text entry fields.

The AIX search form allows you to search all documents that are registered on a host. Before any document can be searched using the documentation search service, it must have an index created, and the index must be registered with the search service. Some applications ship prebuilt document indexes inside their install package. When the application is installed, the indexes are automatically registered. The AIX Version 4.3 documentation and the Web-Based System Manager application both ship prebuilt indexes for their documents. You can also create indexes for your own HTML documents and register them with the search engine so that they can be searched on line. For further information on how to create indexes, see AIX Documentation Search Service in *AIX Version 4.3 General Programming Concepts: Writing and Debugging Programs,* SC23-2533.

If more than one language is installed, you can switch to other installed languages. The Documentation Search Service GUI is consistent across languages. Although there may be some locale differences such as the sort technique and display layout.

Functions available for nongraphical displays will depend on function provided by the browser used on the nongraphical display. For example, if an ASCII browser does not support the code sets used, the ASCII user of that browser will not be able to search the translated documents written in that code set.

The function for the double byte languages is implemented by the DBCS IMNSearch search engine. DBCS search duplicates the function provided for single byte languages. The functions of both the single and double byte CGIs in AIX 4.3.2 are the same as that of the AIX 4.3.0 single byte CGI with the following conditions noteworthy:

- Two languages cannot be displayed or searched at the same time. Documentation Search Service GUIs only display a single language at a time. Only the indexes of the current language are visible within the GUI for selection and searching. A selection menu button in the search form allows you to switch between languages.

  For example, if the GUI search from is being displayed in Spanish, then only Spanish books will show in the "select volumes to search" section at the bottom of the form. It is not possible to display indexes from different code sets on the same HTML page and have them display correctly.
- The results page adds a start-over button that allows you to jump back to the home search page from any results page.

Translation of the changes and additions to Documentation Search Service messages, GUI, and help search form are done by program integrated information (PII).

11.8.1 Documentation Libraries

The AIX 4.3.2 documentation library consists of two CDs. The first is the AIX Base Documentation in HTML format and the second is the AIX Extended Documentation in HTML and PDF format. As of this writing, the AIX Extended Documentation is only available in English.

The base documentation library contains most of the AIX user, system administrator, and application programmer guides. This library also contains basic reference documents such as the Commands Reference, Files Reference, and Technical Reference volumes intended for application programmers.

The extended documentation library contains books concerning adapters, books intended for system programmers, and technical specifications describing industry standards.

Most of the documentation in these libraries is in HTML format and must be viewed using an HTML version 3.2-compliant Web browser, such as the Netscape Navigator 4.0 browser. A few documents in these libraries are in PDF format and must be viewed using the Adobe Acrobat Reader, Version 3.0. Documentation Search Service does not support PDF files. The Netscape Navigator browser and Acrobat Reader Version 3.0.1 are shipped with AIX 4.3.2 Bonus Pack.

The HTML documents can be searched using the Documentation Search Service, bos.docsearch, an optionally installable component of the AIX base operating system. It is highly recommended to install and configure the Documentation Search Services since this service may be used by other applications installed on the system. Chapter 10, “Online Documentation” on page 565 for installation information.

Online documentation is also available at:
http://www.rs6000.ibm.com/aix/library

11.8.2 Limitations

East European locales are not supported for searching, however product libraries may be available if they are translated.

For the supported locales, it is not necessary to do any conversions for the components to communicate with each other.

11.8.3 Invoking Documentation Search Service

The global search form can be accessed by:

- Type `docsearch` on the command line
- Click the Documentation Search Service icon in the CDE Desktop Help subpanel.

The Documentation Search Service desktop action and the `docsearch` command will both check for the existence of an environment variable DOC_LANG. This variable is used to define the language in which to display the search form GUI. It also specifies the language in which the search engine will conduct the search. If not defined, the language used will be that of the launching environment, if possible.

Set the DOC_LANG environment variable with the language locale you want to search. The search form will only display indexes of the currently selected language. The hit list will return hits from all books that were selected for search in the search form. The `chdoclang` command will set the DOC_LANG environment variable for you. See the AIX 4.3.2 Release Notes for more information on the `chdoclang` command.

The language selection menu in the search form allows you to select the language of the GUI. When a language is selected, the GUI is displayed in that language and displays only the indexes for that language. The selection menu is intelligent in that its contents are dynamically generated and it will display only those languages that are currently available on the system being used.

11.8.3.1 Japanese Documentation Search

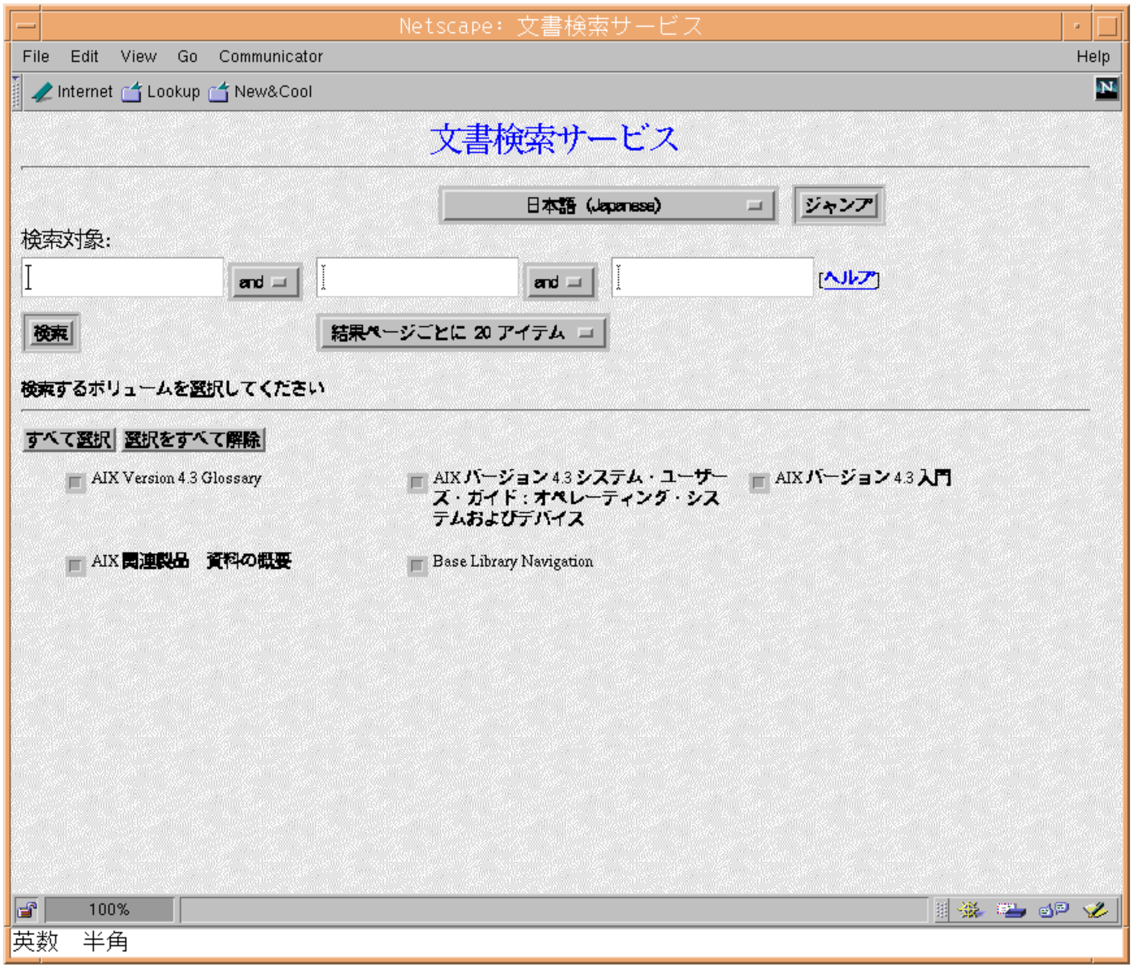
A Japanese search can be done in a Japanese CDE environment. If you are in another language's CDE environment, you need to set LANG to Ja_JP and map the keyboard using the following command:

```
#xmodmap /usr/lpp/X11/defaults/xmodmap/Ja_JP/keyboard
```

When you enter:

```
#docsearch
```

you get the screen such as shown in Figure 118 on page 656.



文書検索サービス

日本語 (Japanese)

検索対象:

結果ページごとに 20 アイテム

検索するボリュームを選択してください

AIX Version 4.3 Glossary

AIX バージョン 4.3 システム・ユーザーズ・ガイド：オペレーティング・システムおよびデバイス

AIX バージョン 4.3 入門

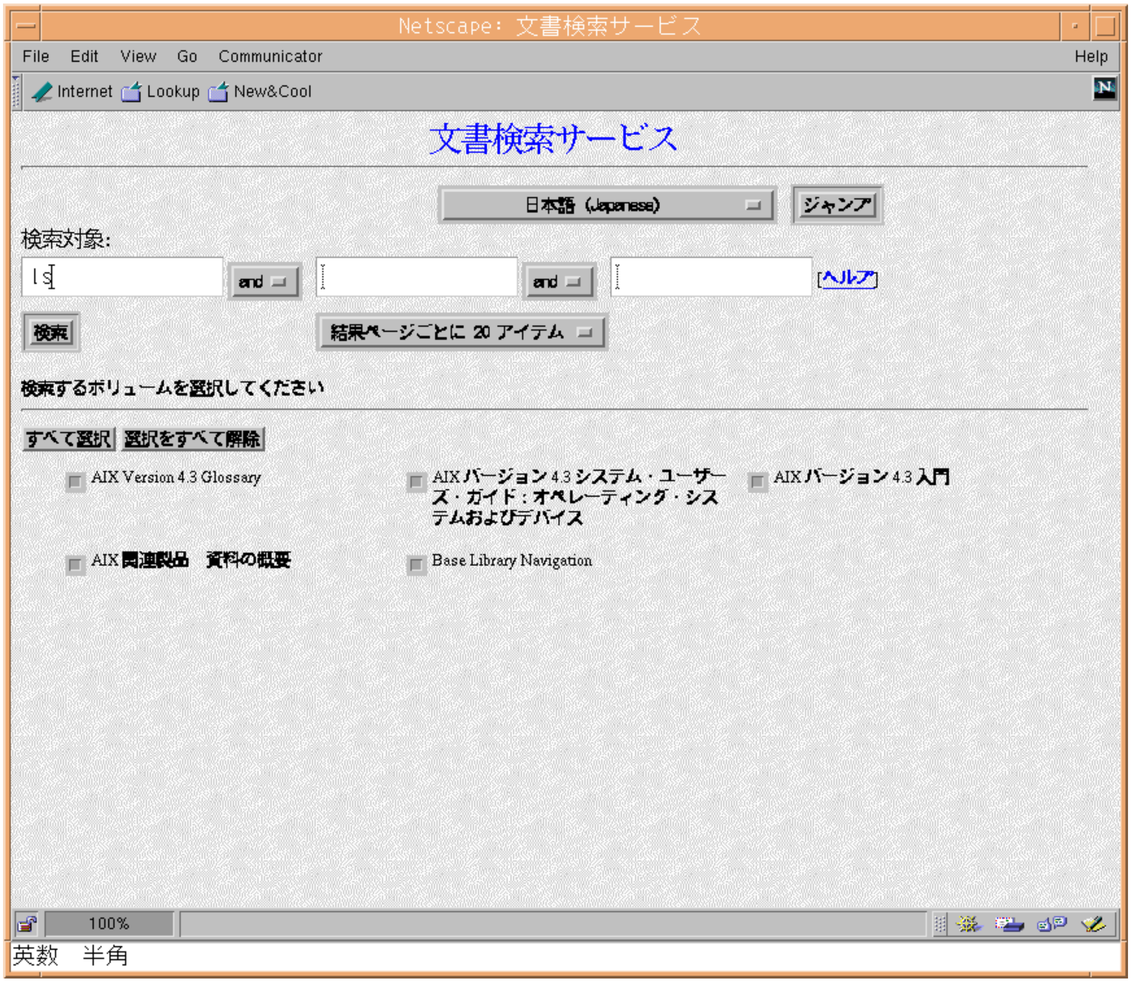
AIX 関連製品　資料の概要

Base Library Navigation

英数　半角

Figure 118. Japanese Search Form

After remapping the keyboard, using a US keyboard, you can input the words you want to search into the search field. Note that the entire product library is not installed for this example. Figure 119 shows an example of searching for "ls".



文書検索サービス

検索対象:

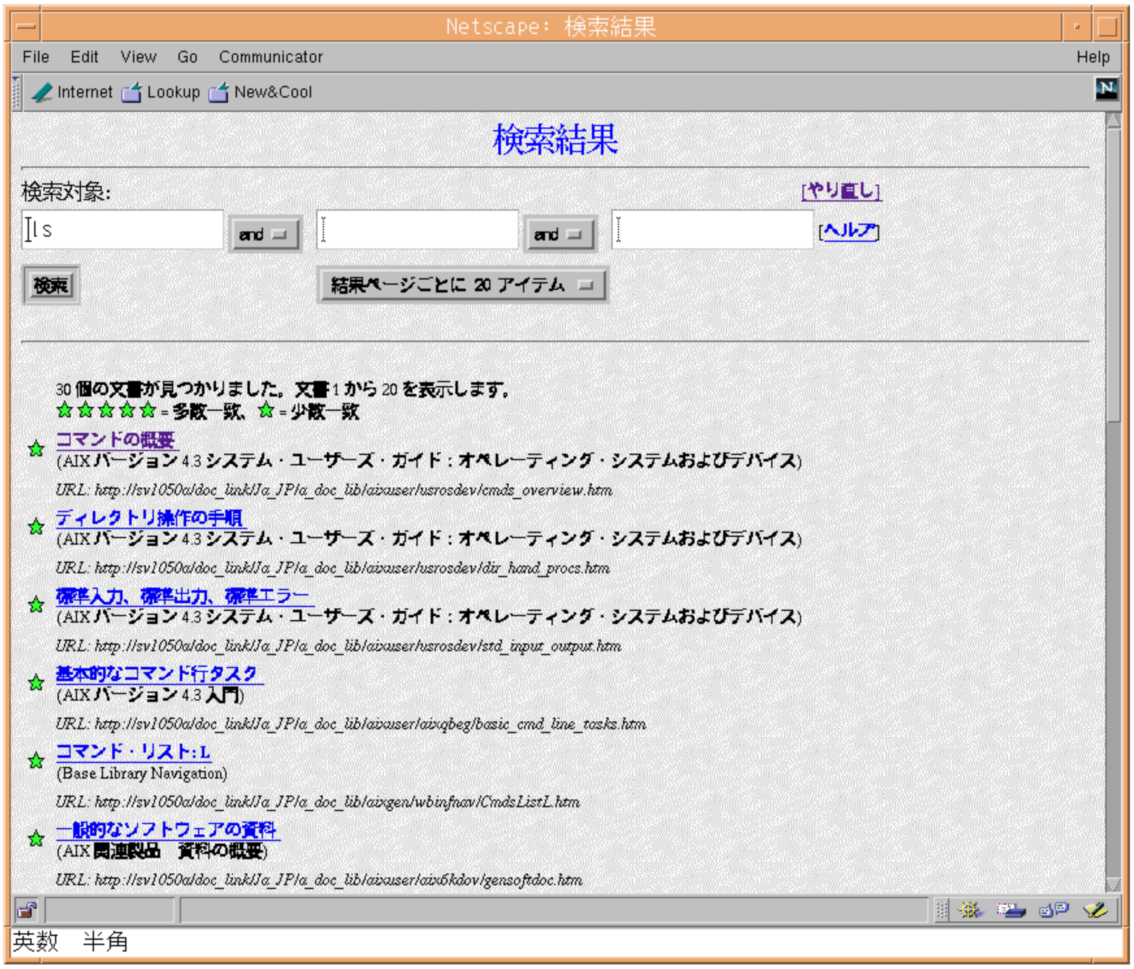
ls

検索するボリュームを選択してください

- AIX Version 4.3 Glossary
- AIX バージョン 4.3 システム・ユーザーズ・ガイド：オペレーティング・システムおよびデバイス
- AIX バージョン 4.3 入門
- AIX 関連製品 資料の概要
- Base Library Navigation

Figure 119. Searching Japanese Documentation

The resulting page is shown in Figure 119 on page 657.



