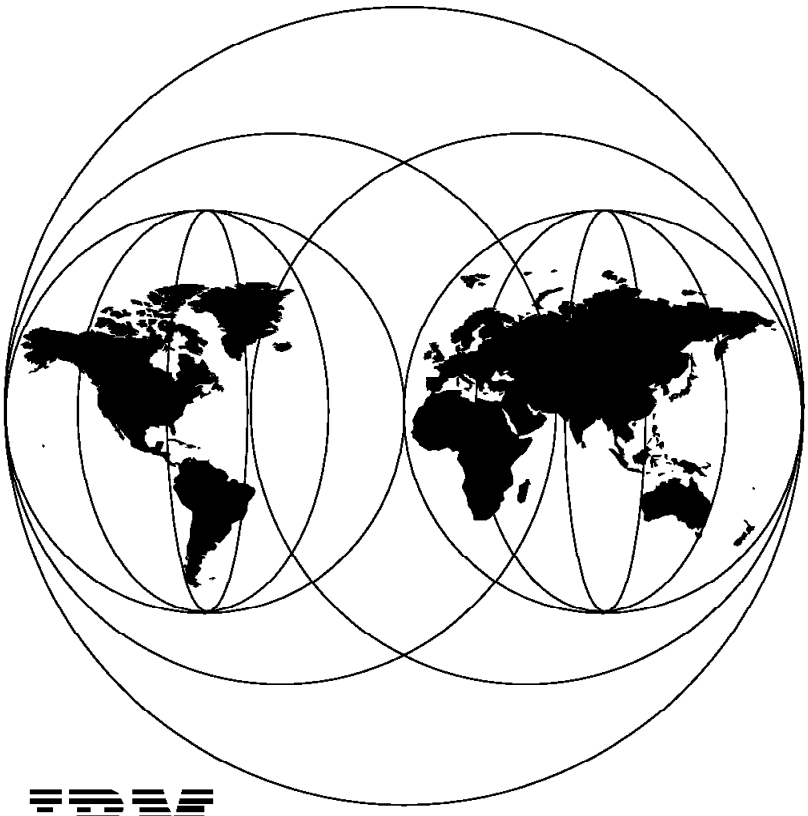
SG24-4807-00

AIX Version 4.2 Differences Guide

December 1996



IBM

International Technical Support Organization

Austin Center

International Technical Support Organization

SG24-4807-00

AIX Version 4.2 Differences Guide

December 1996

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 255.

First Edition (December 1996)

This edition applies to the initial release of AIX Version 4.2 for the RISC System/6000.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. JN9B Building 045 Internal Zip 2834
11400 Burnet Road
Austin, Texas 78758-3493

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© **Copyright International Business Machines Corporation 1996. All rights reserved.**
Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

© Copyright IBM Corp. 1996

Figures

© Copyright IBM Corp. 1996

Tables

© Copyright IBM Corp. 1996

Preface

This redbook focuses on the differences between AIX Version 4.1 and AIX Version 4.2. Its purpose is to assist systems administrators, developers, and users to understand these differences and the potential benefits to their own particular installations. This document contains important information covering changes made to the packaging of AIX offerings, including the new Entry Client Package and the Bonus Pack for AIX. Enhancements to the kernel and filesystem, specifically support for large files (greater than 2 GB) and large executables, are also covered. In addition, the many changes to standards compliance, communications, graphics, NIM, and a number of other areas are explained.

Existing AIX users intending to upgrade to Version 4.2 and new users wishing to take advantage of these features will find this redbook an excellent starting point for discussion, especially the sections that discuss binary compatibility, standards compliance, and systems management.

Some knowledge of AIX and, in particular, Version 4.1 is assumed.

How This Redbook Is Organized

This redbook contains 272 pages. It is organized as follows:

- Chapter 1, “AIX 4.2 Packaging and Installation Changes”

  This section explains the changes that have been made to the packaging of AIX in Version 4.2 including the introduction of a new Entry Client package and the Bonus Pack for AIX.

- Chapter 2, “Standards”

  This chapter begins with an introduction to X/Open UNIX and the background to the Single UNIX Specification. It explains the changes that were made to AIX in Version 4.2 to achieve complete compatibility with Spec 1170. Application developers should make themselves familiar with the contents of this section to avoid possible problems with binary compatibility of existing applications. In particular, note the use of the XPG_SUS_ENV environment variable.

- Chapter 3, “System Management”

  The Systems Management chapter comprises the largest portion of this redbook and contains important information for systems administrators and application developers who wish to understand the major enhancements that may affect their installations. It explains the changes that have been made to AIX for inclusion of large file-enabled filesystems support and large executables. Also, the limitations of some AIX commands in a large file-enabled environment and the introduction of new large file-enabled API′s are covered. Issues affecting the compatibility of older applications that must operate in a large file environment are also discussed along with some common pitfalls when porting applications to this environment. Significant enhancements to system backup support, the Logical Volume Manager, shared libraries and other system management commands are also covered in this section.

- Chapter 4, “Binary Compatibility”

© Copyright IBM Corp. 1996

IBM′s commitment to, and guarantee of, binary compatibility is explained here. Some unsupported backward compatibility tips are also offered.

- Chapter 5, “NIM Enhancements”

  Users administering a NIM environment should consult this chapter for details of additional NIM facilities and the new SMIT interface to NIM which makes setup and control of NIM much simpler and more logical.

- Chapter 6, “Graphic Enhancements”

  Two new Xserver extensions and enhancements to the Common Desktop Environment are covered here. Developers of X11 applications will find the XTEST Xserver extension particularly useful.

- Chapter 7, “Communications”

  This chapter covers additions to the communications subsystems in AIX Version 4.2 including X.25, full duplex Ethernet capability and a new CDLI interface to ATM for 3rd party applications. The section on asynchronous (tty) communications will be of interest to administrators of systems with a large number of directly attached terminals and high-speed serial I/O devices such as document scanners.

- Chapter 8, “TCP/IP Enhancements in AIX V4.2”

  Additions to authentication support in the AIX Point to Point Protocol (PPP) subsystem begin this chapter, followed by a section giving useful tips for problem determination of PPP links. The new version of Sendmail (version 8.7), Network Time Protocol (NTP) Version 3, Streams and TCP/UDP checksum disabling are also covered. Finally, the introduction of Soft5080 HostConnect, a new Licensed Program Product for AIX, is described.

- Chapter 9, “Performance Enhancements”

  The performance improvements accompanying AIX Version 4.2 are explained in this chapter. Although no user action is required to take advantage of these improvements, an understanding of the concepts and implementation may allow application designers to further optimize their code.

- Chapter 10, “License Use Management Runtime for AIX”

  Introduced with this release of AIX is a new utility for control of licensed software products. LUM is an enhanced version of iFOR/LS and Net/LS with a new user interface.

- Chapter 11, “National Language Support (NLS)”

  Following a brief introduction to the components and levels of National Language Support this chapter covers the most recent additions to AIX′s already impressive national language capabilities.

- Appendix A, “Problem Determination Update”

  This appendix is intended as an overview for administrators and support staff who need to be aware of the the new commands supplied with AIX Version 4.2 and the changes to existing commands. It also refers readers to the other chapters within this redbook that may have pertinent information for their particular problem. It can be used as a starting point for AIX V4.2 related problems.

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.

Kevin Murrell is an International Technical Support Specialist at the International Technical Support Organization, Austin Center. He is responsible for support of AIX, IBM′s implementation of *UNIX* and the RISC System/6000 platform. He has been supporting *UNIX* platforms for major manufacturers since 1984 and teaches IBM classes worldwide on all areas of AIX. Before joining the ITSO one year ago, Kevin worked in the AIX System Support Center, England as Group Leader for the Communications, Development and Graphics Support Group.

The residents who endured many weeks of intense hardship during the Texas Spring in Austin while collating the information for this book were:

Richard Cutler
IBM UK Ltd

Colin Fearnley
IBM South Africa

Roberto Gonzales
IBM Italy

Zhu Li
IBM China

Trevor Miles
IBM New Zealand

Gonzalo Quesada
GBM Costa Rica

Poh Yee Tiong
IBM Singapore

Laurent Vanel
IBM France

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

Miguel Crisanto
International Technical Support Organization, Austin Center

A special thanks to the many members of the IBM Austin AIX Development and Marketing organizations who endured our innumerable interruptions and questions without complaint. To acknowledge them all individually would make this the largest section of this redbook so, to all of you, many thanks, you *really are* appreciated.

Comments Welcome

We want our redbooks to be as helpful as possible. Should you have any comments about this or other redbooks, please send us a note at the following address:

redbook@vnet.ibm.com

Your comments are important to us!

Chapter 1. AIX 4.2 Packaging and Installation Changes

This chapter covers the changes that have been made to the packaging of AIX in Version 4.2. A new entry client package has been added, along with a bonus pack.

1.1 AIX Version 4.2 Packaging Overview

AIX Version 4.2 is available in six different packages, each designed to meet the demands of a specific operating environment. These packages are:

- AIX Version 4.2 for Entry Client
- AIX Version 4.2 for Workgroups
- AIX Version 4.2 Connections
- AIX Version 4.2 for Entry Servers, 1 to 16 Users
- AIX Version 4.2 for Entry Servers, Unlimited Users
- AIX Version 4.2 for Advanced Servers, 1 to 2 Users

The AIX Version 4.2 for Entry Client and AIX Version 4.2 for Workgroups packages are new for AIX Version 4.2. The other packages are equivalent to their AIX Version 4.1 counterparts.

A Bonus Pack for AIX Version 4.2 will be delivered with each AIX Version 4.2 Package when the media is shipped.

1.1.1 AIX 4.2 for Entry Client

AIX Version 4.2 expands the AIX family of operating system packages by introducing the AIX Version 4.2 Entry Client Package. This package provides the same reliability, scalability and industry-strength architecture that has come to be expected with AIX but with reduced functionality and at a lower cost. The package is designed to address the computing environment where there is a requirement for a lower-function end-user client package where the advanced features traditionally supplied with a UNIX-based operating system client is not needed.

The new Entry Client Package includes:

- Uniprocessor support only
- A Single user plus root login
- AIX Windows 2D development environment
- Common Desktop Environment
- DCE client support
- Support for many popular PCI, ISA, and PCMCIA adapters as provided in the larger AIX packages
- IDE disk and CD-ROM support
- SCSI disk, CD-ROM and tape support
- TCP/IP, NFS and NIS client support
- Available on CD-ROM only

© Copyright IBM Corp. 1996

The AIX Connections Option is not available with the AIX for Entry Client Package.

1.1.2 AIX Version 4.2 for Workgroups

With the introduction of the new AIX Version 4.2 for Entry Client, the previously announced AIX Version 4.1 for Clients package has been renamed in AIX Version 4.2 to AIX Version 4.2 for Workgroups. This name change has been made to better identify the package with its function as an advanced client with server capability. It addresses the computing environment where there is a requirement for additional network support, such as file serving, program serving, time and name serving, beyond that provided by the Entry Client.

The AIX Version 4.2 for Workgroups package contains the same functionality and services contained in the AIX Version 4.2 for Entry Client Package plus:

- Symmetrical multiprocessor support
- One-to-two users plus root user
- Micro Channel device support
- TCP/IP, NFS and NIS Server support
- Selected multiport device support

1.1.3 Bonus Pack for AIX

The Bonus Pack for AIX is being introduced with AIX Version 4.2 to provide a means for the delivery of popular new software products as part of the AIX Version 4.2 deliverable. The software programs provided as part of the Bonus Pack for AIX Version 4.2 include both IBM and non-IBM products.

The Bonus Pack for AIX Version 4.2 for Entry Client Package contains:

- IBM′s AIX implementation of Sun′s Java
- The Adobe Acrobat Reader, Version 2.1
- Ultimedia Services for AIX, Version 2.1.4
- Netscape Navigator, Version 2.01

The Bonus Pack for all other AIX Version 4.2 packages contains:

- IBM′s AIX implementation of Sun′s Java
- The Adobe Acrobat Reader, Version 2.1
- Ultimedia Services for AIX, Version 2.1.4
- Netscape Commerce Server, Version 1.1 (includes Netscape Navigator, Version 2.01)
- IBM Internet Connection Secure Server for AIX

The Netscape Commerce Server in the Bonus Pack will be replaced with the Netscape FastTrack Server when that product becomes available. The Netscape Navigator Version 2.01 is included with the Netscape FastTrack Server product.

1.2 Machine Independence

In AIX Version 3 and Version 4.1, the kernel contained hardware dependant code for all platforms. This made it inefficient when modifying the code in order to support new hardware platforms.

In AIX Version 4.2, the system architecture has been changed to provide a well defined hardware abstraction layer. The hardware dependant code has been extracted from the kernel and placed into new kernel extensions. These new system dependent kernel extensions provide services which isolate the device drivers and other kernel extensions from hardware changes. They also provide services to device drivers for all system planar hardware.

This design change facilitates the introduction of new PowerPC processors by minimizing the code changes required to support new machines. The new kernel extensions have been packaged in a new fileset called *devices.common.rspcbase*.

The only exception to this policy is the legacy RISC System/6000 architecture based systems, where hardware support remains in the kernel.

This change in the design of the AIX kernel has no impact on device drivers or user applications.

1.3 Compatibility Filesets

When AIX 4.1 was announced, it was stated that the compatibility filesets provided to aid migration of AIX Version 3 applications and users to AIX Version 4 would be removed in AIX Version 4.2. It is now envisaged that many customers will migrate from AIX Version 3.2 directly to AIX Version 4.2, therefore there is still a requirement for compatibility filesets. Only two unnecessary filesets have been removed:

- bos.compat.msg
- X11.compat.samples.util

It was the intention that AIX Version 4.2 would not contain the fileset *bos.compat.net*, as it contains only obsolete commands. However, removal of the fileset was not possible, as certain licensed program products have this fileset as a prerequisite for installation. This dependancy will be removed and the fileset will be deleted from the next major release.

The need for other compatibility filesets will be reviewed on an individual basis in future releases. Users with dependencies on these filesets are advised to contact their IBM Representative.

1.4 PowerDesktop

The marketing agreement which allowed IBM to provide the PowerDesktop package has been withdrawn, therefore the package is no longer supplied with AIX Version 4.2.

1.5 Fileset Repackaging

A number of changes have been made to the contents and naming of some filesets.

1.5.1.1 Common Adapter Code

The packaging of device support on AIX Version 4.1 was sometimes confusing. Code which was common to both ISA and MCA adapters and devices was packaged in such a way that ISA adapter code required installation of the MCA code as a prerequisite.

To address this problem the adapter and device support in AIX Version 4.2 has been repackaged into three types of filesets:

- devices.common.* for the code which is common to both environments.
- devices.isa.* for the code specific to the ISA bus.
- devices.mca.* for the code specific to the MCA bus.

1.5.1.2 Fileset Name Changes

The following filesets have been renamed, to better reflect their contents:

- bos.rte.up becomes bos.up
- bos.rte.mp becomes bos.mp
- bos.adt.client becomes bos.adt.lib

1.5.1.3 7318 Network Terminal Accelerator

In AIX Version 4.1, the filesets for the 7318 Network Terminal Server broke the fileset naming conventions. They were named *devices.7318.** which disguised the fact that support for the device is entirely in software. The packaging of the product has been entirely reworked for AIX Version 4.2. The new filesets are named *bos.cns.**

1.6 Installation Changes, Elapsed Time Information

The AIX Version 4.2 install process now gives a consistent indication of the elapsed time since the start of the installation. The elapsed time is displayed in two different places. The most obvious display of elapsed time is on the status screen which is visible for most of the time during installation. See Figure 1 on page 5 for an example.

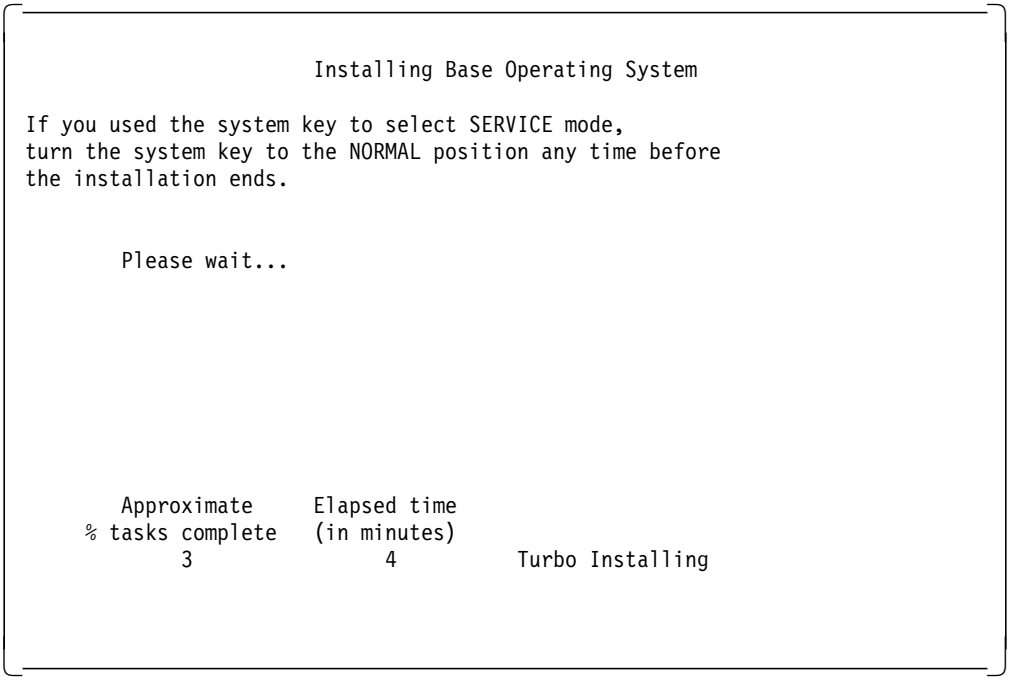


Figure 1. Install Process Elapsed Time Indication

In addition to this display of elapsed time, the `installp` command also prints elapsed time information when it is applying the base filesets during the installation process.

The `installp` command in AIX Version 4.2 has also been changed to display new status information when it is invoked as part of the base system install process. The status information gives details of the number of filesets which have been processed, and the total time taken. In AIX Version 4.1 the total time was measured from the start of `installp` processing, the time information printed is now calculated from the same starting point as the installation status screen.

The status information displayed in an AIX 4.2 installation is of the form:

```
Filesets processed: x of y
System Installation Time: M minutes          Tasks Complete P%
```

as shown in Figure 2 on page 6.

```
installp:  APPLYING software for:
        bos.txt.tfs.data 4.2.0.0
        bos.txt.spell.data 4.2.0.0
        bos.txt.bib.data 4.2.0.0

Filesets processed:  11 of 51
System Installation Time: 20 minutes        Tasks Complete: 56%

installp:  APPLYING software for:
        bos.terminfo.wyse.data 4.2.0.0
        bos.terminfo.visual.data 4.2.0.0
        bos.terminfo.tymshare.data 4.2.0.0
        bos.terminfo.ti.data 4.2.0.0
        bos.terminfo.televideo.data 4.2.0.0
        bos.terminfo.teleray.data 4.2.0.0
        bos.terminfo.tektronix.data 4.2.0.0
        bos.terminfo.sperry.data 4.2.0.0
        bos.terminfo.pci.data 4.2.0.0
        bos.terminfo.pc.data 4.2.0.0
        bos.terminfo.misc.data 4.2.0.0
        bos.terminfo.microterm.data 4.2.0.0
```

Figure 2. installp Status Information

1.7 lslpp Enhancement: lslpp -w

`lslpp -w` lists which filesets own a particular file. It allows users to specify a file name and returns a list of all files that match that name, and which fileset they belong to. Only files which are already installed on the system will be listed.

For example:

- To list the fileset that owns `installp`, enter:

```
 lslpp -w /usr/sbin/installp

File                    Fileset          Type
-----------------------------------------------------------
/usr/sbin/installp      bos.rte.install  File
```

- To list the fileset that owns all file names that contain `installp`, enter:

```
lslpp -w "*installp*"

File                                        Fileset               Type
-----------------------------------------------------------------------
/usr/sbin/installp                          bos.rte.install       File
/usr/sbin/linstallpv                        bos.rte.lvm           File
/usr/lpp/bos.sysmgt/nim/methods/c_installp
                                    bos.sysmgt.nim.client         File
/etc/linstallpv                              bos.compat.links     Symlink
```

- To display all files in the inventory database, enter:

```
lslpp -w
```

This new option is also included in SMIT panel *List Installed Software*. The SMIT interface now looks like:

```
                List Installed Software and Related Information

Move cursor to desired item and press Enter.

List Installed Software
List Applied but Not Committed Software Updates
Show Software Installation History
Show Fix (APAR) Installation Status
List Fileset Requisites
List Fileset Dependents
List Files Included in a Fileset
List Fileset Containing File




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

The *List Fileset containing File* is the new option for the `lslpp -w` command. After selecting it, you will see the following screen:

```
                   List Fileset Containing File

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

                                                [Entry Fields]
  File Name to Search For                      [all]




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

You can use `smit what_fileset` fastpath to jump straight to this screen.

Chapter 2. Standards

AIX V4.2 is designed to conform to X/Open's Single UNIX Specification (formerly known as Spec 1170) and IBM has received X/Open UNIX 95 branding for this release. To conform to the specification, a number of minor changes to the operating system were necessary, particularly in the behavior of certain system calls. The changes were made in such a way that binary compatibility with earlier releases was not compromised. These changes are described in more detail later in this chapter.

2.1 Introduction to X/Open UNIX

Founded in Europe by a group of vendors, X/Open was responsible for developing and publishing the X/Open Portability Guide (XPG) which was based on the earlier System V Interface Definition. X/Open realized the importance of formal standards and worked within the IEEE POSIX community contributing to standards creation. Later, as POSIX.1 and POSIX.2 became available, X/Open adopted them into the XPG documentation.

In 1993, IBM, Novell, Sun, HP and the OSF formed the Common Open Software Environment (COSE) project and began to develop a single specification for the UNIX programming interface. Each member participated with the understanding they would implement the interface specification within their own products. This work came to be known as Spec 1170 and in October 1994 was published as an X/Open document called The Single UNIX Specification.

The Single UNIX Specification is a collection of documents that form part of the X/Open Common Application Environment (CAE), and consist of:

- System Interface Definitions, Issue 4, Version 2 (XBD)
- System Interfaces and Headers, Issue 4, Version 2 (XSH)
- Commands and Utilities, Issue 4, Version 2 (XCU)
- Networking Services, Issue 4
- X/Open Curses, Issue 4

Note that each of the first three documents in the list has a three-letter acronym in parenthesis alongside it and together they make up the X/Open System Interface document set, sometimes referred to as XSI.

Other important events have occured in the last few years to change the nature of UNIX systems. While UNIX-derived systems have long been identified with portability, *the* UNIX operating system was a licensed product developed at AT&T Bell Labs. AT&T licensed the source code to other vendors who wished to develop UNIX operating systems, but AT&T owned the registered trademark. Within AT&T, the UNIX Software Group was responsible for the UNIX source code product and eventually the group was renamed UNIX System Laboratories (USL) and became a separate wholly-owned subsidiary of AT&T. USL was in turn purchased by Novell in 1993 and the source licensing arrangements and trademarks all fell to Novell.

© Copyright IBM Corp. 1996

Novell gained control of the UNIX source, licensing rights and trademark during the beginning of the Spec 1170 work and Novell was one of the vendors supporting this project. As the Spec 1170 work moved to X/Open for fast tracking as an X/Open specification, the sole licensing rights to the UNIX trademark were transfered to X/Open by Novell.

UNIX is no longer the operating system product from AT&T, documented by the System V Interface Definition (SVID), controlled and licensed from a single point. Neither is it a mixture of slightly different products from different vendors, each extended in slightly different ways. The UNIX specification has been separated from its licensed source-code product and *UNIX* has become a single stable specification to be used to develop portable applications that run on X/Open UNIX conforming systems.

X/Open UNIX is built from a number of components, all documented in the Single UNIX Specification which includes:

- XPG4 Internationalized System Calls and Libraries (Extended) covering POSIX.1 and POSIX.2 callable interfaces, the ISO C library and Multibyte Support Extension Addendum, the X/Open UNIX extension including STREAMS, the Shared Memory calls, application internazionalization interfaces, and a wealth of other application interfaces
- XPG4 Commands and Utilities V2, covering the POSIX.2 Shell, Utilities, and a large number of additional commands and development tools
- XPG4 C Language
- XPG4 Sockets
- XPG4 Transport Interfaces (XTI)
- XPG4 Internationalized Terminal Interfaces including the new extensions to support color and multibyte characters

The Single UNIX Specification is protected by an X/Open brand and supported by a brand verification program. X/Open brand programs provide the assurance that products that carry the X/Open name adhere to the relevant X/Open specification.

Many names have been applied to the work that has culminated in the Single Unix Specification and its attendant X/Open brand. It began as the Common API Specification, became Spec 1170, has been referred to as Unified UNIX and is now the Single UNIX Specification published in a number of X/Open Common Application Environment (CAE) volumes.

Product branding is the other half of the X/Open strategy for supporting portable and interoperable application development. The X/Open CAE defines a model for program development. X/Open brands are applied to products that conform to these same specifications. A product is branded as conforming to a component or profile and once branded uses the X/Open brand and trademark as a conformance indicator to the marketplace. Components and profiles, brands and trademarks all come together with the specifications to provide a complete model for portable application development.

A component is a well defined piece of functionality that is described in part of the X/Open CAE. For example, the XPG4 Internationalized System Calls and Libraries component covers the operating system kernel and header files as

they are described in the X/Open CAE Specification, System Interfaces and Headers, Issue 4, Version 2.

There are quite a number of components relating to the X/Open CAE, as shown below:

Area	Component
Operating Systems and Languages	XPG4 Internationalized System Calls and Libraries
	XPG4 Internationalized System Calls and Libraries Extended
	XPG4 Commands and Utilities
	XPG4 Commands and Utilities V2
	XPG4 C Language
	XPG4 Cobol Language
	XPG4 Pascal Language
	XPG4 Fortran Language
	XPG4 Ada Language
Data Management	XPG4 ISAM
	XPG4 Relational Database
User Interface	XPG4 X Window System Display
	XPG4 Motif Toolkit
	XPG4 Calendaring and Scheduling
	XPG4 X Window System Application Interface V2
	XPG3 X Window Management (Xlib)
	XPG4 Terminal Interfaces
	XPG4 Internationalised Terminal Interfaces
General Interworking	XPG4 BSFT Initiator
	XPG4 X.400 Gateway
	XPG4 X.400 Message Access
	XPG4 Directory Access
	XPG4 Network File System
	XPG4 Sockets
	XPG4 Transport Service (XTI)
Mainframe Interworking	XPG4 (PC)NFS Server
	XPG4 LMX Server
Media	XPG4 Magnetic Media

Profiles are built up from components and provide a convenient method of developing more complete functional models from the component building blocks. For example, the XPG4 Internationalized System Calls and Libraries, XPG4 Commands and Utilities and XPG4 C Language Components together define the XPG4 Base Profile. Requirements for conformance for each component need to be met to conform to a profile and additional requirements may exist because of the interactions between components.

There are several key XPG4 profiles, all of which are completely defined in the XPG4 Profile Definitions, Version 3. These are:

- XPG4 Base

  This is the XPG4 Base profile originally published in 1992. It describes a fully functional environment for portable applications development. It requires full conformance to the POSIX.1 (System Interfaces) standard and a transitional path for POSIX.2 (Shell and Utilities). Required components are:

  – XPG4 Internationalized System Calls and Libraries

  – XPG4 Commands and Utilities

  – XPG4 C Language

- XPG4 Base 95

  This is the current enhanced XPG4 Base profile requiring full conformance to the XPG4 Commands and Utilities component. Required components are:

  - XPG4 Internationalized System Calls and Libraries
  - XPG4 Commands and Utilities V2
  - XPG4 C Language

- XPG4 UNIX

  This profile is a superset of the XPG4 Base 95 and describes a platform supporting the additional functions for applications portability for programs originally developed on traditional UNIX and UNIX-derived systems. Required components are:

  - XPG4 Internationalized System Calls and Libraries Extended
  - XPG4 Commands and Utilities V2
  - XPG4 C Language
  - XPG4 Transport Service (XTI)
  - XPG4 Sockets
  - XPG4 Internationalized Terminal Interfaces

X/Open UNIX is defined by the the XPG4 profile which maps to the Single UNIX Specification. There are authorized X/Open test suites for the XPG4 UNIX profile components to be used as indicators of conformance:

- VSX4 tests the XPG4 Internationalized System interfaces and Libraries component.
- VSC4 tests the XPG4 Components and Utilities component.
- VSU4 tests the X/Open UNIX Extension of the system interfaces as well as the X/Open Sockets interfaces.
- VST4 tests the X/Open Transport Interfaces (XTI).

The XPG4 Internationalized Terminal Interfaces test suite is still under construction.

In order to clearly indicate to purchasers the exact specification to which a branded product conforms, an additional branding attribute is required when reference is made to the X/Open brand and branded products, or the trademark and the product registration.

The XPG4 profile attributes are:

- Base for XPG4 Base
- Base 95 for XPG4 Base 95
- UNIX 95 for XPG4 UNIX
- XPG4 CDE Profile
- Baseline Security 96

2.2 Operating System Changes for Spec 1170 Compliance

The vast majority of AIX V4.1 code already conformed to Spec 1170. Only minor changes were introduced in AIX V4.2 in order to provide complete conformance.

2.2.1 Compatibility and the XPG_SUS_ENV Environment Variable

To avoid breaking binary compatibility with applications which expect current AIX behavior, differences in functionality for Spec 1170 support are conditionally based upon an environment variable, the value of which is specifiable by the user. This environment variable is examined at exec() time and results in the setting of a flag in the process indicating whether Spec 1170 behavior is desired or not. The name of the environment variable is *XPG_SUS_ENV*. If the value of the variable is set to *ON*, Spec 1170 behavior is desired. Otherwise, previous AIX behavior will be used. The environment variable is checked at the same time as the environment variable for early paging space allocation. This results in negligible effects on performance. Also the behavior is always inherited across a fork() or exec() unless the environment variable is explicitly changed.

2.2.2 VMM Changes for Spec 1170 Conformance

These changes conform to the CAE Specification; *System Interfaces and Headers Issue 4, Version 2* document.

The following sections describe functional differences in the Spec 1170 definitions of VMM-related interfaces from the AIX V4.1 implementation. Each section identifies the relevant part of the standard, how the current code differs from this standard, and how the code will operate when Spec 1170 behavior is specified.

2.2.2.1 Fixed Mappings over Previous Mappings

V4.1 Implementation: AIX mmap() currently fails with ENOMEM if a previous mappings exists in the specified range.

Spec 1170 Implementation: Although the standard does not require that mappings be replaced (the code could just fail with EINVAL) applications written for other operating systems probably expect this behavior. AIX mmap() will clear any previous mappings for the region before creating the new mapping when MAP_FIXED and Spec 1170 behavior is specified. This will be done by calling vm_map_remove() for the specified range before entering the new mapping.