検索結果

検索対象: [やり直し]

ls and and [ヘルプ]

検索 結果ページごとに 20 アイテム

30 個の文書が見つかりました。文書 1 から 20 を表示します。
★★★★★ = 多数一致、★ = 少数一致

★ コマンドの概要
(AIX バージョン 4.3 システム・ユーザーズ・ガイド：オペレーティング・システムおよびデバイス)
URL: http://sv1050a/doc_link/Ja_JP/a_doc_lib/aixuser/usrosdev/cmds_overview.htm

★ ディレクトリ操作の手順
(AIX バージョン 4.3 システム・ユーザーズ・ガイド：オペレーティング・システムおよびデバイス)
URL: http://sv1050a/doc_link/Ja_JP/a_doc_lib/aixuser/usrosdev/dir_hand_procs.htm

★ 標準入力、標準出力、標準エラー
(AIX バージョン 4.3 システム・ユーザーズ・ガイド：オペレーティング・システムおよびデバイス)
URL: http://sv1050a/doc_link/Ja_JP/a_doc_lib/aixuser/usrosdev/std_input_output.htm

★ 基本的なコマンド行タスク
(AIX バージョン 4.3 入門)
URL: http://sv1050a/doc_link/Ja_JP/a_doc_lib/aixuser/aixqbeg/basic_cmd_line_tasks.htm

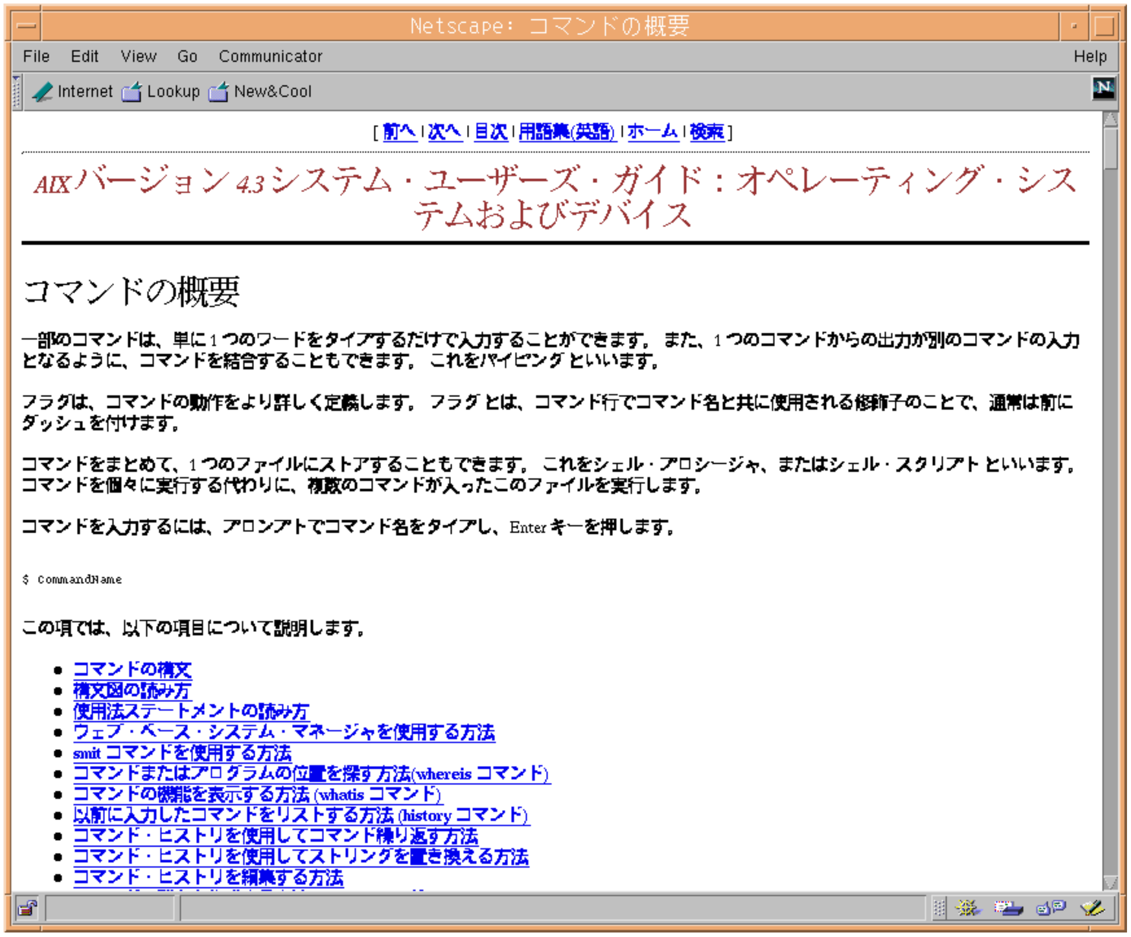
★ コマンド・リスト:L
(Base Library Navigation)
URL: http://sv1050a/doc_link/Ja_JP/a_doc_lib/aixgen/wbinfnav/CmdsListL.htm

★ 一般的なソフトウェアの資料
(AIX 関連製品　資料の概要)
URL: http://sv1050a/doc_link/Ja_JP/a_doc_lib/aixuser/aix6kdov/gensoftdoc.htm

英数　半角

Figure 120. A Japanese Search Result

The results page lists documents that contain the term you searched for. To see one of the documents containing the term you searched for, click on the title of the document that you want to see. It is a hyperlink to that document. For example, Figure 121 on page 659 displays a page containing the term searched for in this example.


Netscape: コマンドの概要
File Edit View Go Communicator Help
Internet Lookup New&Cool
[前へ | 次へ | 目次 | 用語集(英語) | ホーム | 検索]

AIX バージョン 4.3 システム・ユーザーズ・ガイド：オペレーティング・システムおよびデバイス

コマンドの概要

一部のコマンドは、単に 1 つのワードをタイプするだけで入力することができます。 また、1 つのコマンドからの出力が別のコマンドの入力となるように、コマンドを結合することもできます。 これをパイピング といいます。

フラグは、コマンドの動作をより詳しく定義します。 フラグ とは、コマンド行でコマンド名と共に使用される修飾子のことで、通常は前にダッシュを付けます。

コマンドをまとめて、1 つのファイルにストアすることもできます。 これをシェル・プロシージャ、またはシェル・スクリプト といいます。 コマンドを個々に実行する代わりに、複数のコマンドが入ったこのファイルを実行します。

コマンドを入力するには、プロンプトでコマンド名をタイプし、Enter キーを押します。

```
$ CommandName
```

この項では、以下の項目について説明します。

- コマンドの構文
- 構文図の読み方
- 使用法ステートメントの読み方
- ウェブ・ベース・システム・マネージャを使用する方法
- smit コマンドを使用する方法
- コマンドまたはプログラムの位置を探す方法(whereis コマンド)
- コマンドの機能を表示する方法 (whatis コマンド)
- 以前に入力したコマンドをリストする方法 (history コマンド)
- コマンド・ヒストリを使用してコマンド繰り返す方法
- コマンド・ヒストリを使用してストリングを置き換える方法
- コマンド・ヒストリを編集する方法

Figure 121. A Japanese Book

11.8.3.2 Simplified Chinese Search

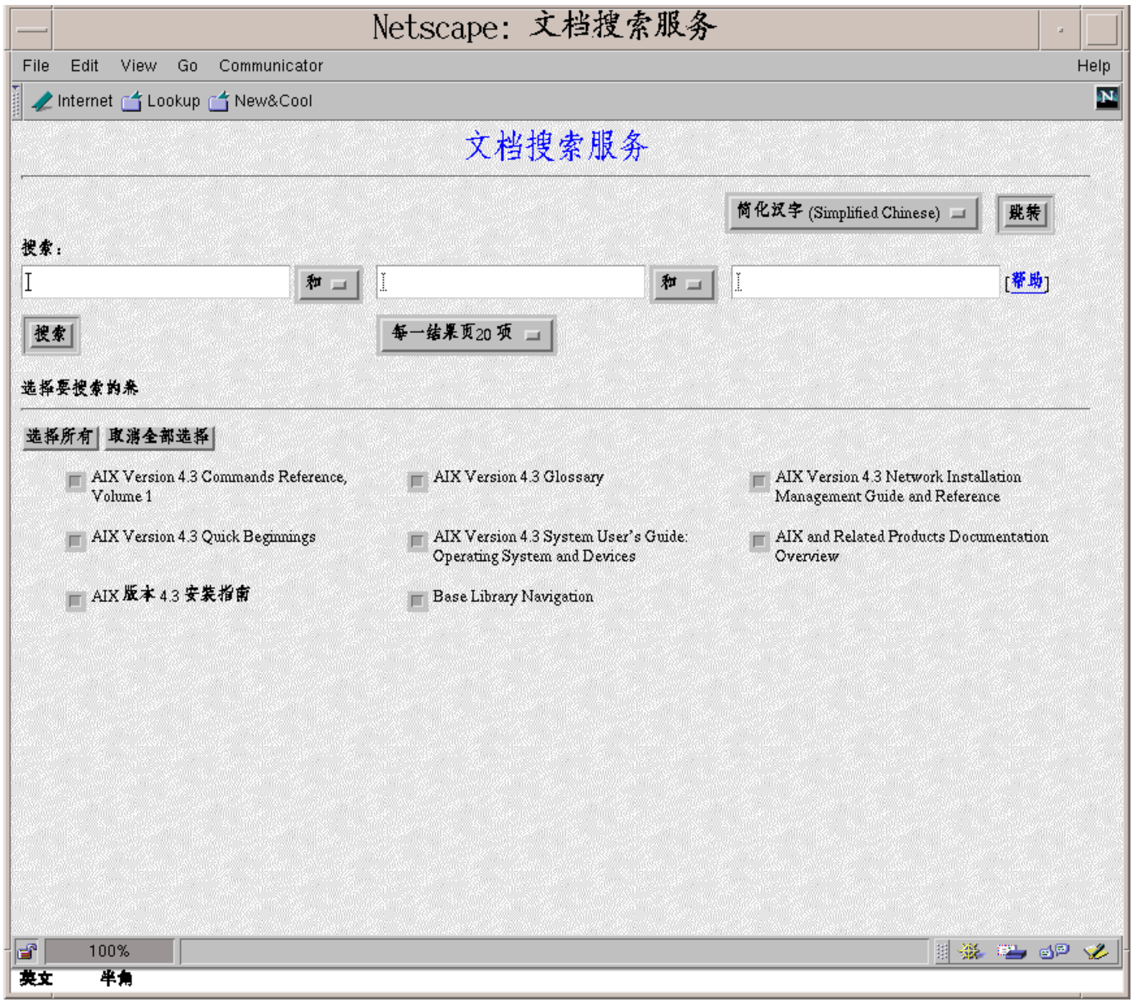
A Chinese document search can be done in a Chinese CDE environment. If you are in other language's CDE environment, you need to set the LANG to zh_CN and map the keyboard using the following command:

```
#xmodmap /usr/lpp/X11/defaults/xmodmap/zh_CN/keyboard
```

When you enter:

```
#docsearch
```

you see the screen similar to Figure 122:



文档搜索服务

简化汉字 (Simplified Chinese) 跳转

搜索：

和 和 [帮助]

搜索 每一结果页20 项

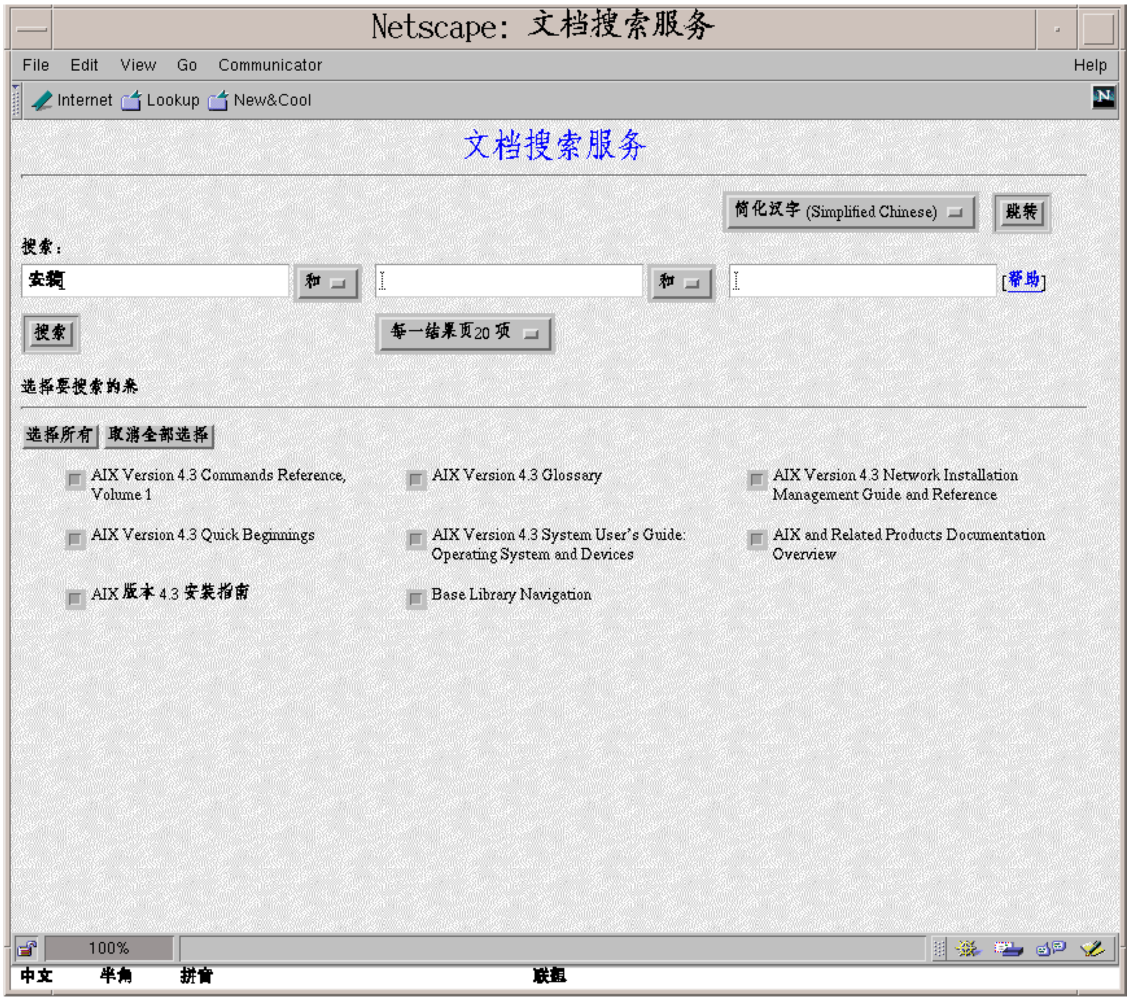
选择要搜索的书

选择所有 取消全部选择

- AIX Version 4.3 Commands Reference, Volume 1
- AIX Version 4.3 Glossary
- AIX Version 4.3 Network Installation Management Guide and Reference
- AIX Version 4.3 Quick Beginnings
- AIX Version 4.3 System User's Guide: Operating System and Devices
- AIX and Related Products Documentation Overview
- AIX 版本 4.3 安装指南
- Base Library Navigation

Figure 122. Chinese Search Form

After remapping the keyboard, using a US keyboard, you can input the Chinese words you want to search into the search field, as shown in Figure 123. Note that the entire product library is not installed for this example.