2.2.2.2 Clearing Partial Pages beyond EOF

X/Open mmap() statement: The implementation always zero-fills any partial page at the end of a memory region. Further, the implementation never writes out any modified portions of the last page of a file that are beyond the end of the mapped portion of the file.

V4.1 Implementation: AIX clears the partial page beyond EOF at the time the page is first faulted in. Any data that is written to this partial page by a mapper is not saved if it is beyond the end of the file. However, if the file has become larger due to a filesystem operation (such as lseek() or write()), any data in the partial page that is now within the file and was not modified by the filesystem operation will appear in the file.

Spec 1170 Implementation: The standard has been interpreted to mean that, at first creation of the page at the end of a file, the part of the page after the end of the file will be initially zero filled. Any data written after the end of file will never appear in the file on disk. It is difficult, if not impossible, to ensure that mapper stores to a partial page never appear in the file. It is extremely unlikely that an application would depend on any behavior related to mapper references beyond EOF and it was decided not to make any changes to the code.

2.2.2.3 Updates to Access, Modification, and inode Change Times

V4.1 Implementation: AIX updates either st_atime or st_time at munmap() time based on whether the mapping is shared and whether or not the file is open for writing.

Spec 1170 Implementation: It is not feasible to attempt to detect individual mapper loads and stores and so changes to the various time fields must be handled during explicit mmap calls. The requirement for st_atime is met by marking it for update at mmap time when Spec 1170 behavior is specified. Although the standard does not appear to require marking of st_ctime and st_mtime if modifications have occurred and no msync() is performed it allows this. Also, it is clear that st_ctime and st_mtime should only be set when modifications have actually occurred. Without detecting mapper loads and stores it is impossible to determine from which mapping a modification occurred or that a modification occurred via a mapper store instead of a write(). The text in msync() implies that such a determination is not necessary and that it is sufficient to update the times if any modifications are detected which result in changes being written to disk. Unfortunately, the same logic does not apply at munmap() time since modifications may exist without having caused pageouts. As a result it is really only feasible to mark the times based on any modification occuring to a file and that this be independent of whether those changes were written to disk. The existing vcs_qmodify() service can be used to determine if modifications have occurred as long as the modification state is not lost. The modification state is preserved if the JFS routine imark() is called to set the IUPD flag in the inode. The existing xix_map() and xix_unmap() routines are modified to mark times appropriately when Spec 1170 behavior is specified and a new interface is defined for use by msync(). So, when Spec 1170 behavior is specified both st_ctime and st_mtime are marked for update by msync() and munmap() if the mapping is MAP_SHARED and PROT_WRITE and the underlying file has been modified.

2.2.2.4 Unmapped Pages within an mprotect() Range

V4.1 Implementation: AIX allows the specified range to contain holes, so mprotect() succeeds.

Spec 1170 Implementation mprotect() detects a hole and fails with errno ENOMEM. The vm_map_check_protection() routine is used to ensure that the entire region is mapped.

2.2.2.5 Zero Length munmap()

V4.1 Implementation: munmap() succeeds for a length of zero.

Spec 1170 Implementation: munmap() fails with errno EINVAL for length values of zero.

2.2.2.6 Out of Kernel Resources

V4.1 Implementation: mmap() fails and sets errno to ENOMEM if no more address map entries can be allocated.

Spec 1170 Implementation mmap() fails and sets errno to EMFILE if no more address map entries can be allocated. This is accomplished by having vm_map_enter() return the value KERN_RESOURCE_SHORTAGE when creation of an address map fails. This return value results in EMFILE being returned when Spec 1170 behavior is specified and ENOMEM otherwise.

2.2.3 Filesystem Modifications for Spec 1170 Conformance

These changes conform to the CAE Specification; *System Interfaces and Headers Issue 4, Version 2* document.

2.2.3.1 fattach/fdetach Design Choices

fattach() and fdetach() are two new interfaces for attaching and detaching a STREAMS-based file descriptor to a pathname. They allow a process which has created a STREAM to associate a pathname with it. All subsequent lookups on that pathname by any process result in redirection to the STREAM object.

Examination of current implementations (OSF/Novell) show implementations of fattach() via mounts. OSF has the FFS filesystem while Novell has the NAMEFS filesystem. These filesystems are layers of indirection that provide the equivalent of a vnode stack. The implementation of fattach() and fdetach() in AIX V4.2 has been made through a stack via the specfs filesystem.

2.2.3.2 fattach() Interface Description

Library - libc.a

Synopsis

```
#include <stropts.h>

int fattach(int fildes, const char *path);
```

Arguments

fildes A file descriptor identifying an open STREAMS-based object

path An existing pathname which will be associated with fildes

Description: The fattach() function attaches a STREAMS-based file descriptor to a file, effectively associating a pathname with `fildes`. The `fildes` argument must be a valid open file descriptor associated with a STREAMS file. The `path` argument points to a pathname of an existing file. The process must have appropriate privileges, or must be the owner of the file named by `path` and have write permission. A successful call to fattach() causes all pathnames that name the file named by `path` to name the STREAMS file associated with `fildes`, until the STREAMS file is detached from the file. A STREAMS file can be attached to more than one file and can have several pathnames associated with it.

The attributes of the named STREAMS file are initialized as follows: the permission, userID, groupID, and times are set to those of the file named by `path`, the number of links is set to 1, and the size and device identifier are set to those of the STREAMS file associated with `fildes`. If any attributes of the named

STREAMS file are subsequently changed, neither the attributes of the underlying file nor the attributes of the STREAMS file to which fildes refers are affected.

File descriptors refering to the underlying file, opened prior to a fattach() call, continue to refer to the underlying file.

Application usage: The fattach() function behaves similarly to the traditional mount() function in the way a file is temporarily replaced by the root directory of the mounted filesystem.

2.2.3.3 fdetach() Interface Description

Library - libc.a

Synopsis

```
#include <stropts.h>

int fdetach(const char *path);
```

Arguments

path Pathname of a file previously associated with a STREAMS-based object using the fattach() subroutine.

Description: The fdetach() function detaches a STREAMS-based file from the file to which it was attached by a previous call to fattach(). The path argument points to the pathname of the attached STREAMS file. The process must have appropriate privileges or be the owner of the file. A successful call to fdetach() causes all pathnames that named the attached STREAMS file to again name the file to which the STREAMS file was attached. All subsequent operations on path will operate on the underlying file and not on the STREAMS file.

All open file descriptors established while the STREAMS file was attached to the file referenced by path will still refer to the STREAMS file after the fdetach() has taken effect.

If there are no open file descriptors or other references to the STREAMS file, then a successful call to fdetach() has the same effect as performing the last close() on the attached file.

The unmount command may be used to detach a file name if an application exits before performing fdetach().

2.2.4 Process Management Modifications for Spec 1170 Conformance

These changes conform to CAE Specification; *System Interfaces and Headers Issue 4, Version 2* document.

2.2.4.1 Signal Changes

X/Open Spec 1170 specifies two new APIs for signal management that are not currently provided by AIX Version 4. The two new APIs are as follows:

bsd_signal() - simplified signal facilities

sigalstack() - set and/or get signal alternate stack context

In addition, the specification documents changes to the following API that is currently provided by AIX Version 4:

sigaction() - examine and change signal action

2.2.4.2 bsd_signal()

Synopsis

```
#include <signal.h>

void (*bsd_signal(int sig, void(*func))) (int);
```

X/Open Spec 1170 states that bsd_signal() ″is a direct replacement for the BSD signal() function for simple applications that are installing a single-argument signal handler function.″

This function exists in AIX V4.1 as signal() in libbsd.a

The signal() function has been ported from libbsd.a to bsd_signal() in libc.a.

2.2.4.3 sigalstack()

Synopsis

```
#include <signal.h>

int sigalstack(const stack_t *ss, stack_t *oss);
```

X/Open Spec 1170 specifies this new API for signal stack management. The sigalstack() function replaces the current signal stack management function sigstack(), which will be withdrawn from the specification in the future. AIX Version 4.2, though, will continue to support sigstack().

2.2.4.4 sigaction()

AIX Version 4 currently supports the sigaction() API; however, functionality has been expanded to fully support the definition in X/Open Spec 1170. The following flags are new to the specification:

SA_SIGINFO: If this flag, not implemented in AIX V4.1, is not set and the signal is caught, according to X/Open Spec 1170, the signal-catching function will be entered as:

```
void func(int signo);
```

where signo is the only argument to the signal catching function.

In fact, AIX Version 4.1 implements the signal handling function to be entered as follows:

```
void function(int signo, int code, struct sigcontext *scp);
```

It is at the discretion of the application to make use of the code and scp parameters.

However, X/Open Spec 1170 specifies that if SA_SIGINFO is set and the signal is caught, the handler will be entered as:

void func(int signo, siginfo_t *info, void *context);

where info points to an object of the type siginfo_t and the context can be cast to a pointer to an object of the type ucontext_t.

If SA_SIGINFO is not set the binary compatibility is maintained.

SA_NOCLDWAIT: X/Open Spec 1170 states ″if set, and sig equals SIGCHLD, child processes of the calling processes will not be transformed into zombie processes when they terminate.″ With respect to process termination, this has the same effect as setting SIGCHLD to SIG_IGN.

SA_NODEFER: X/Open Spec 1170 states that ″if this flag is set and the signal is caught, the signal handler is invoked unless it is included in a sa_mask. Otherwise, the signal will always be added to the signal mask upon entry to the signal handler.″

AIX V.4 always adds the signal to the process′ signal mask upon entry to the signal handler unless the SA_OLDSTYLE flag is set.

2.2.5 waitid() API

X.Open Spec 1170 specifies two new APIs for child process management. They are as follows:

waitid()

wait3()

The specification also documents changes to the following APIs:

wait()

waitpid()

AIX Version 4.1 currently provides wait(), waitpid(), and wait3(), however modifications to all three functions were necessary for conformance.

2.2.5.1 waitid()

Synopsis

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

int waitid(idtype_t idtype, it_d id, siginfo_t *infop, int options);
```

X/Open Spec 1170 states ″the waitid() function suspends the calling process until one of its children changes state. It records the current state of a child in the structure pointed to by infop.″ This is, in fact, an extension of the functionality provided by the system call kwaitpid().

The calling interface of the waitid() service differs from waitpid() in that the child processes are classified using the idtype and id parameters. This interface, though, translates directly to the convention used by waitpid().

X/Open Spec 1170 states, ″if idtype is P_PID, waitid() will wait for any child with a process group ID equal to (pid_t)pid.″ This is equivalent to calling waitpid() and specifying a process ID less than -1. Finally, the specification states ″if idtype is P_ALL, waitid() will wait for any children and is ignored.″ This is equivalent to calling waitpid() and specifying a process ID of -1.

The function waitid() differs from waitpid(), wait(), and wait3() in that if a process wishes to wait for children that have stopped or exited, this must be specified by setting options to WSTOPPED or WEXITED, respectively. These state changes are always specified by default in the other services.

In addition, the new option WNOWAIT can be specified only in waitid() and is used to query the state of the child process without changing the state of the process. If this option is specified, the process can be waited for again after the call completes.

X/Open Spec 1170 also specifies another new option, WCONTINUED that can be specified to return the status of a child process that was stopped and has been continued, but not reported.

2.2.5.2 wait(), waitpid()

The waitpid() service now accepts the WCONTINUED option which allows reporting of a child process that has been stopped and continued, but not reported.

In addition, a new macro has been defined to interpret the status for processes that have been continued. The new macro is WIFCONTINUED(stat_loc) and evaluates to a non-zero value if the status was returned for a child process that has continued from a job control stop.

There is no specific change to the wait() service, but the specification states ″if the calling process has SA_NOCLDWAIT set or has SIGCHLD set to SIG_IGN, and the process has no unwaited for children that were transformed into zombie processes, it will block until all of its children terminate, and wait() and waitpid() will fail and set errno to ECHILD.″ So, the operation of wait() has been changed based on the use of the SA_NOCLDWAIT signal flag.

2.2.5.3 wait3()

AIX V4.1 currently provides the service wait3(); however, the extensions to the function waitpid() apply to wait3() as well.

2.2.6 getpid() and getsid() APIs

2.2.6.1 Description

X/Open Spec 1170 specifies two new APIs providing process specific information. These two new APIs are as follows:

getpgid() - get process group ID

getsid() - get process group ID of a session leader

AIX Version 4.1 already included these two APIs; however their specific implementations fell short of the requirements.

2.2.6.2 getpgid()

Synopsis

```
#include <unistd.h>

pid_t getpgid(pid);
```

X/Open Spec 1170 specifies, ″the getpgid() function returns the process group ID of the process whose process ID is passed as pid. If pid is equal to 0, getpgid() returns the process group ID of the calling process.″ Upon failure, it returns (pid_t)-1 and sets errno to indicate the error.

AIX Version 4 already implemented this functionality, however, the specification requires permission checking to be done, which AIX 4.1 does not do.

2.2.6.3 getsid()

Synopsis

```
#include <unistd.h> pid_t getsid (pid_t pid);
```

X/Open Spec 1170 specifies, ″the getsid() function obtains the process group ID of the process that the session leader of the process specified by pid. If pid is (pid_t)0, it specifies the calling process.″

AIX V4.1 already implements this functionality. Although kgetsid() does return the process group ID, it is returning the session ID which happens to be equivalent for AIX 4.1. As with getpgid(), the specification indicates permission checking should be done which AIX 4.1 does not do.

2.2.7 Resource Limits Changes

X/Open Spec 1170 specifies two new APIs providing resource limit controls. These two new APIs are getrlimit() and setrlimit().

AIX Version 4.1 already defines these two APIs; it is lacking however in the type of resources that can be controlled and the actual enforcing of these limits.

2.2.7.1 RLIMIT_NOFILE

X/Open Spec 1170 specifies RLIMIT_NOFILE as a new resource control which is not implemented in AIX Version 4.1. This resource limit controls the maximim number of allocated file descriptors that a process may use.

Thus the new resource limit RLIMIT_NOFILE, has been added to /usr/include/sys/resource.h.

2.2.7.2 RLIMIT_STACK

This resource limit is not fully implemented in AIX V4.1. The limit is properly managed and VMM enforces the limit by posting SIGSEGV to the process when the stack is exceeded. However, X/Open Spec 1170 specifies ″if the process is blocking or ignoring SIGSEV, or is catching SIGSEV and has not made arrangements to use an alternate stack, the disposition of SIGSEGV will be set to SIG_DFL before it is generated.″ This is not implemented in AIX V4.1.

The default exception handler now checks to see if the process is blocking or ignoring SIGSEGV, or is catching SIGSEGV and has not made arrangements to use an alternate stack. If either of these cases is true, default_uexcp() will change the disposition of the signal to SIG_DFL prior to generating the signal.

2.2.7.3 RLIMIT_AS

X/Open SPEC 1170 specifies RLIMIT_AS as a new resource limit which is not implemented in AIX Version 4.1. This resource limit controls the maximum size of a process′ total available memory, in bytes.

Therefore the new resource control, RLIMIT_AS, has been defined in /usr/include/sys/resource.h.

2.2.8 User Context APIs

X/Open Spec 1170 specifies four new APIs for user context management that are not provided by AIX Version 4.1. These new APIs are as follows:

- getcontext() - get current user context
- setcontext() - set current user context
- makecontext() - manipulate user contexts
- swapcontext() - manipulate user contexts

New library functions have been created to support these specifications.

X/Open Spec 1170 is not very clear regarding which of the above functions should work from signal handlers. The Application Usage section for setcontext() indicates that this function could be called from a signal handler. None of the other functions listed above are mentioned in this respect. As such, setcontext() will be callable from signal handlers, but the remaining functions may have undefined behavior when called from signal handlers.

2.2.8.1 getcontext()

Synopsis