文档搜索服务

简化汉字 (Simplified Chinese)

搜索:

安装 和 和

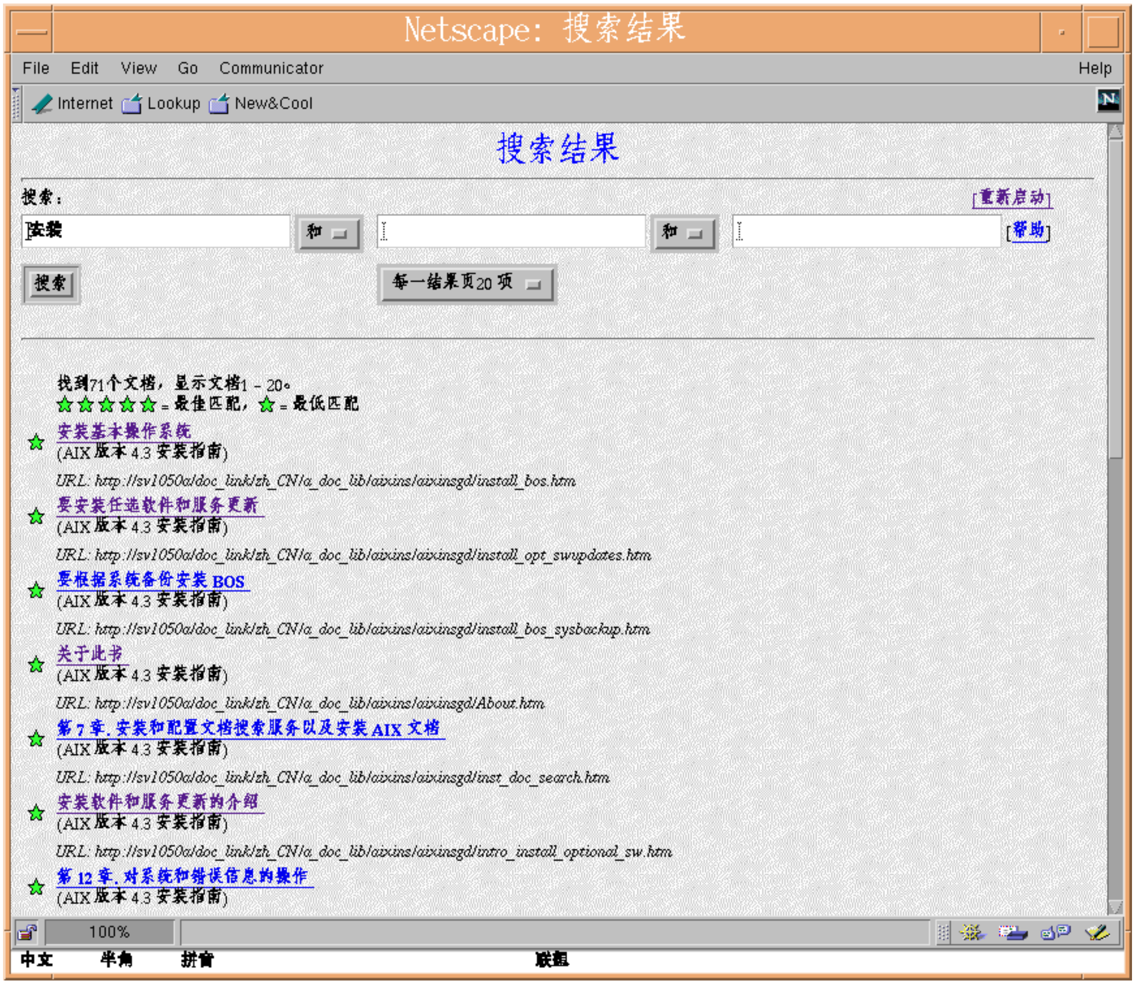
搜索 每一结果页20 项

选择要搜索的书

- AIX Version 4.3 Commands Reference, Volume 1
- AIX Version 4.3 Glossary
- AIX Version 4.3 Network Installation Management Guide and Reference
- AIX Version 4.3 Quick Beginnings
- AIX Version 4.3 System User's Guide: Operating System and Devices
- AIX and Related Products Documentation Overview
- AIX 版本 4.3 安装指南
- Base Library Navigation

Figure 123. Input Chinese Character

The searching result is similar to Figure 124 on page 662:



搜索结果

搜索: [重新启动]

安装 | 和 | | 和 | | [帮助]

找到71个文档，显示文档1 - 20。

★★★★★ = 最佳匹配，★ = 最低匹配

★ 安装基本操作系统
(AIX 版本 4.3 安装指南)

URL: http://sv1050a/doc_link/zh_CN/a_doc_lib/aixins/aixinsgd/install_bos.htm

★ 要安装任选软件和服务更新
(AIX 版本 4.3 安装指南)

URL: http://sv1050a/doc_link/zh_CN/a_doc_lib/aixins/aixinsgd/install_opt_swupdates.htm

★ 要根据系统备份安装 BOS
(AIX 版本 4.3 安装指南)

URL: http://sv1050a/doc_link/zh_CN/a_doc_lib/aixins/aixinsgd/install_bos_sysbackup.htm

★ 关于此书
(AIX 版本 4.3 安装指南)

URL: http://sv1050a/doc_link/zh_CN/a_doc_lib/aixins/aixinsgd/About.htm

★ 第 7 章. 安装和配置文档搜索服务以及安装 AIX 文档
(AIX 版本 4.3 安装指南)

URL: http://sv1050a/doc_link/zh_CN/a_doc_lib/aixins/aixinsgd/inst_doc_search.htm

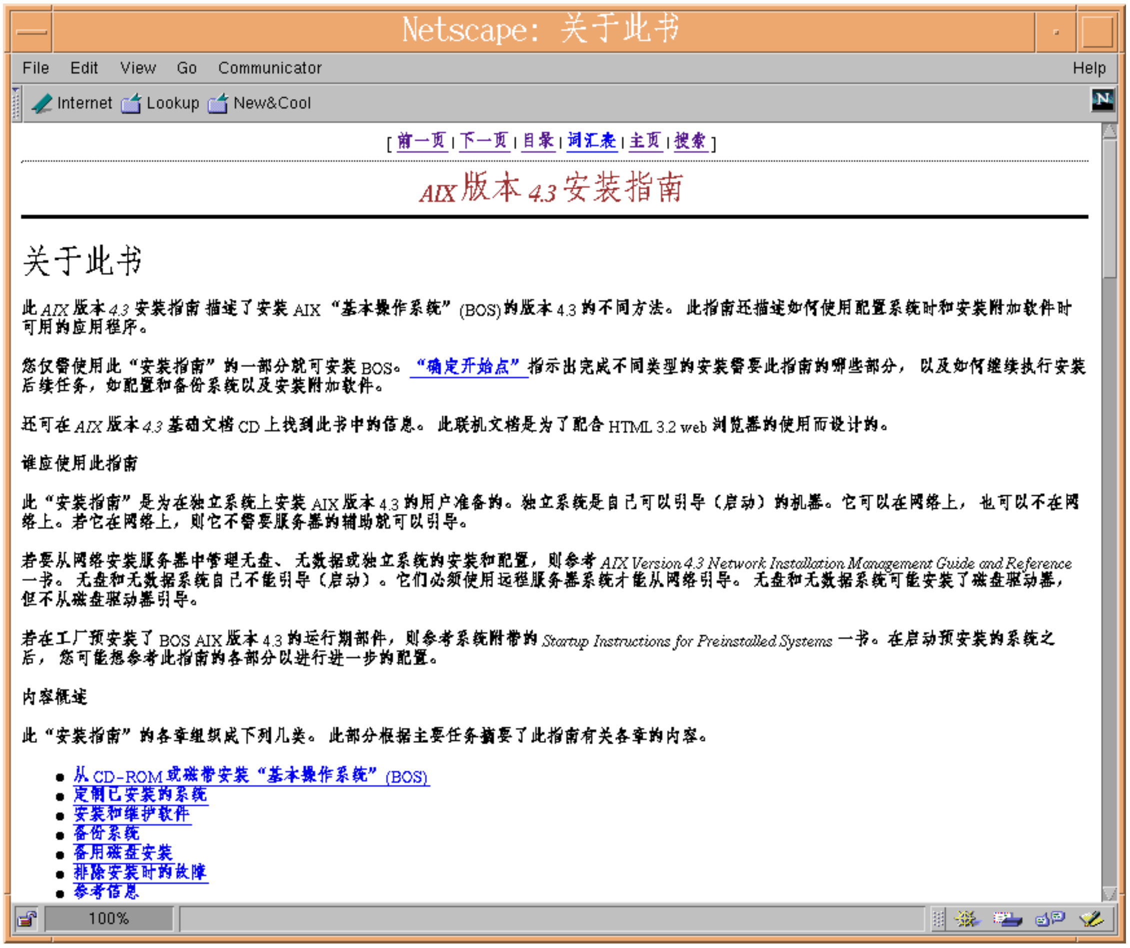
★ 安装软件和服务更新的介绍
(AIX 版本 4.3 安装指南)

URL: http://sv1050a/doc_link/zh_CN/a_doc_lib/aixins/aixinsgd/intro_install_optional_sw.htm

★ 第 12 章. 对系统和错误信息的操作
(AIX 版本 4.3 安装指南)

Figure 124. Chinese Searching Result

The results page lists documents that contain the term you searched for. To see one of the documents containing the term you searched for, click on the title of the document that you want to see. It is a hyperlink to that document. For example, Figure 125 on page 663 displays a page containing the term searched for in this example.



[前一页 | 下一页 | 目录 | 词汇表 | 主页 | 搜索]

AIX 版本 4.3 安装指南

关于此书

此 *AIX* 版本 *4.3* 安装指南 描述了安装 AIX “基本操作系统” (BOS) 的版本 4.3 的不同方法。 此指南还描述如何使用配置系统时和安装附加软件时可用的应用程序。

您仅需使用此 “安装指南” 的一部分就可安装 BOS。 “确定开始点” 指示出完成不同类型的安装需要此指南的哪些部分， 以及如何继续执行安装后续任务，如配置和备份系统以及安装附加软件。

还可在 *AIX* 版本 *4.3* 基础文档 CD 上找到此书中的信息。 此联机文档是为了配合 HTML 3.2 web 浏览器的使用而设计的。

谁应使用此指南

此 “安装指南” 是为在独立系统上安装 AIX 版本 4.3 的用户准备的。独立系统是自己可以引导（启动）的机器。它可以在网络上， 也可以不在网络上。若它在网络上，则它不需要服务器的辅助就可以引导。

若要从网络安装服务器中管理无盘、 无数据或独立系统的安装和配置，则参考 *AIX Version 4.3 Network Installation Management Guide and Reference* 一书。 无盘和无数据系统自己不能引导（启动）。它们必须使用远程服务器系统才能从网络引导。 无盘和无数据系统可能安装了磁盘驱动器，但不从磁盘驱动器引导。

若在工厂预安装了 BOS AIX 版本 4.3 的运行期部件，则参考系统附带的 *Startup Instructions for Preinstalled Systems* 一书。在启动预安装的系统之后， 您可能想参考此指南的各部分以进行进一步的配置。

内容概述

此 “安装指南” 的各章组织成下列几类。 此部分根据主要任务摘要了此指南有关各章的内容。

- 从 CD-ROM 或磁带安装 “基本操作系统” (BOS)
- 定制已安装的系统
- 安装和维护软件
- 备份系统
- 备用磁盘安装
- 排除安装时的故障
- 参考信息

Figure 125. A Chinese Book for Installation

11.8.4 Binary Compatibility

Single byte indexes created using AIX 4.3 will work correctly with the AIX 4.3.2 single byte Documentation Search Service. The Documentation Search Service should be maintained at the latest level.

Chapter 12. AIX Stand-Alone LDAP Directory Product

The AIX Stand-alone Lightweight Directory Access Protocol (LDAP) product provides client access to directory data on a server using standard Internet protocols (LDAP and HTTP). The following discussions are representative of AIX Version 4.3.0.

12.1 What is LDAP?

A directory is a listing of information about objects arranged in some order. It is suitable for storing such information as list of books, e-mail address of people, and so on. A user or a program can search the directory to locate a book or e-mail address for a specific person. It is actually a database but the differences between general purpose databases and directories are:

- Directories are read-mostly whereas databases tend to change rapidly

  Because of the nature of directories that store such static information as phone number, the contents do not change so often. On the other hand, the information stored in databases such as order quantity of an item changes rapidly.

- Data consistency requirement to directories is not strict

  Because directories store static information, they might not have transaction support for data integrity. Duplicates and out of date data is acceptable.