```
#include <ucontext.h>

int getcontext(ucontext_t *ucp);
```

This service is used to get the current user context of the calling process. The structure pointed to by ucp is initialized with the current context specific information that can later be used to return to this context via the setcontext() function. Information returned from getcontext() ″includes the contents of the calling process′ machine registers, the signal mask, and the current execution stack.″

2.2.8.2 setcontext()

Synopsis

```
#include <ucontext.h>

int setcontext(const ucontext_t *ucp);
```

This service is used to restore the context pointed to by ucp. ″A successful call to setcontext() does not return; program execution resumes at the point specified by the ucp argument.″

2.2.8.3 makecontext()

Synopsis

```
#include <ucontext.h>

void makecontext(ucontext_t *ucp, void (*func)(), int argc, ...);
```

This function modifies the context pointed to by ucp, such that when the context is resumed, ″execution is continued by calling func(), passing it the arguments that follow argc in the makecontext() call.″ When func() returns, execution of the process continues with the context specified by the uc_link element in ucp.

Callers of makecontext() should allocate a stack for the context being modified prior to calling makecontext(). This is done by setting the uc_stack element of the ucontext_t structure and specifying the address and size of the stack.

2.2.8.4 swapcontext()

Synopsis

```
#include <ucontext.h>

int swapcontext(ucontext_t *oucp, const ucontext_t *ucp);
```

X/Open Spec 1170 states, ″the swapcontext() function saves the current context in the context structure pointed to by oucp and sets the context to the context structure pointed to by ucp.″

This is equivalent to calling getcontext() passing in oucp, and calling setcontext() passing in ucp. The contexts are not linked together by swapcontext().

2.3 X/Open Branding for Common Desktop Environment

X/Open announced in March 1995 the Common Desktop Environment brand. This brand will identify product implementations that conform to the X/Open Common Desktop Environment specifications for open systems. The X/Open CDE brand will be added to the existing X/Open brands.

The X/Open CDE Brand ensures conformance with:

- XPG4 X Window System Application Interface V2
- XPG4 Motif Toolkit
- XPG4 Calendaring and Scheduling
- XPG4 X Window System Display

2.4 The Year 2000 Problem

Many computer systems and applications use two digits to represent the year. In these environments the change to dates of 1999, 2000 and beyond may be fatal to the accuracy of the data created by a wide range of applications from word processors to databases.

If it is not addressed quickly, this date change may affect the calculations, comparisons and data sorting in applications from the desktop on up to the largest server. The price tag for potential errors could be extremely high and have a major impact on operations of all types. Many computer operating systems and applications use a standard two-digit format, MM/DD/YY, to represent a date. Such a computer would write January 1, 1996 as 01/01/96. January 1, 1999 would be 01/01/99. When the year rolls over to 2000, computers that use the two-digit format will probably write 01/01/00, and therein lies the problem.

Many programs calculate the length of time between dates by subtracting two-digit years from each other. For example, a bank computer does this simple calculation when it figures the principal and interest on a 30-year mortgage. There is no problem if the mortgage was issued in 1965 and paid in full in 1995. But if the mortgage is issued today and the application uses the two-digit date format, it may well read the year as ″00″ and be unable to do the calculation. The application may assume the year is 1900 and return an error message, or even worse, continue regardless.

Why wasn′t it fixed before now? Often date fields can′t easily be expanded to include more than two numbers. Nor can computers be instructed to globally insert the digits *one* and *nine* into years, because some dates may refer to a different century. One solution is to painstakingly find every point in a system that uses the date to trigger a calculation or a routine and rewrite the source code. All applications must be changed in a coordinated fashion and tested to make sure they handle dates in both centuries.

Why didn′t the Information Technology industry foresee from the beginning that two-digit dates would be a problem and use four digits to represent the year?

In the 1960s and 1970s, when many computer programs were created, programmers had incentives to abbreviate dates and save data entry time. Computer memory and storage were costly and in short supply. Saving several characters in every database with millions of records was worth it. Early programming languages and standards didn′t offer application program interfaces (APIs) for four-digit years. Even if they were aware of the pitfall, programmers may have assumed the applications they were writing would be scrapped long before they could cause problems. Remarkably, many of those early programs are still in use, often as the core of company information systems, and that data is still in the same format.

The year 2000 problem is further compounded by the numerous variations used to express year and date notation in data and the mathematical calculations performed on those date notations. These variations can be classified as follows:

First two-digit year problems: Problems can occur when the first two digits in a year are assumed to be 19 and ignored during data entry and update, or hardcoded for output.

Last two-digit year problems: Special values of the last two digits in a year might be used for a special purpose, for example 99 might be used to indicate no expiration date or 00 to indicate an unknown year.

Incorrect field format determination: Some existing format programs determine the date-time format (MMDDYY, DDMMYY, YYMMDD) by testing an appropriate part of the date field. For instance, checking if the first two of the date field is between 01 and 31 to determine if the date format is YYMMDD or DDMMYY. This checking is suitable for years from 1932 through 1999 but will fail when the year is in the range of 2000 through 2031.

Arithmetic calculation problems: The arithmetic calculations that operate on dates with two digit year representation might have potential exposures. For example, a person with a birthday of March 30, 1966 could be considered to be 66 years old rather than 34 years old on March 30, 2000 if the years 1966 and 2000 are represented by 66 and 00 respectively.

Sequence problems: When only two digits are used to represent a year, programs that collate year data will sort that data out of sequence in some cases. For example the year 2000 will be ordered prior to the year 1999.

Data integrity problems: In programs where historical dates are used, for example all events occurring in different centuries are not distinguishable when the years are represented by only two digits.

Leap year calculation: A specific year is considered to be a leap year if it is either evenly divisible by 400 or by four and not evenly divisible by 100. Potential exposures caused by the identification of the year 2000 as a non-leap year are day in a year calculations, day of the week calculations and week of the year calculations.

The challenge of handling the Year 2000 changes may occur well before the year 2000 arrives. Typically, forecasting applications that deal with future dates will encounter problems well in advance of the year 2000.

2.4.1 Solutions to Year-Date Notation

Different solutions can be applied to remove the above Year 2000 exposures. The following techniques require both program and data changes and have both advantages and disadvantages.

2.4.1.1 Conversion to Full Four-Digit-Year Format

This solution is a four-digit solution that externalizes a four-digit-year format.

This approach requires changes to both the data and the programs by converting all references and/or uses of two-digit-year format (YY) to four-digit-year format (YYYY). It also requires that you convert all software programs that reference or use the updated data simultaneously, or use a *bridging* mechanism to perform the conversion between old and new data and programs. Otherwise, you will immediately encounter data integrity problems caused by the inconsistency of date/time data formats.

To ease your migration, you might consider ignoring any non-impact (cosmetic) data fields in the YY format. A cosmetic date is one, that if externalized, is only interpreted by humans. Such occurrences might include the date on an output separator page or a display-only date on a screen in a panel-driven application.

Pros

Provides a full four-digit-year format. It is considered to be the *only* complete, permanent, and obvious solution.

Provides increased security against potential inappropriate decisions today if you do not selectively ignore *cosmetic-only* situations.

Can ease your migration if you selectively ignore *cosmetic-only* situations.

Cons

Needs conversion of the year data from two-digit format to four-digit format in all cases.

Requires that you relocate adjacent fields in the date field layout, and usually requires that you increase record lengths.

Inherent future risk in initial assessment that determined a particular situation can be ignored as *cosmetic only*.

Increased DASD space usage required due to data field expansion of data (consider including not only active but also archive data) and the duplicate DASD space required during conversion.

2.4.1.2 Windowing Techniques

This is a two-digit solution that externalizes either two-digit or four-digit-year formats. This approach requires changes to your programs only; no data changes are required.

Caution

These windowing techniques can only be applied to dates within a maximum 100-year period at any one time. This solution is considered temporary since there is no guarantee that in the future your applications will not expand to encompass dates that are more than 100 years apart. Therefore, this approach always carries with it a potential future exposure. For example, humans are living longer, therefore, dayabases that include birthdays (medical, civil, insurance and so on) and the applications that access that data are already at risk with many dates spanning 100+ years.

Two types of windowing techniques have been defined: the fixed window technique and the sliding (rolling) window technique.

Fixed Window Technique: The fixed window technique uses a static 100-year interval that generally crosses a century boundary. This technique determines the century of a two-digit year by comparing the two-digit year against a window of 100 years. The user specifies the number of years in the past and future relative to a specific year within the 100-year interval.

Consider this specific example: If the years of date-related data of your application fall in the range of 1 January 1960 to 31 December 2059, you can use a two-digit year to distinguish dates prior to the year 2000 from the year 2000 and beyond. If using the current system year of 1996, the number of years in the

past and future are specified as 36 and 63, respectively. Program logic determines the century based on the following data checking. If the two-digit year representation of a specific year is ″xy″ then if:

″xy″ is more than/equal to 60, then it is a 20th century date (19″xy″)

Otherwise (that is, ″xy″ is less then/equal to 59), it is a 21st century date (20″xy″)

If, for example, you need to maintain a window of 36 past years and 63 future years, such that next year, 1997, your application can successfully deal with dates in the range 1961 through 2060, you need to adjust this program checking every year. The inherent future risk when employing this technique is obvious, and when compared to the sliding window technique is far less desirable.

Pros

No need to expand the two-digit-year data to a four-digit format.

Can provide four-digit-year format for data reference.

Can distinguish years from different centuries using only two-digit-year format (provided the years being processed are in the range of 100 years at any one time).

Can be useful if the particular program is being phased out, and a temporary solution is appropriate.

Cons

Potential exposures exist when/if the function of the software application needs to process years beyond the range of 100 years.

Expect a performance impact in direct proportion to the quantity of date processing the particular application handles due to the overhead of two to four-digit-year conversion.

All programs that use the fixed window technique may need to be manually updated on a yearly basis depending on how your date routine is packaged.

All programs that accept output from the fixed window technique must use the same assumptions (current date, past and future windows).

Retaining a two-digit year representation does not provide collating sequence support. Nor does the use of a fixed window technique provide indexing sequence support when two-digit years are used as index keys in indexed files. You will need to provide additional processing to obtain correct collating and indexing sequence output.

Sliding Window Technique: The sliding window technique uses a self-advancing 100-year interval that generally crosses a century boundary. This technique determines the century of a two-digit year by comparing the two-digit year against a window of 100 years. The user specifies the number of years in the past and future relative to the system year (generally the current year) that the system sets and maintains. Your applications can access the date that the system sets and automatically advances. This is the main advantage of using a sliding window over the fixed window (where the window is immovable without manually revising the programs each year).

As appropriate to your application environment, you can maintain more than one window. For example, you could set one window to process historical dates, one for mortgage dates, one for birth dates, and so on; and the program adjusts the

system date and past and future windows to meet the specific application's needs.

Consider this specific example. If the dates in your application fall into a range of 35 years in the past and 64 years into the future, based on the current year, 1996, your program can accept and accurately deal with dates of 1961 through 2060. Next year, 1997, the window advances and your application accurately deals with dates of 1962 through 2061.

A sliding window approach requires programming logic to interpret the meaning of all two-digit year data. Such additional programming logic could be packaged into a common date/time service routine, callable from a two-digit year data exploiter. This would reduce the programming overhead and impact to the calling programs.

Pros

No need to expand the two-digit-year format to a four-digit format.

Can provide four-digit-year format for data reference.

Can distinguish years from different centuries using only two-digit-year format (provided the years being processed are in the range of 100 years at any one time).

No need to convert the date data to a new date representation scheme.

Cons

Potential exposures exist when/if the function of the software application needs to process years beyond the range of 100 years.

Potential performance impact in direct proportion to the quantity of date processing the particular application handles.

All programs that accept output from the sliding window technique must use the same assumptions (current date, past and future windows)

Retaining a two-digit year representation does not provide collating sequence support. Nor does the use of a sliding window technique provide indexing sequence support when two-digit years are used as index keys in indexed files. You will need to provide additional processing to obtain correct collating and indexing sequence output.

A Two-Digit Encoding/Compression Scheme: This is a two-digit solution that externalizes only a two-digit-year format. It requires changes to both your data and your programs. It also requires that you convert, simultaneously, all applications that reference or use the updated data.

Caution

Apply this approach with caution. It is considered to be the least desirable approach and should only be used if absolutely necessary. Be certain that the new encoding or numbering scheme does not affect the proper functioning of your programs after all the data changes are implemented. This solution is considered temporary because there is no guarantee that in the future, your applications will not expand to encompass dates that are outside the encoding limits.

Pros

No need to expand the two-digit-year data format to four-digit data format. (For example, there is no need to increase the fields in data bases and tables to accommodate dates above 99 which would increase DASD usage.) Further, this saves the effort that would be required to rebuild your database(s).

Can distinguish years from different centuries using the two-character-year format.

If you use a flagged Julian format (CYYDDD) where ″C″ is used as the ′century indicator′, the format does not require expansion of the date field.

Cons

Depending upon the choice of data representation you implement, this scheme can be applied only to a limited date range. For example, you are limited to 255 years when using hexadecimal representation.

All programs that use this scheme and need to access the output of the two-character conversion must change simultaneously.

Due to data conversion (calls and processing) you might experience a performance impact in direct proportion to the quantity of date processing the particular application handles.

Depending upon the choice of data representation you implement, you might experience incorrect data sequencing if you do not add further programming logic.

Encoded dates require conversion whenever you work with that data. Therefore, the presence of encoded dates will add another layer of complexity to such tasks as problem determination.

You must convert the data before it can be displayed in Gregorian format, and some encoded data can only be viewed in hexadecimal format. This is both impractical for human reading and also impractical or impossible to print.

2.4.2 IBM Year 2000 Customer Assistance

IBM′s ISSC has developed a comprehensive set of solutions that takes into account applications, systems software and hardware in both centralized and distributed environments. This comprehensive solution set is called Transformation 2000.

The Transformation 2000 approach seeks to balance Year 2000 investment activities with current and planned strategic initiatives (for example, new architectures and new application development). ISSC brings together state-of-the-art techniques and technologies developed and proven through both internal and external projects for the Year 2000 and other data field expansions. This experience enables ISSC to help reduce both the cost and the complexity of implementing the Year 2000 change.

See *IBM Announcement Letter 25744* dated 31 October 1995 for additional information.

2.5 Operating System Changes for Year 2000 ISO 8601 Compliance

AIX 4.2 conforms to Year 2000 requirements and to proper leap year calculations. It also conforms with *ISO 8601: 1988 Data Elements and Interchange Formats - Information Interchange Representation of Dates and Time*.

The hardware timers on RS/6000 machines are not sensitive to change of the century, as are the timers on S/390 and AS/400.

AIX V4.1 is almost capable of fully supporting the change of the century. Only minor changes have been implemented to make AIX V4.2 a platform totally Year-2000 safe.

Regardless of the solutions used to fix these problems you should remember that the underlying UNIX design only supports dates up to the year 2038. This limitation isn't considered a current problem, at least until that year, since system date is used to simply represent the date of the current day.

The design suggested to fix this problem uses a fixed window technique maintaining a two-digit representation working on two ranges 00-38 and 39-99.

2.5.1 User Account Expiry Attributes

On AIX V4.1 the *expires* attribute in the /etc/security/user file only uses two digits to represent the expiration date of an account. To fix this problem both libs.a (chkexpires()) and the `usrck` command have been changed.

The change is to allow the expires attribute to assume the year range is from 1970 through 2038 (like the `date` command does), instead of 1900 through 1999, using a fixed window technique.

2.5.2 chuser Command

Similar to the expires attribute mentioned above, the `chuser` command has been changed to support years from 1970 through 2038 using a fixed window technique.

2.6 AIX/6000 Year-2000-Ready Program Products

This paragraph presents AIX/6000 products that are available at the time of writing as Year 2000 ready.

This list is not meant to be exhaustive or exclusive, but is intended to answer the most common questions.

PROGRAM PRODUCT NUMBER	PRODUCT NAME (VERSION/RELEASE)
5765-564	ADSTAR Distributed Storage Manager V2 R1
5697-078	ADSTAR Distributed Storage Manager V1 R2.1
5765-268	AIX Asynch.Term.Svr.Accelerator/6000
5765-266	AIX CallPath Server/6000
5765-117	AIX DCE Base Services/6000 V1
5765-119	AIX DCE Cell Dir Server/6000 V1
5765-121	AIX DCE Enhanced Dist.File Sys./6000 V1
5765-120	AIX DCE Global Dir.Server/6000 V1
5765-259	AIX DCE Global Dir.Client/6000 V1
5765-118	AIX DCE Security Server/6000 V1
5765-232	AIX DCE Threads/6000 V1
5765-001	AIX DirectTalk/6000
5696-902	AIX Distributed SMIT V2.2 for AIX
5765-042	AIX EngSci Subroutine Lib./6000 V2 R2
5696-708	AIX File Storage Facility (FSF)
5756-030	AIX for RISC System/6000 Version 3.2.5
5696-923	AIX HACMP/6000 V3.1
5696-933	AIX HACMP/6000 V4.1 for AIX V4
5765-551	AIX HIPPI/6000 V3 R2 M3
5696-108	AIX InfoCrafter/6000 V1.1
5696-893	AIX InfoCrafter V2.1
5621-107	AIX NetView Service Point 1.2
5621-013	AIX OSL/6000 Version 1.2
5765-296	AIX Parallel System Support Programs V1.2
5765-529	AIX Parallel System Support Programs for AIX V2.1
5696-899	AIX Performance Aide/6000 V2.1
5696-900	AIX Performance Toolbox/6000 V2.1
5765-349	AIX /SMARTsort for Workstation
5765-261	AIX SNA Gateway/6000 V2.2
5696-906	AIX Ultimedia Services/6000 V2.1.1
5765-393	AIX Version 4.1.3
5696-868	AIX X.25 V1.1
5765-011	AIX X-Windows 3270 Emulator/6000
5601-248	AIX XL FORTRAN/6000 Version 2.3
5765-018	AIX XL Fortran Compiler/6000
5765-019	AIX XL Fortran Runtime Env/6000
5765-176	AIX XL Fortran for AIX V3.2
5601-251	AIX XL Pascal Runtime Env./6000 (withdrawn April 1995)
5601-254	AIX XL Pascal Compiler/6000 Version 1.1.2 (service withdrawn April 1995)
5765-245	AIX XL Pascal Compiler/6000 Version 2 (replaces 5601-251 and 5601-254 above)
5601-260	AIX 3270 Host Conn/6000 Version 1.3
5765-249	AIX 5080 Emulation Program/6000
5765-398	AIXlink V2.1.1
5696-926	AIXlink/X.25 V1.1
5696-904	AIXwindows Display Postscript V1.1
5601-257	AIXwindows Environment/6000 V1.2.5
5756-027	AIXwindows Interface Composer/6000 (AIC) V1.2
5765-423	C for AIX
5765-421	C Set ++ for AIX
5626-COM	CATIA Object Manager V4 R1 M6 (and associated products)
5765-148	CICS for AIX V1 R2
5765-152	CICS Client for AIX V1 R2
5765-427	CICS System Manager V1.1 for AIX
5801-AAR	COBOL Set V1.1.0 for AIX
5765-561	CommonPoint Application System for AIX V1.1
5765-562	CommonPoint Application Development Toolkit V1.1
5765-652	Communications Server for AIX V4
5765-022	Consumer Transaction Definition/6000
5767-023	Consumer Transaction Runtime/6000

5765-418	Data Encryption Std Lib Routine V1.1
5765-642	Database Server V4
5765-256	DataHub Support/6000 V3.2 (service withdrawn October 1995)
5801-AAR	DataHub for UNIX Operating Systems (replaces 5765-256 above)
5871-AAA	DB2 for AIX V2.1
5765-459	DB2 SDK/6000 V1 R2
5765-453	DB2 SDK for AIX V2 R1
5765-464	DB2/6000 V1 R2
5871-AAA	DDCS for AIX V2.3
5696-239	Encina Monitor for AIX/6000
5696-238	Encina PTP Exec for AIX/6000
5696-347	Encina PTP Gateway for AIX/6000
5696-240	Encina Server for AIX/6000
5696-237	Encina Structured File Svr for AIX/6000
5765-527	Extended Systems Administration Feature (SystemView)
5696-923	High Availability Cluster Multi-Processing V3 R1 M1
5765-482	LANDP/6000 V2.1
5801-AAR	FlowMark for AIX V2 R2 (replaces 5765-270 FlowMark for AIX V1 R1)
5765-026	geoGPG/6000
5696-919	Hypertext Information Base Libraries V1.1
5696-898	InfoExplorer License Extension V1.1
5765-223	InterMix for AIX Version 1.2
33H4191	Internet Connection Secure Server for AIX
33H4190	Internet Connection Server for AIX
33H4290	Internet Connection Secured Network Gateway V2.1
5765-527	Job Scheduler Feature (SystemView)
5765-264	LAN Management Utilities/6000 V1.1.3
5765-145	LoadLeveler Version 1.2.1
5765-449	MERVA AIX V1 R1 (supports the definitions in the SWIFT User Handbook)
5765-115	MQSeries for AIX Version 2.1.0 (service to be withdrawn 31 Dec 1996. Announced 28 March 1992, letter No 295-144)
5765-550	NetBIOS & IPX/SPX Support/6000 V2
5871-BBB	NETSP Secured Network Gateway 1.2 (feature 6362)
5765-196	NetView Distribution Manager for AIX V1.1 (service to be withdrawn Sep 1996. Announced 5 Dec 1995, letter No 995-140)
5765-214	NetView Distribution Management Agent for AIX V1.1 (service to be withdrawn Sep 1996. Announced 5 Dec 1995, letter number 995-140)
5696-731	NetView for AIX 3.1
5622-242	NetView FTP Client V1.1 for AIX
5765-435	NetView FTP Server V1.1 for AIX
5696-236	Netware for AIX/6000
5696-939	OpenGL & GL 3.2 Version 4.1.3
5765-352	OpenMail for AIX B.02 (withdrawn from service Jan 1996. Announced 10 Oct 1995, letter number 995-090)
5765-543	Parallel Environment for AIX V2.1
5765-422	Parallel ESSL for AIX V1.1
5765-297	Parallel I/O File System V1.1
5765-392	Parallel OSL (OSLp) V1.1.1
5765-469	Parallel Visual Explorer
5765-527	Performance Reporter Feature (SystemView)
5696-907	Pex & PHIGS Version 4.1.3
5765-193	ProductManager Application Svcs Mgr (and associated products for DB2)
5765-440	ProductManager Application Svcs Mgr (and associated products for Oracle)
5765-605	ProductManager Version 3 Release 1 for AIX
5756-094	PROFESSIONAL CADAM Interactive Design (and assoc. products)
5765-505	PSF for AIX Version 2.1
5765-246	PVMe for AIX V1 R3
5765-544	PVMe for AIX V2.1

5696-038	Realtime Interface Co-Processor AIX Support V1.1.1
5765-444	Recoverable Virtual Shared Disk V1.1
5765-292	RMONitor V1.1 for AIX
5765-343	Router and Bridge Manager/6000 V2.3
5765-233	SNA Manager/6000 V1.1
5765-247	SNA Server/6000 V2.2
5765-410	Systems Monitor for AIX V2.2
5765-527	SystemView for AIX
5697-213	The Multimedia Server for AIX
5765-265	Trouble Ticket V3.2
5765-400	UIM/X V2.8
5696-398	UniTree for AIX/6000 (service withdrawn Dec 1995)
5765-315	WABI V2.0

Notes:

- 5706-294, AIX ADA Run-Time Env/6000 Composer and 5706-291, AIX ADA/6000 Version 1.2.3 were withdrawn from service November 1994 and have since been transferred to OC Systems, Inc.

- 5756-085, AIX OSI Messaging and Filing/6000 was withdrawn from service December 1994 and has been replaced by ObjectStore by Object Design, Inc.

Chapter 3. System Management

Some of the most important enhancements to AIX in Version 4.2 were in the area of system management and facilities. Support for large files, big executables and dynamic loading and other changes are covered in this chapter.

3.1 Overview - Large File Support

In previous versions of AIX, the size of a single file was limited to 2 GB due to the signed 32 bit definition of off_t. off_t represents the offset of the file pointer in a file (see /usr/include/sys/types.h). AIX now provides applications developed on Version 4.2 the ability to create and use data files greater than 2 GB in size. The AIX large file support is based on the 64 bit data type now available in the C compiler.

3.2 Large File Enabled Journaled File System

In order to support large files, a new filesystem geometry was developed for the AIX Journaled File System. To identify this new filesystem geometry to the system it was given a new File System Version Number of 2. Only filesystems created specifically for large files are able to accomodate files greater than 2 GB.

3.3 Large File Geometry

In filesystems enabled for large files, file data stored before the first 4 MB file offset is allocated in 4096 byte blocks. File data stored beyond the 4 MB file offset is allocated in large disk blocks of 128 KB. These large disk blocks are actually 32 contiguous 4096 byte blocks. For example, a 132 MB file in a filesystem enabled for large files has 1024 4 KB disk blocks and 1024 128 KB disk blocks. In a regular filesystem, the 132 MB file would require 33 single indirect blocks (each filled with 1024 4 KB disk addresses). However, the large file geometry requires only two single indirect blocks for the 132 MB file. The maximum file size for this geometry is:

(1*1024*4K) + (511*1024*128K) or 68,589,453,312 bytes.

© Copyright IBM Corp. 1996

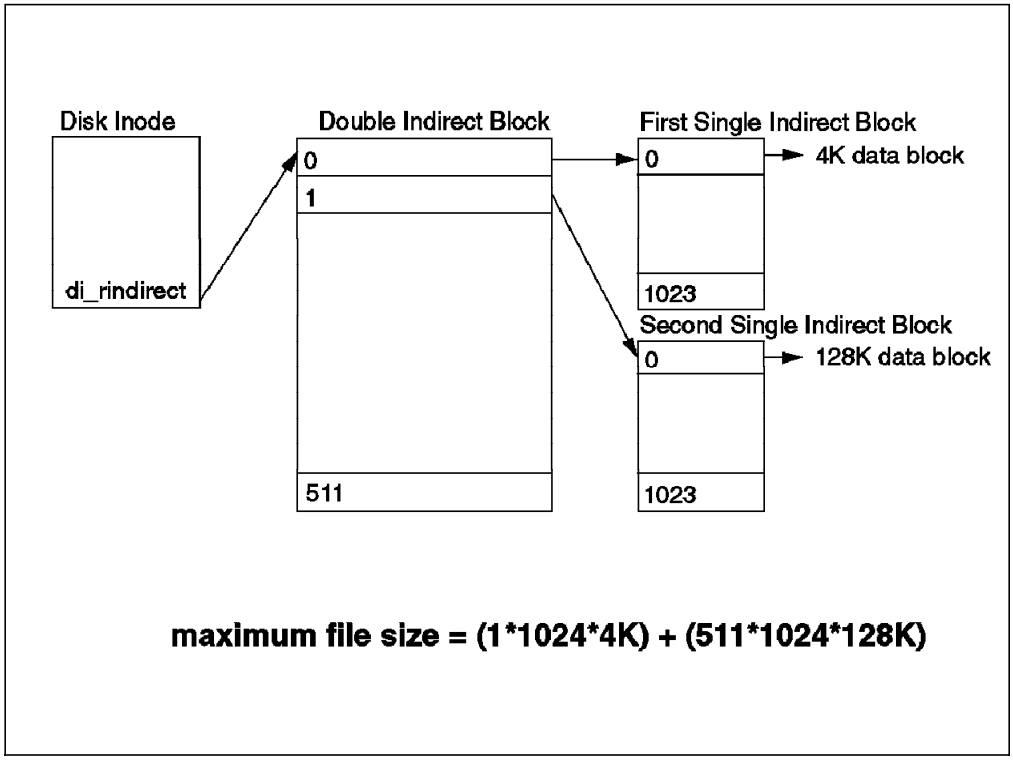


Figure 3. Large Filesystems Geometry

3.3.1 Sparse File Allocation

Files that do not have disk blocks allocated for each logical block are called sparse files. Sparse files are created by seeking to two different file offsets and writing data. If the file offsets are greater than 4 MB, then a large disk block of 128 KB is allocated. Applications using sparse files larger than 4 MB may require more disk blocks in a filesystem enabled for large files than in a regular filesystem.

3.3.2 Free Space Fragmentation

Large disk blocks require 32 contiguous 4 KB blocks. If an attempt is made to write beyond the 4 MB file offset and the filesystem does not contains 32 unused contiguous blocks then the write will fail with error ENOSPC.

Note

The filesystem may have thousands of free blocks, but if 32 of them are not contiguous, the allocation will fail. The `defragfs` command reorganizes disk blocks to provide larger contiguous free block areas.

3.3.3 crfs Command

The `crfs` command creates a file system on a logical volume within a previously created volume group. An entry for the filesystem is put into the /etc/filesystems file.

To accomodate filesystems enabled for large files in excess of 2 GB two new options have been added to the `-a` flag.

-a ag={ **8 | 16 | 32 | 64** }

Specifies the allocation group size in megabytes. An allocation group is a grouping of inodes and disk blocks similar to BSD cylinder groups. The default ag value is 8.

-a bf={ **true | false** }

Specifies a large file enabled file system. If you do not need a large file enabled filesystem, set this option to false this is the default. Specifying bf=true requires a fragment size of 4096 and compress=no.

-a nbpi={ **512 | 1024 | 2048 | 4096 | 8192 | 16384 | 32768 | 65536 | 131072** }

Specifies the number of bytes per inode (nbpi). The nbpi affects the total number of inodes on the filesystem. The nbpi value is inversely proportional to the number of inodes on the filesystem. The default nbpi value is 4096 bytes.

3.3.4 mkfs Command

The mkfs command makes a new filesystem on a specified device. The mkfs command initializes the volume label, filesystem label, and startup block.

Similar to the crfs command, a new option has been added to the -o flag for mkfs that allows creation of filesystems capable of containing files greater than 2 GB in size.

-o bf={ **true | false** }

Specifies a large file enabled file system. If you do not need a large file enabled filesystem, set this option to false this is the default. Specifying bf=true requires a fragment size of 4096 and compress=no.

The nbpi option for the -o flag has also been increased.

-o nbpi={ **512 | 1024 | 2048 | 4096 | 8192 | 16384 | 32768 | 65536 | 131072** }

Specifies the number of bytes per inode (nbpi). The nbpi is the ratio of filesystem size in bytes to the total number of inodes. The default nbpi value is 4096 bytes.

The following option has been removed from the mksysb command.

-B Bootable

3.3.5 Creating Large File Enabled Filesystems

Filesystems enabled for large files can be created with the crfs and mkfs commands shown above. Both commands have a new option (bf=true) to specify filesystems enabled for large files. The JFS SMIT menus can also be used to create large file enabled filesystems.

The following examples show creation of a 3 GB Large File Enabled Journaled File System on the rootvg volume group using an nbpi=131072 and ag=64.

From the command line

```
# crfs -v jfs -a bf=true -g'rootvg' -a size='6291456' \
      -m'/3GBFS' -p'yes' -t'no' -A'yes' \
      -a nbpi='131072' -a ag='64'
Based on the parameters chosen, the new /3GBFS JFS file system
is limited to a maximum size of 266338304 (512 byte blocks)

New Filesystem size is 6291456

# lsfs -q /3GBFS
Name          Nodename   Mount Pt   VFS   Size    Options Auto Accounting  Auto Accounting
/dev/lv00     --         /3GBFS     jfs   6291456  rw     yes  no          yes  no
(lv size: 6291456, fs size: 6291456, frag size: 4096, nbpi: 131072,
 compress: no, bf: true, ag: 64)
#
```

This command creates the /3GBFS filesystem on the rootvg volume group with a fragment size of 4096 bytes, a number of bytes per inode (nbpi) ratio of 131072, and an initial size of 3 GB (512 * 6291456).

Using SMIT

```
# smitty jfs
```

-> Add a Journaled File System

```
                          Add a Journaled File System

Move cursor to desired item and press Enter.