LDAP itself is a protocol for a client to communicate with LDAP servers that implement X.500 like directory. LDAP has its origin from directory access protocol (DAP) of X.500. Because DAP depends on OSI, it has not been adopted widely by commercial environments. LDAP has been developed as a lightweight alternative to DAP. Compared with DAP, LDAP has advantages such that it runs over TCP/IP, its function model is simpler and it uses string representation rather than ASN.1.

Basic characteristics of LDAP are described below

- Client/Server Model

  LDAP adopts the client/server model. An LDAP client performs protocol operations such as query or modify using LDAP API against an LDAP server and the server returns response after completing these operations.

- Distributed Directories

  Although this is not a characteristic of LDAP but of LDAP servers that implement directory, a directory can be distributed. A distributed directory can be partitioned or replicated. When partitioned, an LDAP client can access information not stored in local directory through use of a kind of link stored in local directory. The link points to the location of the information stored in remote directory. This link is called referral. When replicated, a client can access information stored in the nearest directory server. LDAP itself has some mechanisms for accessing distributed directories efficiently. LDAP, as its name implies, is essentially optimized for high speed access to the information and use of URL as resource locator makes a human or a program locate information it needs easier.

- Security

  LDAP incorporates a security model in its specification. The model, SASL, will be described later. On LDAP server's side, typically some kind of ACL mechanisms are implemented although they are not standardized at this time of writing. Also, some transport level security mechanisms are incorporated. Such mechanism as SSL and TLS are used to authenticate client or server or both and encrypt data transferred between client and server.

12.2 LDAP Naming Model

The LDAP naming model defines how entries are identified and organized. Although this model is just an aspect of LDAP architecture, understanding this model gives you a conceptual view of LDAP and directory. Entries are organized in a tree-like structure called the directory information tree (DIT). Entries are arranged within the DIT based on their distinguished name (DN). A DN is a unique name that unambiguously identifies a single entry. DN are made up of relative distinguished names (RDNs). Each RDN in a DN corresponds to a branch in the DIT leading from the root of the DIT to the directory entry. In other words, RDNs are nodes and leaves of the DIT. Figure 126 illustrates a basic DIT.

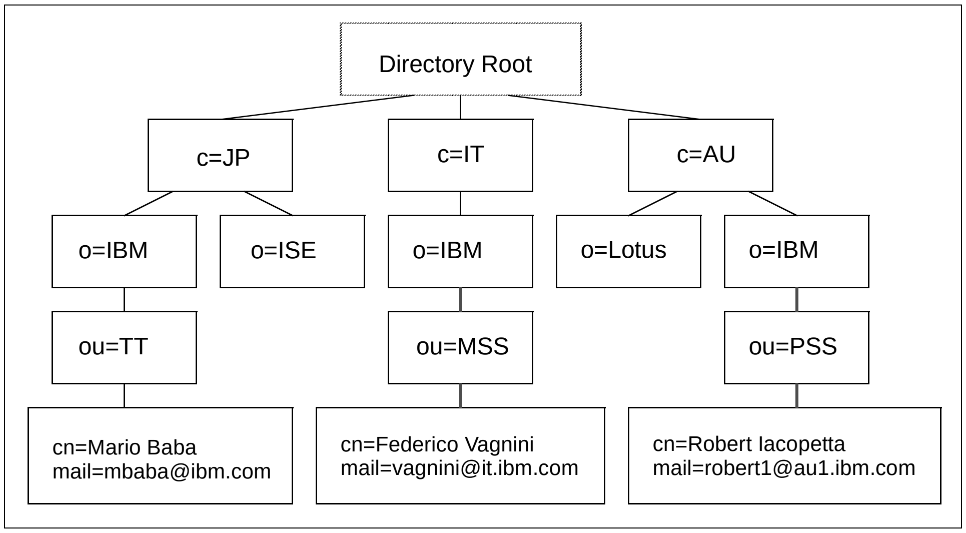


Figure 126. Directory Information Tree (DIT)

In the above figure, DN that represents Mario Baba is:

```
cn=Mario Baba, ou=ITSO, o=IBM, c=JP
```

Each RDN in a DN is read from leaf to root separated by comma. The words, cn, ou, o, and c are predefined attributes of RDN and each represents CommonName, OrganizationUnit, Orgazantion and Country.

Each RDN has one or more attributes. In the figure, RDN cn=Mario Baba has attributes cn=Mario Baba and mail=mbaba@ibm.com. You can also specify an RDN by concatenating its attributes separated by "+". For example, RDN for Mario Baba can be specified as

```
cn=Mario Baba+mail=mbaba@ibm.com
```

Also, the following DN is valid.

```
cn=Mario Baba+mail=mbaba@ibm.com, ou=ITSO, o=IBM, c=JP
```

12.3 Typical Configurations

Figure 127 shows a typical client/server network with examples of many of the capabilities provided:

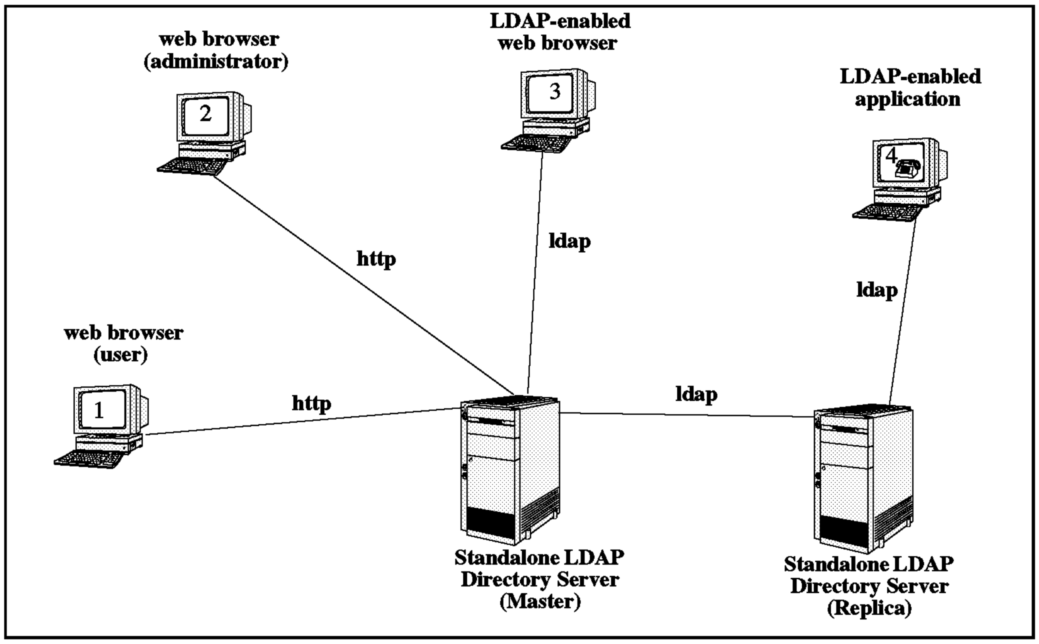

web browser (administrator)
2
LDAP-enabled web browser
3
LDAP-enabled application
4
http
ldap
ldap
web browser (user)
1
http
ldap
Standalone LDAP Directory Server (Master)
Standalone LDAP Directory Server (Replica)

Figure 127. Typical AIX Stand-Alone LDAP Client/Server Configuration

The components of the figure are as follows:

Client 1

This represents an end-user of the directory working with a generic Web browser that has no built-in LDAP support. The user may connect to the server using standard HTTP and supply the URL of the HTTP gateway provided with the product. The user is then presented with a form to fill in to specify the desired search parameters. The HTTP gateway performs the search on behalf of the client and returns the matching entries from the directory to the Web browser.

Client 2

At this client, a directory administrator uses a Web browser to monitor or configure the directory. The Web browser requires no specific LDAP capabilities. In this case, the administrator connects through an install-time-selected port to a Web server located on the system containing the directory server. The HTML panels of the administration interface are presented to the administrator through the Web browser and guides them through viewing or setting configuration options for the directory.

Client 3
This represents an end-user who has a Web browser that is LDAP-enabled. The protocol flow from this client to the server is LDAP; therefore, it requires no protocol conversion on the LDAP server system.

Client 4
This client is running an application that is LDAP-enabled. The application may have been built using the LDAP Client Toolkit provided with this product, or it may have been built using an LDAP client library from another source. Note that the client is shown connecting to the Replica server but is able to search the directory by connecting to either the Master or Replica.

Replica directory server
This server is shown with the AIX Stand-alone LDAP Directory product installed. However, because the replication is achieved using a standard LDAP connection, the replica server could be any LDAP server that supports Version 2 of the LDAP. Replicas are read-only. A given master directory server may have multiple replicas configured.

Master directory server
This system runs the server software from the AIX Stand-alone LDAP Directory product.

12.4 LDAP Protocol Support

AIX Stand-alone LDAP supports Version 3 of the LDAP protocol. There is currently no LDAP Version 3 RFC that is approved. This product has been developed using Internet Draft "Lightweight Directory Access Protocol (v3) <draft-ietf-asid-ldapv3-protocol-04.txt>", that replaces LDAP (v2) RFC 1777.

Both V2 and V3 LDAP clients and servers are supported. The following combinations of clients and servers have been tested:

- AIX Stand-alone LDAP version 3 client with Netscape version 2 server
- A University of Michigan version 2 client with AIX Stand-alone LDAP server
- Netscape Version 2 client with AIX Stand-alone LDAP server

12.5 LDAP Client Toolkit

The LDAP client is represented as a toolkit that provides two types of LDAP interface. The first is a set of C APIs with associated header files and shared libraries. The second is a set of Java classes providing function equivalent to the C interface. The toolkit includes `man` pages describing the LDAP programming interface.

The LDAP client toolkit includes the following components:

- LDAP APIs (for Version 3, based on the Internet Draft "The LDAP Application Program Interface", <draft-howes-ldap-api-00.txt>)
- LDAP command line utilities
  - `ldapsearch`
  - `ldapadd`
  - `ldapmodify`
  - `ldapdelete`
  - `ldapmodrdn`
- C header files
- LDAP client Java classes
- Sample directory data
- Sample applications using LDAP

The application is linked with the shared library that exports the APIs and contains the routines behind the APIs that handle the client-side protocol.

12.6 Stand-Alone LDAP Directory Server

The key distinguishing feature between the AIX Stand-alone LDAP server implementation and other LDAP server implementations is the use of DB2 as the back-end data store. See Figure 128 for the major components included in the server package. The numbers along the top of the diagram refer back to the client types represented in Figure 127 as examples of the sort of clients that generate HTTP or LDAP flows to the server.

The features included in the AIX Stand-alone LDAP product are:

- DB2 back-end
- ODBC Driver Manager and DB2 driver
- RDB Glue
- SLAPD
- Server replication

- Administration utilities
- Administration GUI
- HTTP gateway

Note: The references to Oracle and Oracle driver in Figure 128 are merely an example of future possibilities. DB2 is the only supported database in this release.

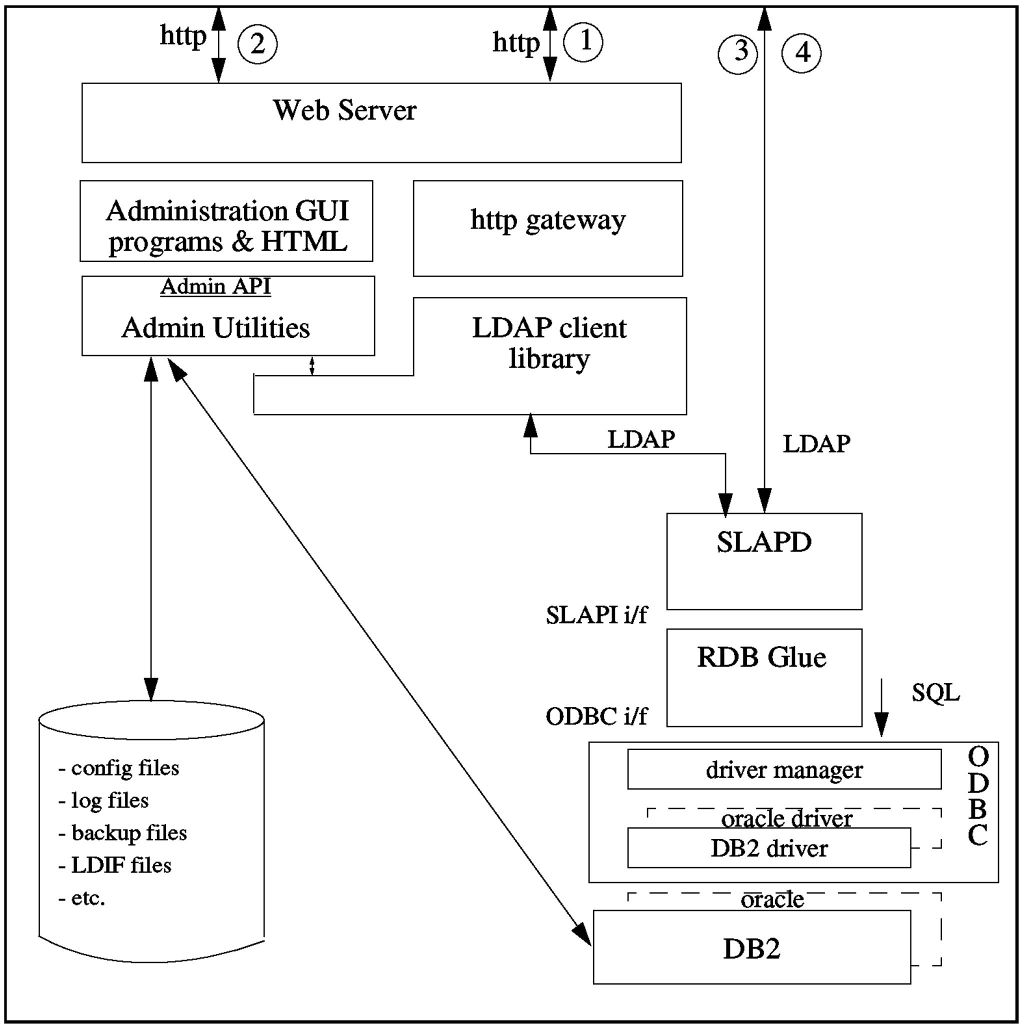


Figure 128. Stand-Alone LDAP Directory Server - Details

12.6.1 DB2 Back End

DB2 provides a robust, scalable, industry-tested basis for storage of the directory data. It includes support of Binary Large Objects (BLOBs) that facilitates use of the directory as an efficient object store. The Open DataBase Connectivity (ODBC) interface is used to connect DB2 as the directory back end. Using ODBC as the interface allows for the future inclusion of other relational databases as back-ends.

12.6.2 ODBC

This includes ODBC Driver Manager and the DB2 (ODBC) Driver. The ODBC Driver Manager provides the ODBC API to the LDAP directory server. The DB2 driver plugs into the ODBC framework and connects with the DB2 interfaces. Both of these components of ODBC are provided with the single-user version of DB2 that is shipped with the AIX Stand-alone LDAP Directory product.