  Add a Standard Journaled File System
  Add a Compressed Journaled File System
  Add a Large File Enabled Journaled File System

F1=Help             F2=Refresh          F3=Cancel           F8=Image
```

```
# smitty jfs
```

-> Add a Journaled File System
-> Add a Large File Enabled Journaled File System
-> Volume Group Name

```
                Add a Large File Enabled Journaled File System

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

                                                      [Entry Fields]
  Volume group name                                   rootvg
* SIZE of file system (in 512-byte blocks)           [6291456]          #
* MOUNT POINT                                        [/3GBFS]
  Mount AUTOMATICALLY at system restart?              yes                +
  PERMISSIONS                                         read/write         +
  Mount OPTIONS                                      []                  +
  Start Disk Accounting?                              no                 +
  Number of bytes per inode                           131072             +
  Allocation Group Size (MBytes)                      64                 +

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

3.3.6 JFS Filesystem Size Limitations

AIX 4.2 JFS Filesystems have the following size limitations:

NBPI	Minimum AG Size	Fragment Size	Maximum Size (GB)
512	8	512, 1024, 2048, 4096	8
1024	8	512, 1024, 2048, 4096	16
2048	8	512, 1024, 2048, 4096	32
4096	8	512, 1024, 2048, 4096	64
8192	8	512, 1024, 2048, 4096	128
16384	8	512, 1024, 2048, 4096	128
32768	16	1024, 2048, 4096	128
65536	32	2048, 4096	128
131072	64	4096	128

3.3.7 File Size Limits

The default largest file a particular user's process can create or extend is set in the /etc/security/limits file with the fsize_hard and fsize stanza. The fsize_hard is set to fsize and the fsize is set to 2097151 by default when AIX is first installed.

To be able to create or extend files greater than 2 GB, fsize must be set to greater than 2 GB. fsize must be set within the boundary of fsize_hard.

The ulimit command may also be used to change the fsize up to the maximum value set by the fsize_hard.

Note: Setting fsize to a value of -1 implies *unlimited*.

3.4 Large File Enabled Filesystem Compatibility

Filesystems enabled for large files cannot be migrated to earlier releases of AIX. These releases of AIX do not recognize the superblock version number and will generate a message of: `unknown file system type`.

Note: File systems from Versions 3 and 4.1 of AIX can be migrated forward.

3.4.1 Filesystem Version Numbers

Command to check version numbers :

```
# dumpfs < Filesystem > | grep "file system version"
```

Example :

```
# dumpfs /3GBFS | grep "file system version" |cut -f 3,4,5

        file system version     2
```

Filesystem Version Number

0 fs created with default fragment size of 4096, nbpi=4096, ag=8

1 fs created with other values of fragment size and nbpi with ag=8

Note that the filesystem version number will be 1 if values other than the above defaults are used.

2 fs created with Large File Enabled Journaled File System

3.4.2 Understanding Fragments and Variable Number of inodes

Journaled File System (JFS) fragment support allows disk space to be divided into allocation units that are smaller than the default size of 4096 bytes. Smaller allocation units or *fragments* minimize wasted disk space by more efficiently storing the data in a file or directory's partial logical blocks. The functional behavior of JFS fragment support is based on that provided by Berkeley Software Distribution (BSD) fragment support. Similar to BSD, JFS fragment support allows users to specify the number of inodes that a filesystem contains.

3.4.3 Disk Utilization

Many UNIX filesystems only allocate contiguous disk space in units equal in size to the logical blocks used for the logical division of files and directories. These allocation units are typically referred to as *disk blocks* and a single disk block is used exclusively to store the data contained within a single logical block of a file or directory.

Using a relatively large logical block size (4096 bytes for example) and maintaining disk block allocations that are equal in size to the logical block are advantageous. It reduces the number of disk I/O operations that must be performed by a single filesystem operation since a file or directory's data is stored on disk in a small number of large disk blocks rather than in a large number of small disk blocks. For example, a file with a size of 4096 bytes or less would be allocated a single 4096-byte disk block if the logical block size is 4096 bytes. A read() or write() operation would therefore only have to perform a single disk I/O operation to access the data on the disk. If the logical block size

were smaller requiring more than one allocation for the same amount of data, then more than one disk I/O operation may be required to access the data. A large logical block and equal disk block size is also advantageous for reducing the amount of disk space allocation activity that must be performed as new data is added to files and directories, since large disk blocks hold more data.

Restricting the disk space allocation unit to the logical block size can, however, lead to wasted disk space in a filesystem containing numerous files and directories of a small size. Wasted disk space occurs when a logical block's worth of disk space is allocated to a partial logical block of a file or directory. Since partial logical blocks always contain less than a logical block's worth of data, a partial logical block will only consume a portion of the disk space allocated to it. The remaining portion remains unused since no other file or directory can write its contents to disk space that has already been allocated. The total amount of wasted disk space can become excessive for filesystems containing a large number of small files and directories. A filesystem using 4096-byte allocation units may experience up to 45% wasted disk space. (Statistic taken from *UNIX System Manager's Manual, Computer Systems Research Group, University of California at Berkeley, The Regents of the University of California and/or Bell Telephone Laboratories, 1988, SMM 14*).

3.4.4 Optimizing Disk Utilization

In the JFS however, the disk space allocation unit, referred to as a fragment, can be smaller than the logical block size of 4096 bytes. With the use of fragments smaller than 4096 bytes, the data contained within a partial logical block can be stored more efficiently by using only as many fragments as are required to hold the data. For example, a partial logical block that only has 500 bytes could be allocated a fragment of 512 bytes (assuming a fragment size of 512 bytes), thereby greatly reducing the amount of wasted disk space. If the storage requirements of a partial logical block increase, one or more additional fragments will be allocated.

3.4.5 Fragment Sizes

The fragment size for a filesystem is specified during its creation. The allowed fragment sizes for Journaled File Systems (JFS) are 512, 1024, 2048, and 4096 bytes. For consistency with previous versions of AIX the default fragment size remains 4096 bytes. Different filesystems can have different fragment sizes, but only one fragment size can be used within a single filesystem. Different fragment sizes can also coexist on a single system (machine) so users can select a fragment size most appropriate for each filesystem.

JFS fragment support presents the filesystem as a contiguous series of fragments rather than as a contiguous series of disk blocks. To maintain the efficiency of disk operations however, disk space is often allocated in units of 4096 bytes so the disk blocks or allocation units remain equal in size to the logical blocks. A disk-block allocation in this case can be viewed as an allocation of 4096 bytes of contiguous fragments.

Both operational overhead (additional disk seeks, data transfers, and allocation activity) and better utilization of disk space increase as the fragment size for a filesystem decreases. To maintain the optimum balance between increased overhead and increased usable disk space, the following factors apply to JFS fragment support:

1. Disk space allocations of 4096 bytes of fragments are maintained for a file or directory's logical blocks where possible.
2. Only partial logical blocks for files or directories less than 32 KB in size can be allocated less than 4096 bytes of fragments.

Maintaining 4096-byte disk space allocations allows disk operations to be more efficient as explained previously in 3.4.3, "Disk Utilization" on page 38. As the files and directories within a filesystem grow beyond 32KB in size, the benefit of maintaining disk space allocations of less than 4096 bytes for partial logical blocks diminishes. The disk space savings as a percentage of total filesystem space decreases while the performance cost in maintaining small disk space allocations remains constant. Since disk space allocations of less than 4096 bytes provide the most effective disk space utilization when used with small files and directories, the logical blocks of files and directories equal to or greater than 32KB are always allocated 4096 bytes of fragments. Any partial logical block associated with such a large file or directory is also allocated 4096 bytes of fragments.

3.4.6 Variable Number of inodes (NBPI)

Since fragment support optimizes disk space utilization, it increases the number of small files and directories that can be stored within a filesystem. However, disk space is only one of the filesystem resources required by files and directories. Each file or directory also requires a disk inode. The JFS allows the number of disk inodes created within a filesystem to be specified in case more or fewer than the default number of disk i-nodes is desired. The number of disk inodes can be specified at filesystem creation as the number of bytes per inode (NBPI). For example, an NBPI value of 1024 causes a disk inode to be created for every 1024 bytes of filesystem disk space. Another way to look at this is that a small NBPI value (512 for instance) results in a large number of inodes, while a large NBPI value (such as 16,384) results in a smaller number of inodes. The set of allowable NBPI values vary according to the allocation group size (agsize). The default is 8 MB. The allowed values for agsize are 8, 16, 32, and 64. The range of legal NBPI values scales up as agsize increases. For consistency with previous versions of AIX, the default NBPI value is 4096 and the default agsize is 8.

Note: When enough files have been created to use all the available inodes, no more files can be created, even if the filesystem has free space. The number of available inodes can be determined by using the `df -v` command.

3.4.7 Specifying Fragment Size, NBPI and AG

Fragment size, the number of bytes per inode (NBPI) value and the allocation group size (AG) are specified during filesystem creation with either the `crfs` and `mkfs` commands (See 3.3.3, "crfs Command" on page 34) or by using the System Management Interface (SMIT). The decision of fragment size and how many inodes to create for the filesystem should be based on the projected number number of files contained by the filesystem and their size.

3.4.8 Identifying Fragment Size, NBPI and AG

The filesystem fragment size, the number of bytes per inode (NBPI) and the allocation group (AG) size can be identified through the `lsfs` command or the System Management Interface Tool (SMIT). For application programs, the statfs() subroutine can be used to identify filesystem fragment size.

3.4.8.1 Query Filesystem Information

Command to query filesystem information:

```
# lsfs -q
```

Queries the logical volume manager (LVM) for the logical volume size (in 512-byte blocks) and queries the JFS superblock for the filesystem size, the fragment size, the compression algorithm (if any), and the number of bytes per inode (NBPI)

Example:

```
# lsfs -q /3GBFS

Name            Nodename   Mount Pt               VFS   Size    Options    Auto Accounting
/dev/lv00       --         /3GBFS                 jfs   6291456 --         no   no
(lv size: 6291456, fs size: 6291456, frag size: 4096, nbpi: 4096, compress: no, bf: true, ag: 8)
```

3.4.9 Compatibility and Migration

Previous versions of AIX are compatible with the current JFS, although filesystems with a nondefault fragment size, NBPI value, or allocation group size will require special attention if migrated to a previous version.

3.4.10 Filesystem Images

AIX 4.2 JFS fully supports JFS filesystem images created under previous versions of AIX. These filesystem images and any JFS filesystem image created with the default fragment size, NBPI value of 4096 bytes, and default allocation group (AG) size of 8 can be interchanged with the current and previous versions of AIX without requiring any special migration activities.

JFS filesystem images created with a fragment size or NBPI value or allocation group (AG) size other than the default values may be incompatible with previous versions of AIX. Specifically, only filesystem images less than or equal to 2 GB in size and created with the default parameters can be interchanged among AIX Versions 3.2, 4.1 and 4.2. Filesystem images created with fragment size of either 512, 1024, 2048, or 4096, and an NBPI value of either 512, 1024, 2048, 4096, 8192, or 16384, and an (AG) size of 8 MB can be interchanged among AIX Version 4.1 and AIX Version 4.2. Finally, creating a filesystem with NBPI value greater than 16384 or with an AG size greater than 8 MB will result in a JFS filesystem that is only recognized by AIX Version 4.2.

The following procedure must be used to migrate incompatible filesystems from one version of AIX to another:

1. Backup the filesystem by file name on the source system.
2. Create a filesystem on the destination system.
3. Restore the backed-up files on the destination system.

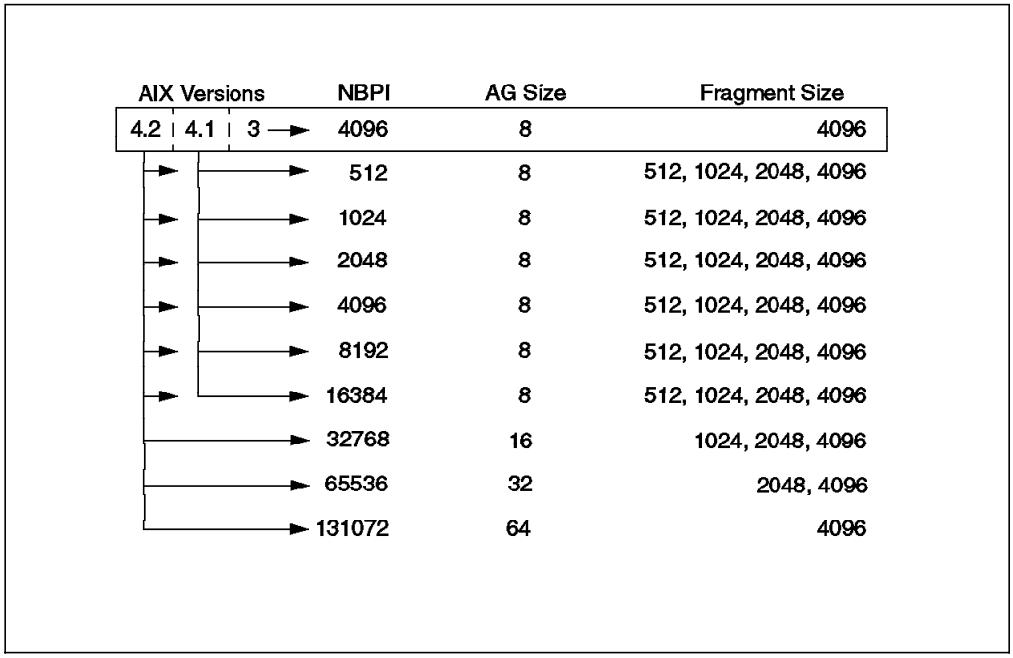


Figure 4. AIX Journaled File System Compatibility

3.4.11 Backup/Restore

Although backup and restore sequences can be performed between filesystems with different fragment sizes and NBPI values, due to increased disk utilization and a different number of inodes, restore operations may fail due to a lack of free fragments or disk inodes if the fragment size or NBPI value of the source filesystem is smaller than the fragment size or NBPI value of the target filesystem. This is of particular importance for full filesystem backup and restore sequences and may even occur when the total filesystem size of the target filesystem is larger than that of the source filesystem.

3.4.12 RAM Disks

Filesystems utilizing fragments with a size other than 4096 cannot be used on a RAM disk. Any valid NBPI value can, however, be specified on a RAM disk filesystem.

3.4.13 Performance Costs of Large File Enabled Filesystems

Though filesystems that use fragments smaller than 4096 bytes as their allocation unit may require substantially less disk space than those using the default allocation unit of 4096 bytes, the use of smaller fragments may incur performance costs.

3.4.14 Increased Allocation Activity

Since disk space is allocated in smaller units for a filesystem with a fragment size other than 4096 bytes, allocation activity may occur more often when files or directories are repeatedly extended in size.