12.6.3 RDB Glue

The Relational DataBase (RDB) Glue code ties together two architected interfaces to provide the data store for the directory. The SLAPD component handles incoming LDAP requests and generates calls to a set of APIs defined as the SLAPI interface. The RDB Glue provides a matching set of routines that plug into this API, take the previously mentioned API calls and generate SQL statements in the form required by the ODBC interface to read or write information to DB2.

12.6.4 SLAPD

SLAPD is the portion of the directory server that understands LDAP. It is a multithreaded daemon that receives client requests, works with the DB2 back-end to process them, and returns the results.

12.6.5 Server Replication

SLAPD process threads also monitor the replication log file and pass the corresponding update requests on to the replica server(s).

No shutdown of LDAP server is necessary to copy directory data to initialize a replica server. The process of setting up a replica server involves the following steps:

- Put the master directory into read-only mode.
- Create a backup of the directory contents.
- Use the backup file to populate the replica directory.

- Update the master directory configuration with all the information about the replica server.
- Dynamically reconfigure the master directory server (SLAPD) to pick up the new configuration.
- Take the master out of read-only mode (back to read-write).

12.6.6 HTTP Access to Directory

An HTTP gateway is provided to allow web browsers that are not LDAP-enabled to do searches on the directory. The gateway is a cgi-bin program that presents a form to the user, through the browser, to gather the parameters for the search (such as a search base, scope, search filter, and so on). Once the search information has been passed to the gateway program, it acts as an LDAP client, generating the requests to do the search, then receiving the results and passing them back to the browser for display to the end-user.

12.7 Security

The AIX Stand-alone LDAP client and server implementation supports SSL (Version 2.0 or higher), an emerging standard for World Wide Web security. SSL provides encryption of data and transport of X.509v3 public-key certificates and revocation lists. The server may be configured to run with or without the SSL support. When the server is configured to support SSL (accepting connections over a secure port - defaults to 636), it still accepts connections from clients that choose not to use SSL (these clients still specify the standard unsecure port - defaults to 389). The LDAP either flows directly over TCP/IP (using the standard sockets interfaces) or over the SSL.

12.7.1 Authentication

The following authentication options are supported:

- No authentication
- Simple authentication (password)
- X.509v3 public-key certificate at the SSL

Note: Kerberos authentication is not supported.

12.8 Installation

The Stand-alone LDAP Directory is a component of the Base Operating System for AIX 4.3; therefore, installation is done using standard installp facilities. In this release, there are no other BOS components that have dependencies on LDAP so users have the option of installing LDAP or not, depending on their requirements.

12.8.1 Software Prerequisites

The web-based interface for administration of the AIX Stand-alone LDAP Directory server requires a web server to be present, located with the directory server. Only the Netscape Fasttrack Web server has been tested, but any other web server that supports HTML 3.0/3.2 should work.

DB2 Single-User (Version 3) is automatically shipped with the directory server. DB2 is considered an install prereq. on the server system. If a supported version of DB2 has been previously installed, the prereq. requirement will be satisfied. Otherwise, the Single-User version is installed with the directory server (as a separate instance of DB2). The supported versions of DB2 are:

- DB2 Version 2.1.1 and above when packaged with any of the following products:
  - DB2 Single-User
  - DB2 Common Server
  - DB2 Parallel Edition
  - DB2 Universal Database
  - IBM Database Server

12.9 Administrative Interface

Most of the administration can be performed through a graphical user interface accessed through a web browser. The features of the interface and additional command line utilities are described below.

12.9.1 Web-Based Graphical User Interface

A Web-based graphical user interface has been provided to ease the task of administration of the AIX Stand-alone LDAP Directory server. An administrator is able to use a Web browser to do initial set-up of the directory, change configuration options, and manage the day-to-day operation of the server. HTML panels are presented through the browser to guide the administrator in viewing or changing the following:

- General server settings
- General database/back end settings
- Performance tuning options

- Directory server activity/performance data
- Directory replication configuration

and management of the following:

- Start up and shutdown of the directory server
- Access control lists
- Group membership
- Security measures (encryption options, server certificate management)
- Database creation, backup, and restore

12.9.2 Command Line Utilities

There are two command line utilities provided:

```
LDIF2DB
DB2LDIF
```

The `LDIF2DB` command line utility creates a directory database in DB2 based on the directory entries specified in LDAP Directory Interchange Format (LDIF) format in an input file.

The `DB2LDIF` command line utility creates a standard LDIF format file containing all the data from the specified DB2 directory database.

A set of command line utilities to perform LDAP operations has been included as a sample. These utilities are:

- ldapsearch
- ldapadd
- ldapmodify
- ldapdelete
- ldapmoddn
- ldapcompare

12.9.3 Other Administrative Procedures

Definition of the Directory Schema (object classes, attribute types, and directory structure) is managed through ASCII configuration files. A default schema definition is shipped in the configuration files with the product. To add to the defaults, the administrator must edit these configuration files. In this initial release, the administrator is allowed to add object class definitions, attribute type definitions, or directory structure rules.

Changes to schema definitions previously in effect for a database are not allowed.

12.10 LDAP-Related RFCs and Internet Drafts Implemented

The following lists contain Internet drafts and RFCs for LDAP and X.500 implemented in the AIX Stand alone LDAP Directory product.

12.10.1 Internet Drafts

Internet drafts may be viewed at:
http://www.internic.net/internet-drafts

- Protocol Definition (March 25, 1997)
  - <draft-ietf-asid-ldapv3-protocol-04.txt>
- Standard and Pilot Attribute Definitions (March 1997)
  - <draft-ietf-asid-ldapv3-attributes-04.txt>
- A String Representation of LDAP Search Filters (March 1997)
  - <draft-ietf-asid-ldapv3-filter-00.txt> (obsolete)
  - Replaced by <draft-ietf-asid-ldapv3-filter-02.txt> (May 1997)
- Extensions for Dynamic Directory Services (March 25, 1996)
  - <draft-ietf-asid-ldapv3ext-03.txt> (obsolete)
  - Replaced by <draft-ietf-asid-ldapv3ext-04.txt> (May 1997)
- A UTF-8 String Representation of Distinguished Names (March 1997)
  - <draft-ietf-asid-ldapv3-dn-02.txt> (obsolete)
  - Replaced by <draft-ietf-asid-ldapv3-dn-03.txt> (April 1997)
- The LDAP Application Program Interface (October 1996)
  - <draft-howes-ldap-api-00.txt>
- Use of Language Codes in LDAP V3 (March 1997)
  - <draft-ietf-asid-ldapv3-lang-01.txt>
- Definition of an Object Class to Hold LDAP Change (March 25 1997)
  - <draft-ietf-asid-changelog-00.txt>
- LDAP Multi-master Replication Protocol (March 20, 1997)
  - <draft-ietf-asid-ldap-mult-mast-rep-00.txt>
- The LDAP Data Interchange Format(LDIF) (Nov 25, 1996)(March 24 1997)

• <draft-ietf-asid-ldif-00.txt>

12.10.2 LDAP-Related RFCs

For more information on LDAP and its associated components, refer to the RFCs shown in Table 68.

Table 68. LDAP-Related RFCs

RFC Number	RFC Title
1558	A String Representation of LDAP Search Filters
1738	Uniform Resource Locators
1777	Lightweight Directory Access Protocol
1778	The String Representation of Standard Attribute Syntaxes
1779	A String Representation of Distinguished Names
1798	Connectionless LDAP
1823	The LDAP Application Program Interface
1959	An LDAP URL Format
1960	String format of LDAP search filter

12.10.3 X.500-Related RFCs

For information on X.500, refer to the RFCs shown in Table 69. They may be useful as a source of background information for the LDAP RFCs.

Table 69. X.500-Related RFCs

RFC Number	RFC Title
1274	The COSINE and Internet X.500 Schema
1275	Replication Requirements to Provide an Internet Directory Using X.500
1276	Replication and Distributed Operations Extensions to Provide an Internet Directory Using X.500
1279	X.500 and Domains
1308	Executive Introduction to Directory Services Using the X.500 Protocol
1309	Technical Overview of Directory Services Using the X.500 Protocol
1484	Using the OSI Directory to Achieve User Friendly Naming

RFC Number	RFC Title
1485	A String Representation of Distinguished Names
1487	X.500 Lightweight Directory Access Protocol
1488	The X.500 String Representation of Standard Attribute Syntaxes
1491	A Survey of Advanced Uses of X.500
1558	A String Representation of LDAP Search Filters
1564	DSA Metrics
1608	Representing IP Information in the X.500 Directory
1609	Charting Networks in the X.500 Directory
1617	Naming and Structuring Guidelines for X.500 Directory Pilots
1684	Introduction to White Pages Services based on X.500

12.11 LDAP Enhancements (4.3.3)

A new release of IBM eNetwork LDAP Directory is included in AIX 4.3.3. LDAP Version 3.1 has enhances the Version 2.1 product and is used as data store for some new features in AIX 4.3.3. Such features, including QoS policy and user management, and their usage of LDAP are described in the Networking and System Administration sections of this publication.

The following is a list of features that are newly included in LDAP Version 3.1:

- NLS Data Support

  UTF-8, a from of Unicode UCS-2, are supported on both client and server side.

- LDAP V3 Schema

  Schemas define the type of objects that can be stores in the directory. Schemas also list the attributes of each object type and whether these attributes are required or optional. In this release of LDAP following functions are newly supported.

  - Dynamic Extensible Schema

  - Schema Subclassing

  - V2 to V3 Schema Migration

  - RootDSE (Server Specific Information)

- Directory-Enabled Networks (DEN) Schema support

• LDAP V3 Protocol Enhancements

The protocol support has been enhanced according to RFC 2251. The enhancements include:

- V3 Referrals

- Controls

- Bind Sequencing

- Unsolicited Notification

- Extended Operation Plug-in Support

• Simple Authentication and Security Layer (SASL) Support

SASL is a general authentication framework that supports several different authentication methods. In this release, CRAM-MD5 and SSL or TLS are supported.

• Client Certificate-Based Authentication

SSL client certificate authentications is supported.

• Client and Server Plug-in Support

The LDAP client allows users to add SASL plug-in functions to add new authentication mechanisms.

The Version 3.1 server allows users to add plug-in functions that extend server functions based on plug-in APIs published by Netscape.

• Finding an LDAP Server Using DNS

By setting up SRV records in the DNS database as defined in RFC2052, an LDAP client can locate a n LDAP server using DNS. Client side APIs are provided.

• Change Log Support

Change Log as defined in <draft-good-ldap-changelog-00.txt> has been implemented. Change Log information can be obtained from rootDSE.

• One-Way Encrypted Password

When storing a userPassword attribute in the database, the value is encrypted so that it is not compromised through use of database lookups such as an SQL query. Only One-way hash encryption is supported currently.

• Common Schema Support

LDAP Version 3.1 comes with following standard directory schema:

- Schemas defined in RFC 1778, 2252, and 2256.

- Directory Enabled Network

- Common Information Model (CIM) schema defines by Desktop Management Task Force (DMTF)

- The Lightweight Internet Person Schema (LIPS) from Network Application Consortium.

• Enhanced DN Processing

DNs may consist of RDNs that consist of multiple attributes connected by a "+".

• Client Caching (JNDI Only)

LDAP Version 3.1 supports client side caching to achieve greater performance. The cache mechanism is based on Internet Draft <draft-ietf-asid-ldap-cache-01.txt>. An LDAP client can obtain the time-to-live attribute from the server to use it to determine when the entry should be discarded from client’s cache. The caching function is provided for a JNDI client only in this release.

• Full JNDI SSL Support

In LDAP V3.1, the communication between JNDI clients and server can be protected by SSL.

• Java-based GUI Administration

Java based GUI tool is provided to manage the LDAP directory information in the server. It does not provide the server administration functions.

• UDB 5.2 Support

UDB 5.2 is supported as an LDAP directory information store.

12.12 LDAP Configuration (4.3.3)

In this section, the basic LDAP configuration methods are provided.

12.12.0.1 ldapxcfg

LDAP directories are configured using GUI tools as well as commands. The initial configuration can be done using `ldapxcfg` GUI tool (Figure 129). The tasks that can be done using `ldapxcfg` are:

• Define the administrator’s name and password

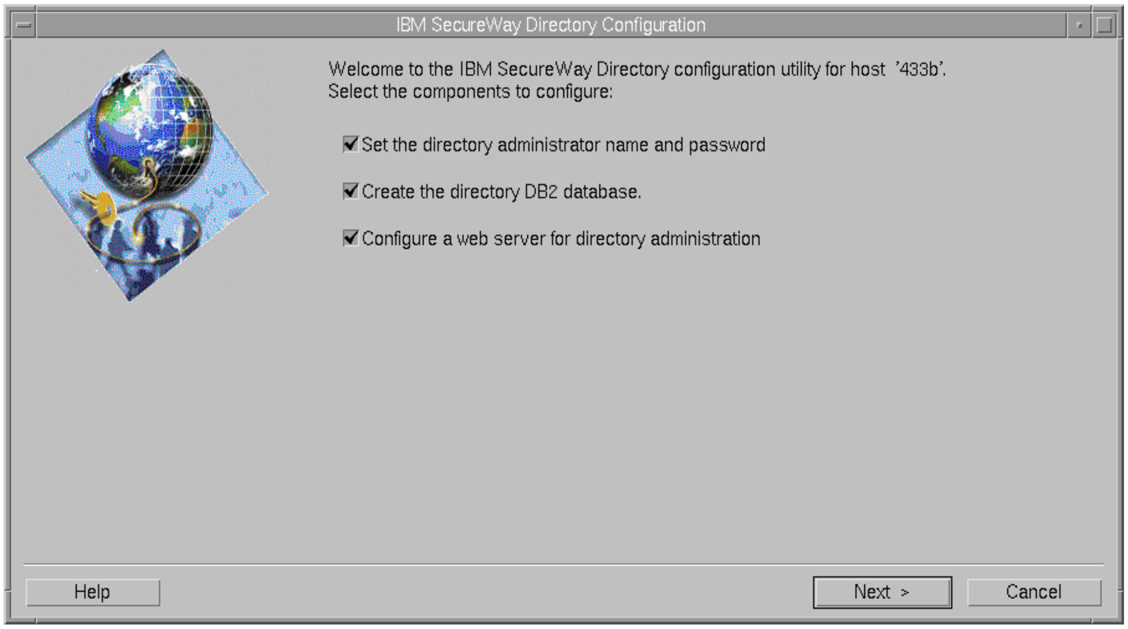
• Create the directory on DB2

You can specify an existing database and database instance or let `ldapxcfg` to create new database instance and database.

- Configure a Web server for directory administration.

  `ldapxcfg` currently supports following WEB servers

  - Lotus Domino Go WEB Server
  - Lotus Domino
  - IBM HTTPD Server
  - Apache
  - Netscape Fast Track Server
  - Netscape Enterprise Server



Welcome to the IBM SecureWay Directory configuration utility for host '433b'.
Select the components to configure:

- [✓] Set the directory administrator name and password
- [✓] Create the directory DB2 database.
- [✓] Configure a web server for directory administration

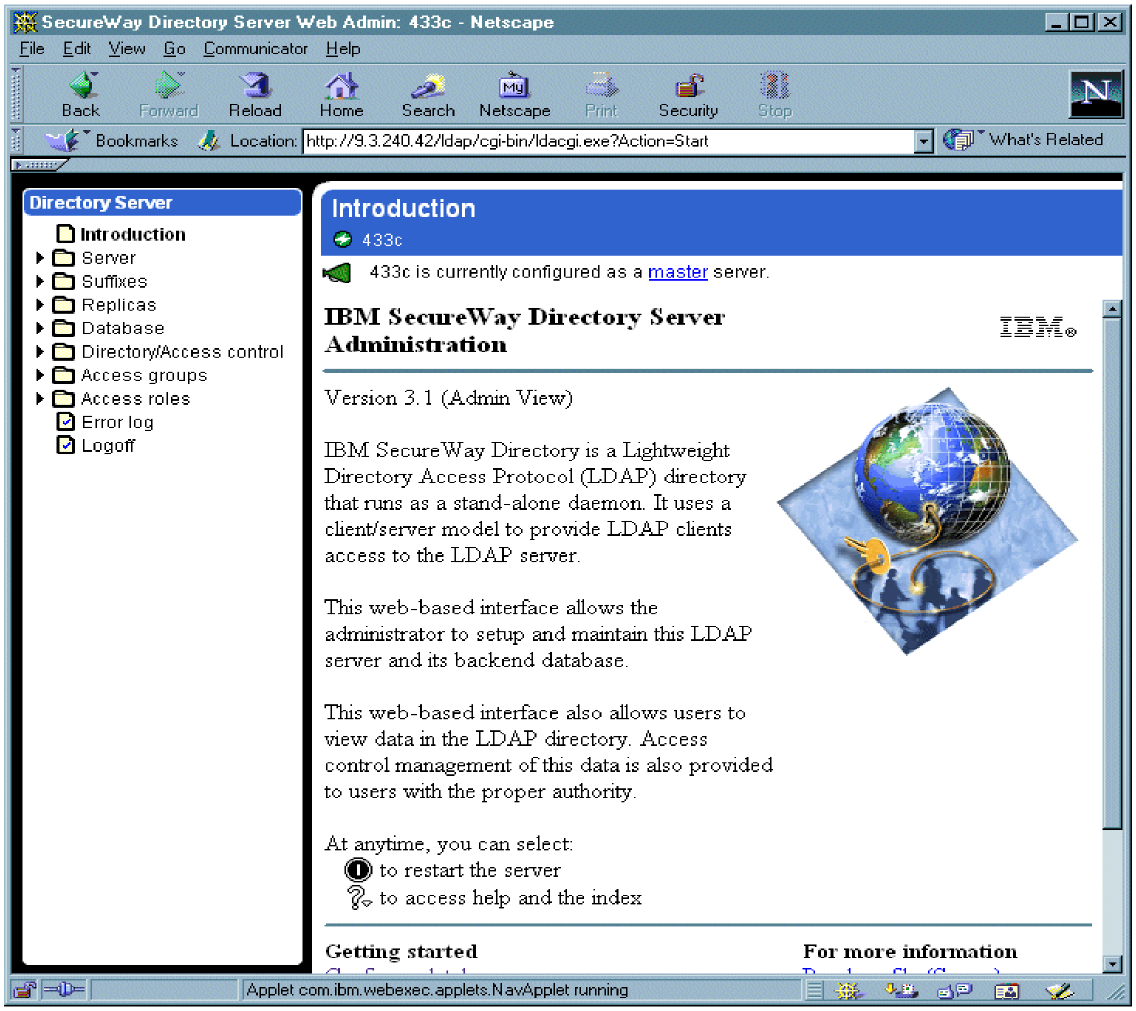
Figure 129. The ldapxcfg Dialog Panel

12.12.0.2 Web-Based Administration GUI

After initial configuration has been done, the administration from a Web browser (Figure 130) is enabled by specifying http://hostname/ldap. Restarting the Web server is necessary. The administration tasks that can be done using a Web browsers includes:

- Starting up and shutting down the directory server

- Adding and deleting suffixes
- Browsing the directory trees
- Creating and deleting replicas
- Database administration tasks


SecureWay Directory Server Web Admin: 433c - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://9.3.240.42/ldap/cgi-bin/ldacgi.exe?Action=Start
What's Related
Directory Server
Introduction
Server
Suffixes
Replicas
Database
Directory/Access control
Access groups
Access roles
Error log
Logoff
Introduction
433c
IBM
Getting started
For more information
Applet com.ibm.webexec.applets.NavApplet running

433c is currently configured as a master server.

IBM SecureWay Directory Server Administration

Version 3.1 (Admin View)

IBM SecureWay Directory is a Lightweight Directory Access Protocol (LDAP) directory that runs as a stand-alone daemon. It uses a client/server model to provide LDAP clients access to the LDAP server.

This web-based interface allows the administrator to setup and maintain this LDAP server and its backend database.

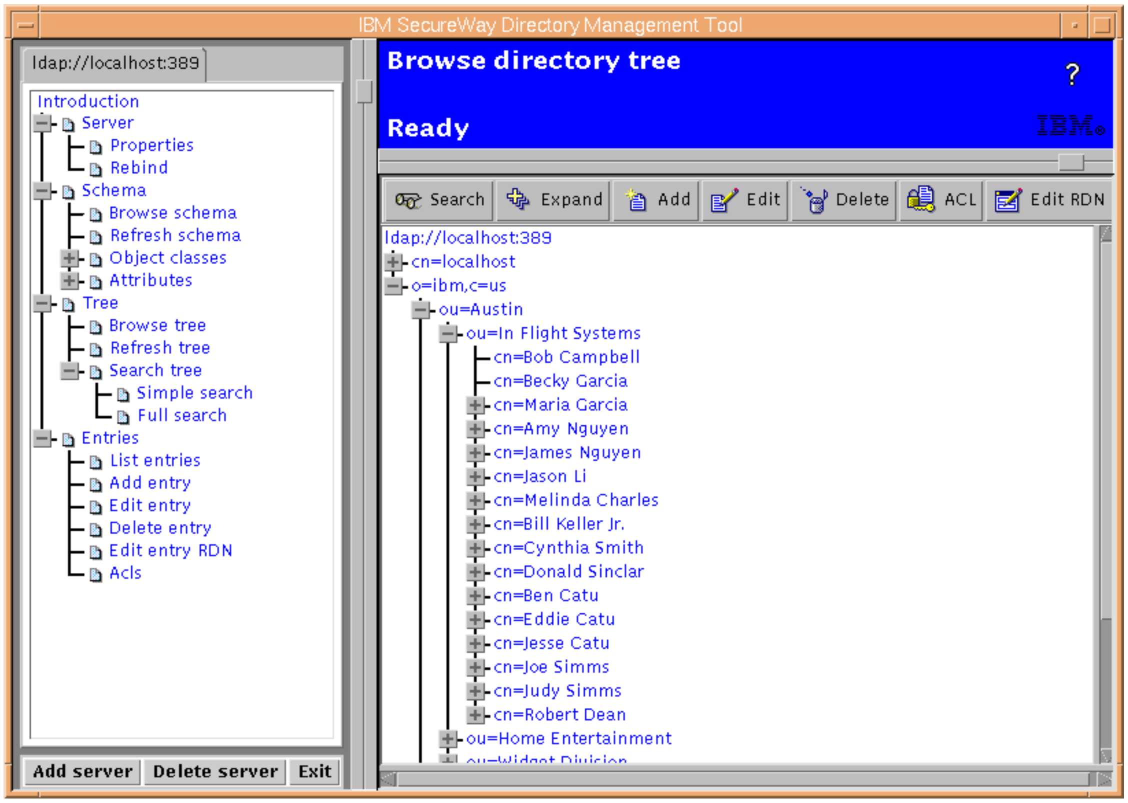
This web-based interface also allows users to view data in the LDAP directory. Access control management of this data is also provided to users with the proper authority.

At anytime, you can select:
to restart the server
to access help and the index

Figure 130. Web-Based Administration GUI

12.12.0.3 The Directory Management Tool

A Java-based GUI tool (Figure 131) that enables directory management tasks is provided with LDAP Version 3.1. By using this tool, administrators can browse, add, and update directory objects. The directory management tool can be invoked using `dmt` command:



```
ldap://localhost:389
+ cn=localhost
- o=ibm,c=us
   - ou=Austin
      - ou=In Flight Systems
           cn=Bob Campbell
           cn=Becky Garcia
         + cn=Maria Garcia
         + cn=Amy Nguyen
         + cn=James Nguyen
         + cn=Jason Li
         + cn=Melinda Charles
         + cn=Bill Keller Jr.
         + cn=Cynthia Smith
         + cn=Donald Sinclar
         + cn=Ben Catu
         + cn=Eddie Catu
         + cn=Jesse Catu
         + cn=Joe Simms
         + cn=Judy Simms
         + cn=Robert Dean
      + ou=Home Entertainment
      + ou=Widget Division
```

Figure 131. Directory Management Tool

Appendix A. SSL File Creation for LDAP

There are several methods to create SSL keys that can be used to secure the communication between an LDAP client and LDAP server. In this section we describe how to do it using the `ikmgui` tool provided by the gskrf301 fileset (shipped in the Bonus Pack) that is the prerequisite of both client and server daemons used by AIX to use LDAP for authentication with SSL connection.

The SSL key creation must be done first on the LDAP server and then on every LDAP client. If you plan to have more than one LDAP server for availability, we suggest to create one different key for each server.

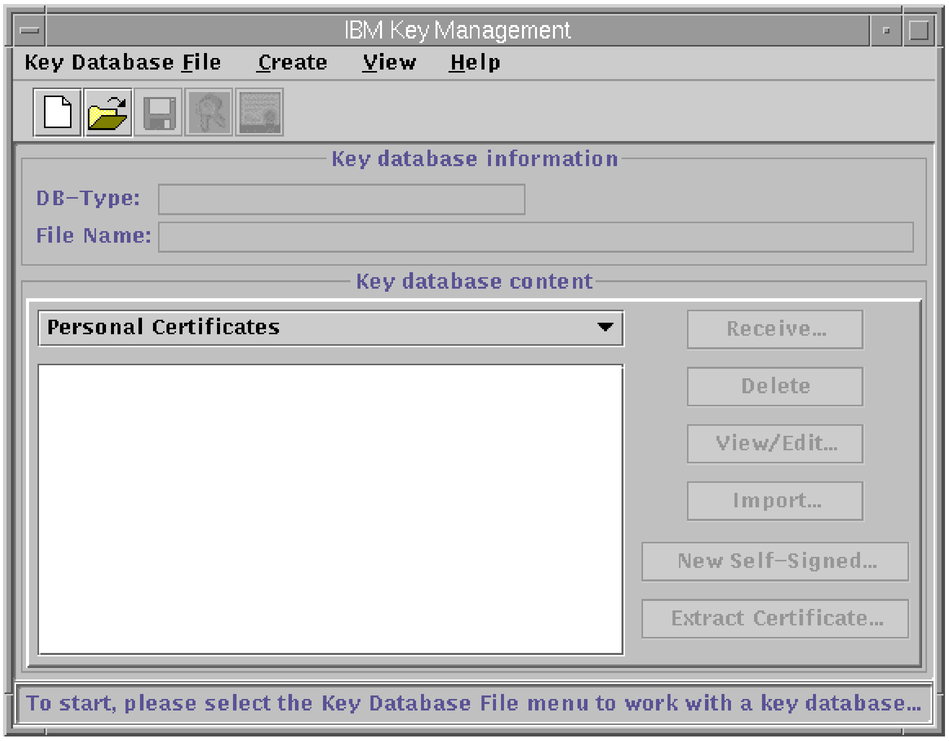
A.1 Server Key Creation

Before executing the `ikmgui` tool you have to define the JAVA_HOME environment variable. If you do not set it, the tool will ask for it. The variable defines where the Java Development Toolkit is located, and must be set to /usr/jdk_base.

If you use the Korn shell, these are the command you must type to start the tool:

```
# export JAVA_HOME=/usr/jdk_base
# ikmgui
```

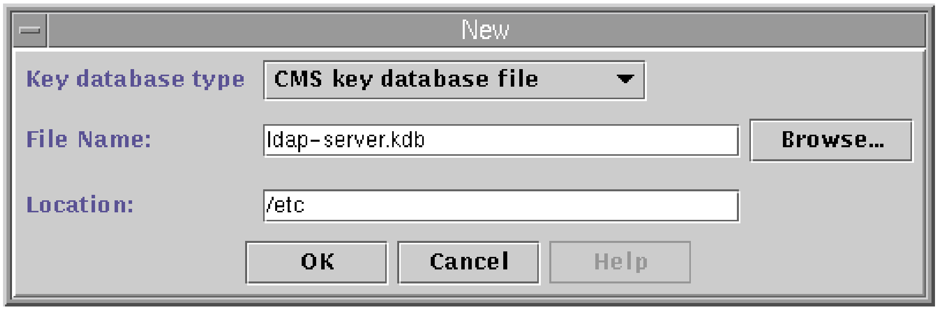
The following window (Figure 132) then appears:



To start, please select the Key Database File menu to work with a key database...

Figure 132. The ikmgui Initial Window

The first step requires you to create a new database file. You must select the **Key Database File** menu option and select **New**. You are then requested to provide the database file type, the name and location. Choose the **CMS key database** file as shown in Figure 133.