For example, a write operation that extends the size of a zero-length file by 512 bytes results in the allocation of one fragment to the file, assuming a fragment size of 512 bytes. If the file size is extended further by another write() of 512 bytes, an additional fragment must be allocated to the file. Applying this

example to a filesystem with 4096-byte fragments, disk space allocation would occur only once, as part of the first write operation. No additional allocation activity must be performed as part of the second write operation since the initial 4096-byte fragment allocation is large enough to hold the data added by the second write operation.

Allocation activity adds performance overhead to filesystem operations. However, allocation activity can be minimized for filesystems with fragment sizes smaller than 4096 bytes if files are extended by 4096 bytes at a time whenever possible.

3.4.15 Free Space Fragmentation

Using fragments smaller than 4096 bytes may also cause greater fragmentation of the disk′s free space. For example, consider an area of the disk that is divided into eight fragments of 512 bytes each. Suppose that different files, requiring 512 bytes each, have written to the first, fourth, fifth, and seventh fragments in this area of the disk, leaving the second, third, sixth, and eighth fragments free. Although four fragments representing 2048 bytes of disk space are free, no partial logical block requiring four fragments (or 2048 bytes) will be allocated for these free fragments, since the fragments in a single allocation must be contiguous.

Since the fragments allocated for a file or directory′s logical blocks must be contiguous, free space fragmentation may cause a filesystem operation that requests new disk space to fail even though the total amount of available free space is large enough to satisfy the operation. For example, a write operation that extends a zero-length file by one logical block requires 4096 bytes of contiguous disk space to be allocated. If the filesystem free space is fragmented and consists of 32 noncontiguous 512-byte fragments or a total of 16 KB of free disk space, the write operation will fail, since eight contiguous fragments (or 4096 bytes of contiguous disk space) are not available to satisfy the write operation.

A filesystem with an unmanageable amount of fragmented free space can be defragmented with the defragfs command. The execution of defrags has an impact on performance.

3.4.16 Increased Fragment Allocation Map Size

More virtual memory and filesystem disk space may be required to hold fragment allocation maps for filesystems with a fragment size smaller than 4096 bytes. Fragments serve as the basic unit of disk space allocation, and the allocation state of each fragment within a filesystem is recorded in the filesystem′s fragment allocation map.

3.5 Accessing Large Files

The AIX filesystem programming interfaces generally revolve around the off_t data type. In prior releases, the off_t data type was defined as a signed 32-bit integer. As a result, the maximum file size that these interfaces would allow was 2 GB minus 1. For AIX Version 4.2 a new set of programming interfaces has been defined so that application programs can be aware of large files.

3.5.1 Implications for Existing Programs

The 32-bit application environment which all applications used in prior releases remains unchanged. Existing application programs will execute exactly as they did before. However, they will not be able to access large files.

For example, the st_size field in the stat structure which is used to return file sizes, is a signed, 32-bit long. Therefore, that stat structure cannot be used to return file sizes which are larger than LONG_MAX. If an application attempts to stat() a file which is larger than LONG_MAX, the stat() subroutine will fail, and errno will be set to EOVERFLOW, indicating that the file size overflows the size field of the structure being used by the program.

This behavior is significant because existing programs which might not appear to have any impacts as a result of large files will experience failures in the presence of large files even though they may not even be interested in the file size.

The errno EOVERFLOW can also be returned by lseek() and by fcntl() if the values which need to be returned are larger than the data type or structure which the program is using. For lseek(), if the resulting offset is larger than LONG_MAX, lseek() will fail and errno will be set to EOVERFLOW. For fcntl(), if the caller uses F_GETLK and the blocking lock′s starting offset or length is larger than LONG_MAX, the fcntl() call will fail, and errno will be set to EOVERFLOW.

3.5.2 Open Protection for Existing Applications

Many of the existing application programs were written under the assumption that a file size could never be larger than could be represented in a signed, 32-bit long. These programs could have unexpected behavior, including data corruption, if allowed to operate on large files. Beginning with AIX Version 4.2, the operating system implements an open-protection scheme to protect applications from this class of failure.

When an application which has not been enabled for large-file access attempts to open a file which is larger than LONG_MAX, the open() subroutine will fail and errno will be set to EOVERFLOW. Application programs which have not been enabled will be unable to access a large file, and the possibility of inadvertent data corruption is avoided. Applications which need to be able to open large files must be ported to the large-file environment described in section 3.6, “Porting Applications to the Large-File Environment” on page 45.

In addition to open protection, a number of other subroutines offer protection by providing an execution environment which is identical to the environment under which these programs were developed. If an application uses the write() family of subroutines and the write request crosses the 2 GB boundary, the write() subroutines will transfer data only up to 2 GB minus 1. If the application attempts to write() at or beyond the 2 GB minus 1 boundary, the write() subroutines will fail and set errno to EFBIG. The behavior of mmap(), ftruncate(), and fclear() are similar.

The read() family of subroutines also participates in the open protection scheme. If an application attempts to read a file across the 2 GB threshold, only the data up to 2 GB minus 1 will be read. Reads at or beyond the 2 GB minus 1 boundary will fail, and errno will be set to EOVERFLOW.

Open protection is implemented by a flag associated with an open file description. The current state of the flag can be queried with the fcntl() subroutine using the F_GETFL command. The flag can be modified with the fcntl() subroutine using the F_SETFL command.

Since open file descriptors are inherited across the exec() family of subroutines, application programs which pass file descriptors enabled for large-file access to other programs should consider whether the receiving program can safely access the large file.

3.6 Porting Applications to the Large-File Environment

In AIX Version 4.2 the operating system provides two different ways for applications to be enabled for large-file access. Application programmers must decide which approach best suits their needs. The first approach is to define _LARGE_FILES, which carefully redefines all of the relevant data types, structures, and subroutine names to their large-file enabled counterparts. The second approach is to recode the application to call the large-file enabled subroutines explicitly.

Defining _LARGE_FILES has the advantage of maximizing application portability to other platforms since the application is still written to the normal POSIX and XPG interfaces. It has the disadvantage of creating some ambiguity in the code since the size of the various data items is not obvious from looking at the code.

Recoding the application has the obvious disadvantages of requiring more effort and reducing application portability. It can be used when the redefinition effect of _LARGE_FILES would have a considerable negative impact on the program or when it is desirable to convert only a very small portion of the program.

It is very important to understand that in either case, the application program *must* be carefully audited to ensure correct behavior in the new environment. Some of the common programming pitfalls are discussed in 3.7, “Common Programming Pitfalls in the Large-File Environment” on page 47.

3.6.1 Using _LARGE_FILES

In the default compilation environment, the off_t data type is defined as a signed, 32-bit long. If the application defines _LARGE_FILES before the inclusion of any header files, then the large-file programming environment is enabled, and off_t is defined to be a signed 64-bit long long. In addition, all of the subroutines which deal with file sizes or file offsets are redefined to be their large-file enabled counterparts. Similarly, all of the data structures with embedded file sizes or offsets are redefined.

Assuming that the application is coded without any dependencies on off_t being a 32-bit quantity, the resulting binary should work properly in the new environment. In practice, application programs rarely require a porting effort this small.

The following table shows the redefinitions which occur in the _LARGE_FILES environment.

Table 1. Redefinitions which Occur in the _LARGE_FILES Environment

Item	Redefined To Be	Header File
off_t	long long	<sys/types.h>
fpos_t	long long	<sys/types.h>
struct stat	struct stat64	<sys/stat.h>
stat()	stat64()	<sys/stat.h>
fstat()	fstat64()	<sys/stat.h>
lstat()	lstat64()	<sys/stat.h>
mmap()	mmap64()	<sys/mman.h>
lockf()	lockf64()	<sys/lockf.h>
struct flock	struct flock64	<sys/flock.h>
open()	open64()	<fcntl.h>
creat()	creat64()	<fcntl.h>
F_GETLK	F_GETLK64	<fcntl.h>
F_SETLK	F_SETLK64	<fcntl.h>
F_SETLKW	F_SETLKW64	<fcntl.h>
ftw()	ftw64()	<ftw.h>
nftw()	nftw64()	<ftw.h>
fseeko()	fseeko64()	<stdio.h>
ftello()	ftello64()	<stdio.h>
fgetpos()	fgetpos64()	<stdio.h>
fsetpos()	fsetpos64()	<stdio.h>
fopen()	fopen64()	<stdio.h>
freopen()	freopen64()	<stdio.h>
lseek()	lseek64()	<unistd.h>
ftruncate()	ftruncate64()	<unistd.h>
truncate()	truncate64()	<unistd.h>
fclear()	fclear64()	<unistd.h>

3.6.2 Using the 64-Bit File System Subroutines

Using the _LARGE_FILES environment may be impractical for some applications due to the far-reaching implications of changing the size of off_t to 64 bits. If the number of changes is small, it may be more practical to convert a relatively small part of the application to be large-file enabled. The 64-bit filesystem data types, structures, and subroutines are listed below:

```
<sys/types.h>
typedef long long off64_t;
typedef long long fpos64_t;

<fcntl.h>
extern int open64(const char *, int, ...);
extern int creat64(const char *, mode_t);
#define F_GETLK64
#define F_SETLK64
#define F_SETLKW64

<ftw.h>
extern int ftw64(const char *, int (*)(const char *,const struct stat64 *, int),
                 int);
extern int nftw64(const char *, int (*)(const char *, const struct stat64 *,
                 int, struct FTW *), int, int);

<stdio.h>
extern int fgetpos64(FILE *, fpos64_t *);
extern FILE *fopen64(const char *, const char *);
extern FILE *freopen64(const char *, const char *, FILE *);
extern int fseeko64(FILE *, off64_t, int);
extern int fsetpos64(FILE *, fpos64_t *);
extern off64_t  ftello64(FILE *);

<unistd.h>
extern off64_t lseek64(int, off64_t, int);
extern int ftruncate64(int, off64_t);
extern int truncate64(const char *, off64_t);
extern off64_t  fclear64(int, off64_t);

<sys/flock.h>
struct flock64;

<sys/lockf.h>
extern int lockf64 (int, int, off64_t);

<sys/mman.h>
extern void *mmap64(void *, size_t, int, int, int, off64_t);

<sys/stat.h>
struct stat64;
extern int stat64(const char *, struct stat64 *);
extern int fstat64(int, struct stat64 *);
extern int lstat64(const char *, struct stat64 *);
```

3.7 Common Programming Pitfalls in the Large-File Environment

Porting of application programs to the large-file environment can expose a number of different problems in the application. These problems are frequently the result of poor coding practices which are harmless in a 32-bit off_t environment, but which can manifest themselves when compiled in a 64-bit off_t environment. The information below illustrates some of the more common problems and solutions.

Note: In the examples below, off_t is assumed to be a 64-bit file offset.

3.7.1 Improper Use of Data Types

The most obvious source of problems with application programs is a failure to use the proper data types. If an application attempts to store file sizes or file offsets in an integer variable, the resulting value will be truncated and lose significance. The proper technique for avoiding this problem is to use the off_t data type to store file sizes and offsets, as shown here:

Incorrect:

```
int file_size;
struct stat s;
file_size = s.st_size;
```

Better:

```
off_t file_size;
struct stat s;
file_size = s.st_size;
```

3.7.2 Parameter Mismatches

Care must be taken when passing 64-bit integers to functions as arguments or when returning 64-bit integers from functions. Both the caller and the called function must agree on the types of the arguments and the return value in order to get correct results.

Passing a 32-bit integer to a function which expects a 64-bit integer causes the called function to misinterpret the caller's arguments, leading to unexpected behavior. This type of problem is especially severe if the program passes scalar values to a function which expects to receive a 64-bit integer.

Many of the problems can be avoided by careful use of function prototypes as illustrated below. In the code fragments below, fexample() is a function which takes a 64-bit file offset as a parameter. In the first example, the compiler generates the normal 32-bit integer function linkage, which would be incorrect since the receiving function expects 64-bit integer linkage. In the second example, the LL specifier is added, forcing the compiler to use the proper linkage. In the last example, the function prototype causes the compiler to promote the scalar value to a 64-bit integer. This is the preferred approach since the source code remains portable between 32 and 64-bit environments.

Incorrect:

```
fexample(0);
```

Better:

```
fexample(0LL);
```

Best:

```
void fexample(off_t);
fexample(0);
```

3.7.3 Arithmetic Overflows

Even when an application uses the correct data types, it is still vulnerable to failures due to arithmetic overflows. This problem usually occurs when the application performs an arithmetic overflow before it is promoted to the 64-bit data type. In the following example, blkno is a 32-bit block number. Multiplying the block number by the block size occurs before the promotion, and overflow will occur if the block number is sufficiently large. This problem is especially destructive because the code is using the proper data types and the code works

properly for small values, but fails for large values. The problem can be fixed by typecasting the values before the arithmetic operation, as shown here:

Incorrect:
```
int blkno;
off_t offset;

offset = blkno * BLKSIZE;
```

Better:
```
int blkno;
off_t offset;
offset = (off_t) blkno * BLKSIZE;
```

This problem can also appear when passing values based on fixed constants to functions which expect 64-bit parameters. In the example below, LONG_MAX+1 results in a negative number, which is sign-extended when it is passed to the function

Incorrect:
```
void fexample(off_t);
fexample(LONG_MAX+1);
```

Better:
```
void fexample(off_t);
fexample((off_t)LONG_MAX+1);
```

3.7.4 fseek() and ftell()

The data type used by fseek() and ftell() subroutines is *long* and cannot be redefined to the appropriate 64-bit data type in the _LARGE_FILES environment. Application programs which access large files and which use fseek() and ftell() need to be converted. This can be done in a number of ways. The fseeko() and ftello() subroutines are functionally equivalent to fseek() and ftell() except that the offset is given as an off_t instead of a long. Make sure to convert all variables which can be used to store offsets to the appropriate type.

Incorrect:
```
long cur_offset, new_offset;

cur_offset = ftell(fp);
fseek(fp, new_offset, SEEK_SET);
```

Better:
```
off_t cur_offset, new_offset;

cur_offset = ftello(fp);
fseeko(fp, new_offset, SEEK_SET);
```

3.7.5 Failure to Include Proper Header Files

In order for application programs to see the function and data type redefinitions, they must include the proper header files. This has the additional benefit of exposing the function prototypes for various subroutines which enables stronger type-checking in the compiler.

Many application programs which call the open and creat subroutines do not #include <fcntl.h> which contains the defines for the various open modes. These programs typically hard code the open modes. This will cause runtime failures when the program is compiled in the _LARGE_FILES environment

because the program does call the proper open subroutine, and the resulting file descriptor is not enabled for large-file access. Programs must make sure to include the proper header files, especially in the _LARGE_FILES environment, to get visiblity to the redefinitions of the environment.

Incorrect:

```
fd = open("afile",2);
```

Better:

```
#include <fcntl.h>

fd = open("afile",O_RDWR);
```

3.7.6 String Conversions

Converting file sizes and offsets to and from strings can cause problems when porting applications to the large-file environment. The printf() format string for a 64-bit integer is different to that for a 32 bit integer. Programs which do these conversions must be careful to use the proper format specifier. This is especially difficult when the application needs to be portable between 32-bit and 64-bit environments since there is no portable format specifier between the two environments. One way to deal with this problem is to write offset converters which use the proper format for the size of off_t, as shown here:

```
off_t
atooff(const char *s)
{
        off_t o;


        if (sizeof(off_t) == 4)
                sscanf(s,"%d",&o);
        else if (sizeof(off_t) == 8)
                sscanf(s,"%lld",&o);
        else
                error();
        return o;
}
main(int argc, char **argv)
{
        off_t offset;
        offset = atooff(argv[1]);
        fexample(offset);
}
```

3.7.7 Imbedded File Offsets

Application programs which imbed file offsets or sizes in data structures may be affected by the change to the size of the off_t in the large-file environment. This problem can be especially severe if the data structure is shared between various applications or if the data structure is written into a file. In cases like this, the programmer must decide if it should continue to contain a 32-bit offset or if it should be converted to contain a 64-bit offset. If the application program needs to have a 32-bit file offset even if off_t is 64 bits, the program may use the new data type soff_t, a short off_t. This data type remains 32 bits even in the large-file environment. If the data structure is converted to a 64-bit offset, then all of the programs which deal with that structure must be converted to understand the new data structure format.