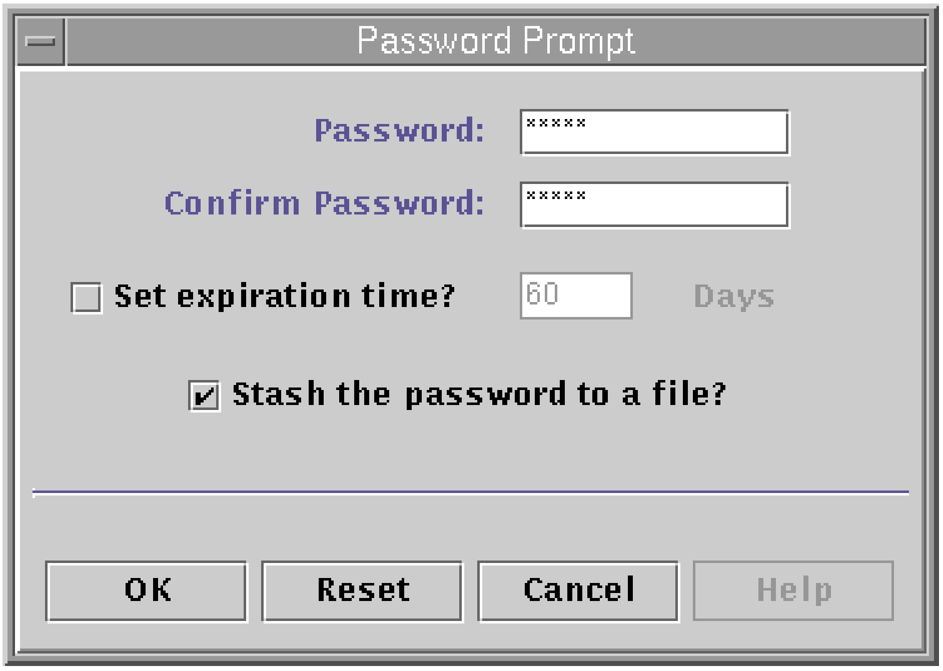


Key database type: CMS key database file
File Name: ldap-server.kdb
Location: /etc

Figure 133. The ikmgui New Database Creation Dialog

Once you press the **OK** key you are requested to define the password required to access to the database, as shown in Figure 134. You will provide that password to the `mksecldap` command when creating the LDAP server configuration.

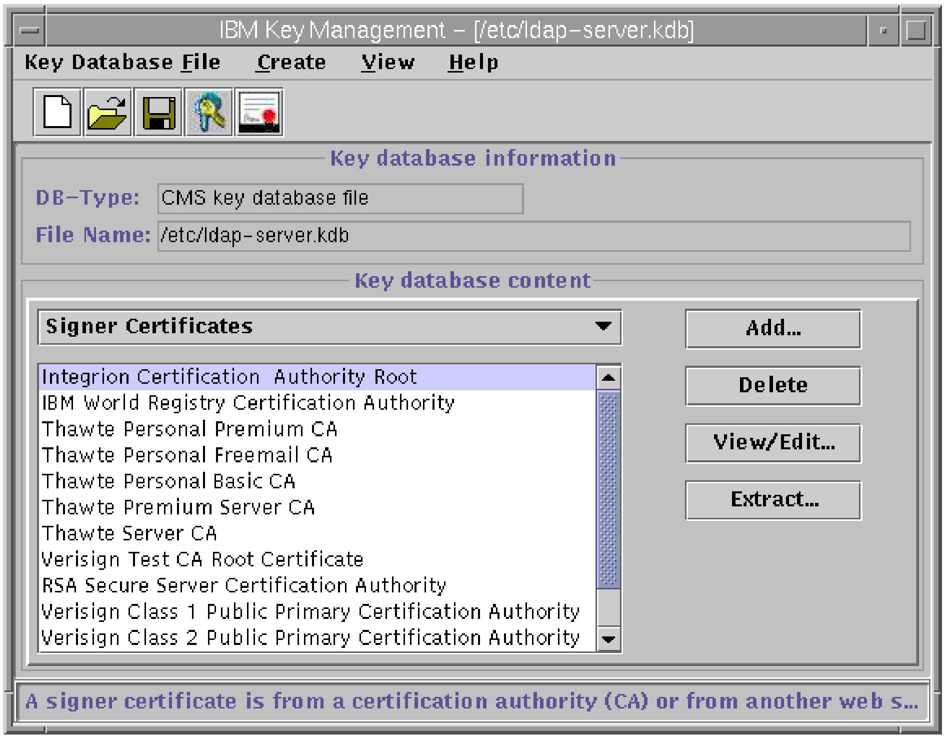
Select also the **Stash the password to a file?** since it is required by LDAP server to access the keys you are creating.



Password: *****
Confirm Password: *****
☐ Set expiration time? 60 Days
☑ Stash the password to a file?

Figure 134. The ikmgui Set Password Dialog

Now you have a key database that already recognizes a set of certificate authorities, as shown in Figure 135.



DB–Type: CMS key database file
File Name: /etc/ldap–server.kdb

Signer Certificates

Integrion Certification Authority Root
IBM World Registry Certification Authority
Thawte Personal Premium CA
Thawte Personal Freemail CA
Thawte Personal Basic CA
Thawte Premium Server CA
Thawte Server CA
Verisign Test CA Root Certificate
RSA Secure Server Certification Authority
Verisign Class 1 Public Primary Certification Authority
Verisign Class 2 Public Primary Certification Authority

A signer certificate is from a certification authority (CA) or from another web s...

Figure 135. The ikmgui Database Dialog

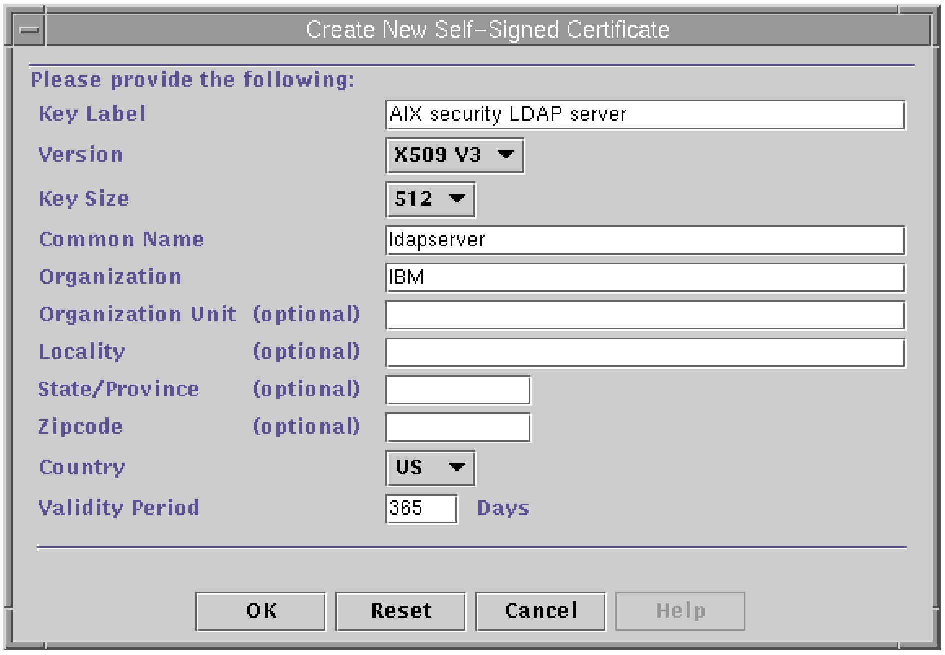
You need a certificate for the server. There are two methods:

- Request a certificate to a certificate authority
- Create a self-signed certificate

Having a certificate from a well known certifiers like those listed in Figure 135 makes it easier for you to use your certificate in different environment since it is guaranteed by the certifier. It may take time, though, to receive the certificate.

The second option is to create a self-certificate. Nobody guaranties this for the correctness of the certificates. If you are the administrator of all machines involved or the other administrators trust you, this may not be an issue. In this section we describe how to generate the self-certificate as it can be produced immediately.

Select **Create** and then **Create New Self-Signed Certificate**. A new window like the following asks you to provide the informations related to the new certificate.

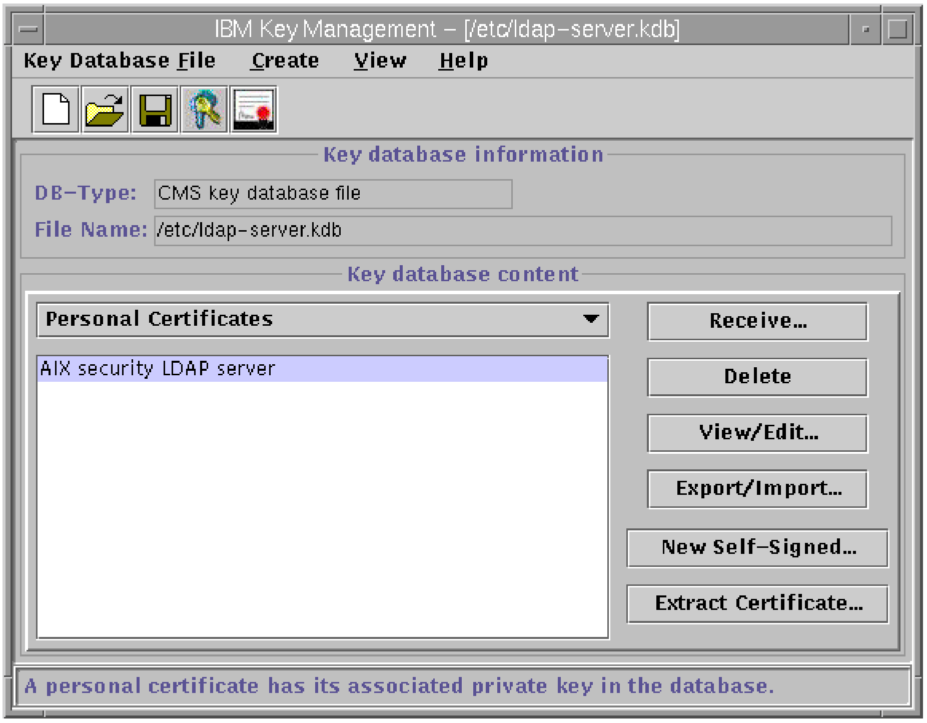


Please provide the following:

Key Label	AIX security LDAP server
Version	X509 V3
Key Size	512
Common Name	ldapserver
Organization	IBM
Organization Unit (optional)	
Locality (optional)	
State/Province (optional)	
Zipcode (optional)	
Country	US
Validity Period	365 Days

Figure 136. The ikmgui Create New Self-Certificate Dialog

When you press **OK**, the tools works for a little bit and then creates and stores in the key database the new certificate, that is shown by the tool in a window like the following:



DB–Type: CMS key database file
File Name: /etc/ldap–server.kdb

Personal Certificates

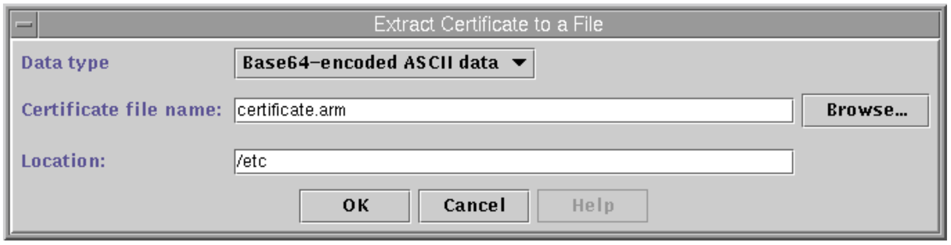
AIX security LDAP server

A personal certificate has its associated private key in the database.

Figure 137. The ikmgui with Self Certificate Dialog

You have now the certificate (/etc/ldap-server.kdb on our example) that enables you to start LDAP with SSL security. An additional step is needed to export from the database the key that is required by the clients to connect to the server.

While AIX Security LDAP Server is still selected on your window, click on **Extract Certificate...** and extract the certificate as a Base64-encoded ASCII data, as shown on Figure 138.



Data type: Base64–encoded ASCII data
Certificate file name: certificate.arm
Location: /etc

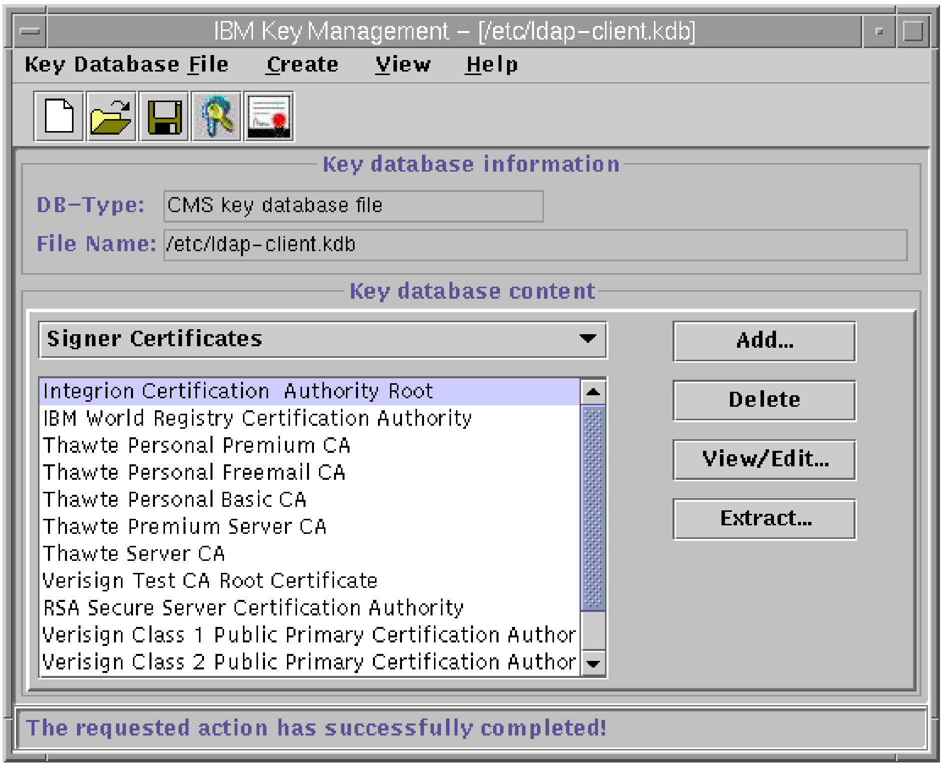
Figure 138. The ikmgui Extract Certificate Dialog

Press **OK** and copy the extracted certificate (in our case /etc/certificate.arm) on the client machines.

A.2 Client Key Creation

Once you have created the key on the server and have extracted the certificate, you need to act on the client.

As a first step, you need to create a client key database, following the same process described in Appendix A.1, “Server Key Creation” on page 685. If you select /etc/ldap-client.kdb as a file name, you end up with a window like Figure 139. Then you have to copy on the client machine the certificate file you have extracted from the server key database.



DB–Type: CMS key database file
File Name: /etc/ldap–client.kdb

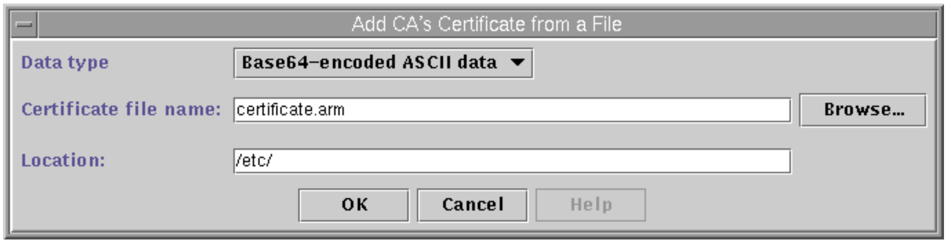
Signer Certificates

Integrion Certification Authority Root
IBM World Registry Certification Authority
Thawte Personal Premium CA
Thawte Personal Freemail CA
Thawte Personal Basic CA
Thawte Premium Server CA
Thawte Server CA
Verisign Test CA Root Certificate
RSA Secure Server Certification Authority
Verisign Class 1 Public Primary Certification Author
Verisign Class 2 Public Primary Certification Author

The requested action has successfully completed!

Figure 139. The ikmgui Client Key Database Dialog

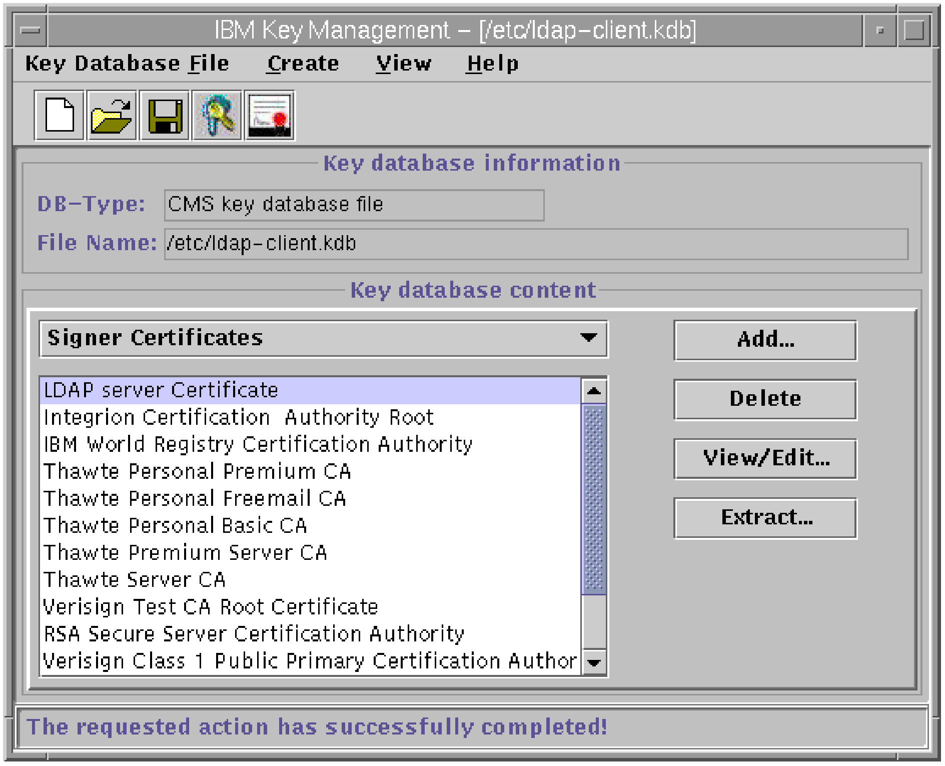
Then press the **Add..** button to add the certificate copied from the server. You have provide the file name and the Base64-encoded ASCII data type, as shown in Figure 140.



Data type: Base64-encoded ASCII data
Certificate file name: certificate.arm
Location: /etc/

Figure 140. The ikmgui Add Certificate Dialog

When you press **OK**, you are requested to provide a label to identify the new certificate, then the certificate is added to the client key database and you see a window like the following:



DB-Type: CMS key database file
File Name: /etc/ldap-client.kdb

Signer Certificates

LDAP server Certificate
Integrion Certification Authority Root
IBM World Registry Certification Authority
Thawte Personal Premium CA
Thawte Personal Freemail CA
Thawte Personal Basic CA
Thawte Premium Server CA
Thawte Server CA
Verisign Test CA Root Certificate
RSA Secure Server Certification Authority
Verisign Class 1 Public Primary Certification Author

The requested action has successfully completed!

Figure 141. The ikmgui Final Client Key Database Dialog

On this machine you now can use the key database (in our example, the file /etc/ldap-client.kdb) with the `mksecldap` command to enable the machine to ask AIX user information to the LDAP server using SSL security.

If you have more clients, you can execute the same steps on each client or copy the client key database on each client machine and reuse it for the `mksecldap` command on each of them.

Appendix B. Special Notices

This publication is intended to help AIX system administrators, developers, and support professionals understand the key technical differences between AIX Version 4.3 and previous releases. It takes AIX Version 4.2 as the basis for these differences. The information in this publication is not intended as the specification of any programming interfaces that are provided by AIX Version 4.3. See the PUBLICATIONS section of the IBM Programming Announcement for AIX Version 4.3 for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

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

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM ("vendor") products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate

Appendix C. Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

C.1 International Technical Support Organization Publications

For information on ordering these ITSO publications see “How to Get ITSO Redbooks” on page 701.

- *A Technical Introduction to PCI-Based RS/6000 Servers*, SG24-4690
- *Managing AIX V4 on PCI-Based RISC System/6000 Workstations*, SG24-2581
- *TCP/IP Tutorial and Technical Overview*, GG24-3376
- *Learning Practical TCP/IP for AIX V3.2/4.1 Users: Hints and Tips for Debugging and Tuning*, SG24-4381
- *Managing One or More AIX Systems - Overview*, GG24-4160
- *AIX/6000 X.25 LPP Cookbook*, SG24-4475
- *AIX Version 4.1 Software Problem Debugging and Reporting for the RISC System/6000*, GG24-2513
- *AIX Version 4.2 Differences Guide*, SG24-4807
- *A Holistic Approach to AIX V4.1 Migration, Planning Guide*, SG24-4651
- *RS/6000 Performance Tools In Focus*, SG24-4989
- *RS/6000 Technical and Scientific Computing: POWER3 Introduction and Tuning Guide*, SG24-5155
- *Comprehensive Guide to Virtual Private Networks, Volume III: Cross-Platform Key and Policy Management*, SG24-5309.

C.2 Redbooks on CD-ROMs

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at `http://www.redbooks.ibm.com/` for information about all the CD-ROMs offered, updates and formats.

CD-ROM Title	**Collection Kit Number**
System/390 Redbooks Collection	SK2T-2177
Networking and Systems Management Redbooks Collection	SK2T-6022

CD-ROM Title	Collection Kit Number
Transaction Processing and Data Management Redbooks Collection	SK2T-8038
Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
AS/400 Redbooks Collection	SK2T-2849
Netfinity Hardware and Software Redbooks Collection	SK2T-8046
RS/6000 Redbooks Collection (BkMgr)	SK2T-8040
RS/6000 Redbooks Collection (PDF Format)	SK2T-8043
Application Development Redbooks Collection	SK2T-8037

C.3 Other Publications

These publications are also relevant as further information sources:

- *SNA Formats*, GA27-3136
- *AIX Version 4.3 Communication Programming Concepts*, SC23-4124

C.4 Internet Sites

The following are valuable resources located on the Internet.

- `http://www.gigabit-ethernet.org`
- `http://grouper.ieee.org/groups/802/`
- `http://www.developer.ibm.com/devcon/`
- `http://www.ibm.com/servers/aix/products/ibmsw/`
- `http://www.ietf.org/html.charters/ipngwg-charter.html`
- `http://www.rs6000.ibm.com/doc_link/en_US/a_doc_lib/aixgen/`
- `http://www.lexmark.com`
- `http://www.ietf.org/html.charters/ipngwg-charter.html`
- `http://europa.eu.int/euro`

How to Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** `http://www.redbooks.ibm.com/`

  Search for, view, download, or order hardcopy/CD-ROM redbooks from the redbooks Web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this redbooks site.

  Redpieces are redbooks in progress; not all redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

- **E-mail Orders**

  Send orders by e-mail including information from the redbooks fax order form to:

	e-mail address
In United States	usib6fpl@ibmmail.com
Outside North America	Contact information is in the “How to Order” section at this site: `http://www.elink.ibmlink.ibm.com/pbl/pbl/`

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the “How to Order” section at this site: `http://www.elink.ibmlink.ibm.com/pbl/pbl/`

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	Fax phone number is in the “How to Order” section at this site: `http://www.elink.ibmlink.ibm.com/pbl/pbl/`

This information was current at the time of publication, but is continually subject to change. The latest information may be found at the redbooks Web site.

IBM Intranet for Employees

IBM employees may register for information on workshops, residencies, and redbooks by accessing the IBM Intranet Web site at `http://w3.itso.ibm.com/` and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access `MyNews` at `http://w3.ibm.com/` for redbook, residency, and workshop announcements.

IBM Redbook Fax Order Form

Please send me the following:

Title	**Order Number**	**Quantity**

First name	Last name	
Company		
Address		
City	Postal code	Country
Telephone number	Telefax number	VAT number

☐ Invoice to customer number ______________________________

☐ Credit card number ______________________________

Credit card expiration date	Card issued to	Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

List of Abbreviations

ABI	Application Binary Interface
AH	Authentication Header
ANSI	American National Standards Institute
API	Application Programming Interface
ARP	Address Resolution Protocol
ASR	Address Space Register
AUI	Attached Unit Interface
BOS	Base Operating System
BLOB	Binary Large Object
CDE	Common Desktop Environment
CDLI	Common Data Link Interface
CEC	Central Electronics Complex
CGE	Common Graphics Environment
CHRP	Common Hardware Reference Platform
CISPR	International Special Committee on Radio Interference
CLVM	Concurrent LVM
CMOS	Complimentary Metal-Oxide Semiconductor
DAD	Duplicate Address Detection
DASD	Direct Access Storage Device
DBE	Double Buffer Extension
DCE	Distributed Computing Environment
DES	Data Encryption Standard
DIT	Directory Information Tree
DMA	Direct Memory Access
DN	Distinguished Name
DSMIT	Distributed SMIT
DTE	Data Terminating Equipment
EA	Effective Address
ECC	Error Checking and Correcting
EIA	Electronic Industries Association
EMU	European Monetary Union
EOF	End of File
ESID	Effective Segment ID
ESP	Encapsulating Security Payload
FCC	Federal Communication Commission
FCAL	Fibre Channel Arbitrated Loop
FDDI	Fiber Distributed Data Interface
FIFO	First-In First-Out
FLASH EPROM	Flash Erasable Programmable Read-Only Memory
GAI	Graphic Adapter Interface
GPR	General Purpose Register

HCON	IBM AIX host connection program/6000
HFT	High Function Terminal
I/O	Input/Output
ICCCM	Inter-Client Communications Conventions Manual
ICE	Inter-Client Exchange
ICElib	Inter-Client Exchange library
ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force
IHV	Independent Hardware Vendor
IJG	Independent JPEG Group
ILS	International Language Support
IM	Input Method
IPL	Initial Program Load
IPSec	IP Security
ISA	Industry Standard Architecture
ISAKMP/Oakley	Internet Security Association Management Protocol
ISNO	Interface Specific Network Options
ISO	International Organization for Standardization
ISV	Independent Software Vendor
ITSO	International Technical Support Organization
JFS	Journaled File System
LDAP	Lightweight Directory Access Protocol
LDIF	LDAP Directory Interchange Format
LFT	Low Function Terminal
LID	Load ID
LP	Logical Partition
LPI	Lines Per Inch
LPP	Licensed Program Products
LPR/LPD	Line Printer/Line Printer Daemon
LP64	Long-Pointer 64
LTG	Logical Track Group
LVCB	Logical Volume Control Block
LVM	Logical Volume Manager
L2	Level 2
MBCS	Multi-Byte Character Support
MCA	Micro Channel Architecture
MDI	Media Dependent Interface
MP	Multiple Processor
MII	Media Independent Interface
MST	Machine State
NBC	Network Buffer Cache
ND	Neighbor Discovery
NDP	Neighbor Discovery Protocol
NFS	Network File System
NIM	Network Install Manager
NIS	Network Information System

NL	National Language
NLS	National Language Support
NVRAM	Non-Volatile Random Access Memory
OACK	Option Acknowledgment
ODBC	Open DataBase Connectivity
ODM	Object Data Manager
OEM	Original Equipment Manufacturer
ONC +	Open Network Computing
OOUI	Object-Oriented User Interface
OSF	Open Software Foundation, Inc.
PCI	Peripheral Component Interconnect
PFS	Perfect Forward Security
PEX	PHIGS extension to X
PHB	Processor Host Bridges
PHY	Physical Layer Device
PID	Process ID
PMTU	Path MTU
PPC	PowerPC
PSE	Portable Streams Environment
PTF	Program Temporary Fix
RAID	Redundant Array of Independent Disks
RAN	Remote Asynchronous Node
RDB	Relational DataBase
RDISC	ICMP Router Discovery
RDN	Relative Distinguished Name
RFC	Request for comments
RIO	Remote I/O
RPC	Remote Procedure Call
RPL	Remote Program Loader
SACK	Selective Acknowledgments
SBCS	Single-Byte Character Support
SCSI	Small Computer System Interface
SCSI-SE	SCSI-Single Ended
SDRAM	Synchronous DRAM
SHLAP	Shared Library Assistant Process
SID	Segment ID
SIT	Simple Internet Transition
SKIP	Simple Key Management for IP
SLB	Segment Lookaside Buffer
SM	Session Management
SMIT	System Management Interface Tool
SMB	Server Message Block
SMP	Symmetrical Multi-Processor
SNG	Secured Network Gateway
SP	Service Processor
SPOT	Shared Product Object Tree
SRC	System Resource Controller

SRN	Service Request Number
SSA	Serial Storage Architecture
SSL	Secure Socket Layer
STP	Shielded Twisted Pair
SVC	supervisor or system call
SYNC	synchronization
TCE	Translate Control Entry
TCP/IP	Transmission Control Protocol/Internet Protocol
UCS	Universal Coded Character Set
UIL	User Interface Language
ULS	Universal Language Support
UP	Uni-Processor
USLA	User-Space Loader Assistant
UTF	UCS Transformation Format
UTM	Uniform Transfer Model
UTP	Unshielded Twisted Pair
VFB	Virtual Frame Buffer
VGDA	Volume Group Descriptor Area
VGSA	Volume Group Status Area
VMM	Virtual Memory Manager
VP	Virtual Processor
VPD	Vital Product Data
VSM	Visual System Manager
XCOFF	Extended Common Object File Format
XIE	X Image Extension
XIM	X Input Method
XKB	X Keyboard Extension
XPM	X Pixmap

Index

ITSO Redbook Evaluation

AIX Version 4.3 Differences Guide
SG24-2014-02

Your feedback is very important to help us maintain the quality of ITSO redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at `http://www.redbooks.ibm.com/`
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to `redbook@us.ibm.com`

Which of the following best describes you?
_ Customer _ Business Partner _ Solution Developer _ IBM employee
_ None of the above

Please rate your overall satisfaction with this book using the scale:
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction __________

Please answer the following questions:

Was this redbook published in time for your needs? Yes___ No___

If no, please explain:

____

____

____

____

What other redbooks would you like to see published?

____

____

____

Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK!)

____

____

____

____

SG24-2014-02
Printed in the U.S.A.

AIX Version 4.3 Differences Guide

SG24-2014-02