3.7.8 File Size Resource Limit

Application programs which are converted to be aware of large files may fail in their attempts to create large files due to the file-size resource limit. The file-size resource limit is a signed, 32-bit value which limits the maximum file offset to which a process can write to a regular file. Programs which need to write large files must have their file size limit set to RLIM_INFINITY as shown here:

```
struct rlimit r;

r.rlim_cur = r.rlim_max = RLIM_INFINITY;
setrlimit(RLIMIT_FSIZE,&r);
```

This ulimit may also be set from the Korn shell by issuing the command:

```
#ulimit -f unlimited
```

3.7.9 JFS Maximum File Size

The maximum size of a file is ultimately a characteristic of the filesystem itself, not just the file size limit or the environment. For the JFS, the maximum file size is determined by the parameters used at the time the filesystem was created. For JFS filesystems which are enabled for large files, the maximum file size is slightly less than 64 gigabytes (0xff8400000). For all other JFS filesystems, the maximum file size is 2 GB minus 1 (0x7fffffff). Attempts to write to a file past the maximum file size in any filesystem format will fail, and errno will be set to EFBIG.

3.8 Command Support for Files Larger than 2 GB

AIX 4.2 or later provides support for files greater than 2 GB so that users can store large quantities of data in a single file. Many, but not all, AIX commands support the use of files larger than 2 GB. Additionally, some commands have large-file support with limitations.

3.8.1 Commands that do not Support Files Larger than 2 GB

In many cases, commands that do not support large files do not utilize files of any size to begin with, such as the `date`, `echo`, `nice`, `kill` commands and others.

This support also does not extend to specific system-controlled files, such as /etc/passwd, /etc/inittab, files in /etc/security, system accounting files, and others. Consequently, commands that only utilize these system files, such as commands to administer users and system security (`mkuser`, `su`), system accounting commands (`acctcom`, `prdaily`), and general system controlling commands (`init`, `penable`) do not have large-file support.

Other commands do not support large files because they work with files of a specific format defined to have a maximum of less than or equal to 2 GB. These include the XCOFF file format, defining the format of object files and executable files. The file headers that define XCOFF do not have fields defined to support files this large, and the system would not be able to load an executable file of this size. Commands that utilize these files, such as `ld`, `as`, `m4`, `strip` and so on, do not have large-file support.

The header format of the `pack`, `unpack`, and `pcat` commands does not have enough characters to store a file size over 2 GB.

Additional file formats also prevent files of their type from being larger than 2 GB. These include some archiving utilities restricted in format by industry standards, such as the `cpio`, `pax`, and `tar` (you can archive large files with `backup`) commands, and the object file archive format, restricting the `ar` command.

The AIX print spooling subsystem has been enabled on the frontend to support the submission, manipulation, and cancelation of files larger than 2 GB. However, the default AIX printer backend, the `piobe` command does not support files of this size. This means print jobs larger than 2 GB can either be sent to a remote printer or print server that can handle these large files, or an alternate user or vendor-supplied backend, that can support large files, can be used.

Note: A print job larger than 2 GB would probably take several days to complete.

Finally, there are commands for which the user files used are not reasonably expected to ever be larger than 2 GB. For example, although a directory may contain large files, the directory file itself may not exceed 2 GB. Hence, commands such as `mkdir` and `rmdir` do not support large directories. Other examples in which support is unnecessary would be using the `wall` command to broadcast the contents of extremely large files to all terminals, or using the `nroff` command to process over 2 GB of written text in a single file.

3.8.2 Commands that Support Files Larger than 2 GB

The following commands all support files larger than 2 GB. Commands which do not appear on the list do not support large files. Commands with limited large file support are marked with an asterisk (*) and an explanation of their limitations follow the list.

Table 2. AIX Command Support for Files Larger than 2 GB

aclget	auditcat *	auditconv *	auditselect *
awk *	backbyinode	backbyname	backup
bdiff	bsh *	cancel *	cat
chgrp	chmod	chown	cksum
cmp	comm	compress	cp
csh *	csplit	ctags	cut
dd	del	devnm	di
du	egrep	enq *	expand
fgrep	file	find *	fold
grep	head	iconv	install
join	ksh *	li	link
ln	lp *	lpd *	lpq *
lpr *	lprm *	lpstat *	ls
make *	move	mv	nawk *
newform	nl	nohup	od
paste	patch	pr	proto
qcan *	qchk *	qdaemon *	qpri *
qprt *	qstatus *	rdist *	rdump
rembak *	restbyinode	restbyname	restore
rev	rm	rrestore	Rsh *
sed	sort	split	strings
sum	tab	tail	tee
test	touch	tr	trbsd
tsh *	uncompress	unexpand	uniq
unlink	untab	update	virscan
wc	whereis	which	zcat

3.8.3 Limitations

The printer commands support files larger than 2 GB on the printer frontend only. The default AIX printer backend, the `piobe` command, does not support files of this size. This applies to the following commands:

```
cancel  lpq     qcan     qprt
enq     lpr     qchk     qstatus
lp      lprm    qdaemon  rembak
lpd     lpstat  qpri
```

The shells support I/O redirection of files that are larger than 2 GB. No other support for files larger than 2 GB is offered in the shells. This applies to the commands bsh, csh, ksh, Rsh, and tsh.

Note: The sh command is a link to the ksh command.

The awk and nawk commands are able to handle data files larger than 2 GB. However, awk and nawk scripts themselves may not be this large.

The find command will process files larger than 2 GB, but it will not allow the use of the -size Number flag where Number is larger than 2 GB.

The make command will operate with targets and dependencies that are larger than 2 GB, but a makefile itself may not be this large.

The audit commands auditcat, auditconv, and auditselect support trail files that are larger than 2 GB, but they do not support binary files larger than 2 GB.

Warning

Do not attempt to send a large file to a pre-AIX 4.2 or non-AIX machine with the rdist command. Doing so will result in undefined behavior and in rare cases, loss of data.

3.9 Big Executables

A big executable is simply an executable program that is larger than a single segment (256 MB) in size. In AIX V3 and AIX V4.1, the executable files are addresses in a single virtual memory segment. This segment is commonly referred to as the TEXT segment. In order to be executable, the text, data and loader sections of the executable file must reside in the first 256 MB of the file. If a program contains a large amount of text or initialized data, then AIX is unable to execute the program. A large data model does exist which was designed to accommodate programs which require a large amount of uninitialized data, or a large heap but not programs with large amounts of initialized data (data section).

In AIX 4.2, we are able to execute programs which are larger than 256 MB in size. However, due to the constraints of AIX′s segmented address space, the following restrictions are placed on the big executable:

- The offset of the text section plus the size of the loader section must be less than 256 MB. Note that the alignment of the loader must be taken into account when making this calculation.

In the big executable design, the text and loader section of an executable will be mapped into the TEXT segment of the process′s address space. The DATA section of the executable will be mapped starting at the BDATASEG(3) segment of the process′s address space.

This design allows the execution of programs with a very large amount of initialized data. The system loader will perform all the necessary mappings to transform the big executable file into a usable process image.

3.9.1 Mapping the Big Executable at Execution Time

At exec time, the system loader is given a file pointer and is responsible for loading that file into the process address space. The loader always attempts to map the file by calling the fp_shmat() subroutine. The fp_shmat() subroutine will return the segment id of the persistent segment associated with the file.

After a successful call to fp_shmat(), the loader will stat the file to obtain its size. If the size of the file is greater than 256 MB, then a new routine called map_big_exec() is invoked to map the executable file into the process address space. Otherwise, exec processing proceeds down the normal path.

The map_big_exec() routine is responsible for mapping the big executable file into the process address space. The first thing this routine does is determine if all the important executable file sections (text, data, loader) are contained in the first segment (256 MB) of the file. It is possible for an executable file to be larger than 256 MB in size, and yet all important data in the file be contained in the first 256 MB. In this case map_big_exec() routine returns and lets the normal exec logic proceed.

Once the map_big_exec() routine has determined that it is dealing with a big executable, it searches the list of currently running big executables to see if it can reuse an existing text segment. If multiple instances of the same big executable program are running at the same time, they can all share the same text segment. The loader is now required to maintain a list of all running big executables. A big executable can be identified by the address of the gnode

associated with the executable file. Therefore, to determine if an instance of a big executable is already running, a search of the big executable list is performed while doing gnode comparisons.

If the map_big_exec() routine does not find an existing instance of the big executable, then it builds a working store segment that serves as the process's text segment.

In order to execute a big executable, the program must specify use of the large data model. Use of the large data model is required so the user can specify the upper limits of the programs data. If an attempt is made to execute a big executable which has not specified the large data model, an error will be returned. Also, the data and bss sections must fit in the space specified by the o_maxdata field.

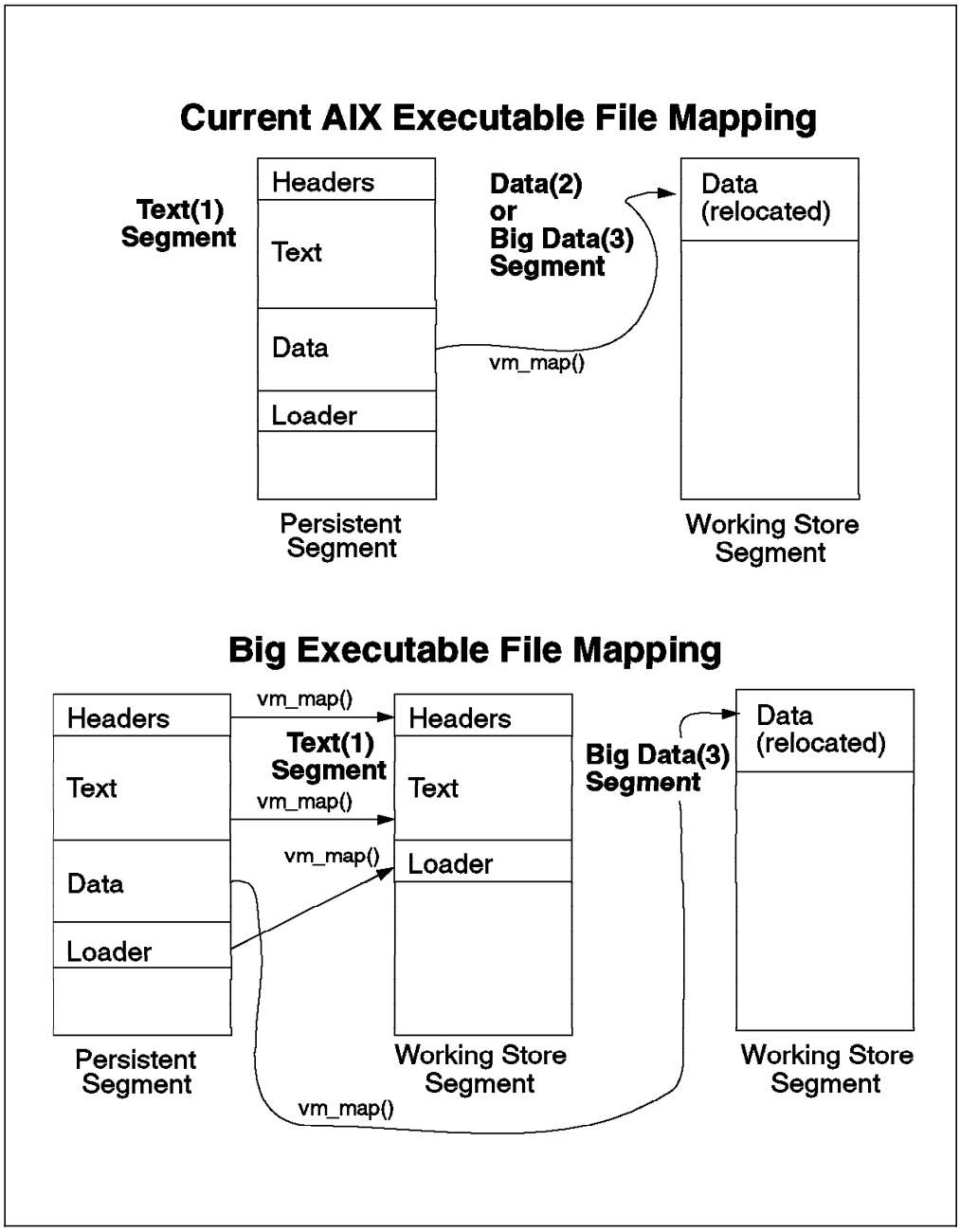


Figure 5. Big Executables Mapping Comparisons

3.9.2 Large Program Support Overview

Some programs need larger data areas than allowed by the default address-space model. Application programs which require a larger data area must use the large address-space model.

The system hardware divides the currently active 32-bit virtual address space into 16 independent segments, each addressed by a separate segment register. The operating system refers to segment 2 (virtual address 0x20000000) as the process private segment. This segment contains most of the per-process information, including user data, user stack, kernel stack, and user block.

Because the system places user data and the user stack within a single segment, the system limits the maximum amount of stack and data to slightly less than 256MB. This size is adequate for most applications. The kernel stack and u-block are relatively small and of fixed size. However, certain applications require large initialized or uninitialized data areas in the data section of a program. Other large data areas can be created dynamically with the malloc(), brk() or sbrk() subroutines.

Programs that need even larger data areas can use the large address-space model to request the necessary amount of data space.

3.9.3 Understanding the Large Address-Space Model

The large address-space model enables large data applications while allowing programs that use a smaller space to follow the smaller model. To allow a program to use the large address-space model, you must set the o_maxdata field in the XCOFF header of the program to indicate the amount of data needed.

In the large address-space model, the data in the program is laid out beginning in segment 3 when the value is non-zero. (The data is laid out beginning in segment 3, even if the value is smaller than a segment size). The program consumes as many segments as needed to hold the amount of data indicated by the o_maxdata field, up to a maximum of eight segments. The program can, therefore, have up to 2 GB of data.

Other aspects of the program address space remain unchanged. The user stack, kernel stack, and u-block continue to reside in segment 2. Also, the data resulting from loading a private copy of a shared library is placed in segment 2. Only program data is placed in segment 3 or higher.

As a result of this organizational scheme, the user stack is still limited by the size of segment 2. (However, the user stack can be relocated into a shared memory segment). In addition, fewer segments are available for mapped files.

While the size of initialized data in a program can be large, there is still a restriction on the size and placement of text. In the executable file associated with a program, the offset of the end of the text section plus the size of the loader section must be less than 256 MB. This is required so this read-only portion of the executable will fit into segment 1 (the TEXT segment). Because of these restrictions, a program cannot have a very large text section.

3.9.4 Enabling the Large Address-Space Model

To enable the large address-space model, use the -bmaxdata flag with the cc or ld command. For example, to link a program that will have the maximum eight segments reserved to it, the following command line could be used:

```
cc sample.o -bmaxdata:0x80000000
```

The number 0x80000000 is the number of bytes, in hexadecimal format, equal to eight 256 MB segments. Although larger numbers can be used, they are ignored because a maximum of eight segments can be reserved. The value following the -bmaxdata flag can also be specified in decimal or octal format.

The large address space model is used if any nonzero value is given for the maxdata keyword. Use the -bmaxdata option only if the program needs very large data areas.

Using the following shell commands, you can patch large programs to use large data without relinking them:

```
/usr/bin/echo '\0200\0\0\0'|dd of=executable_file_name bs=4 count=1 \
seek=19 conv=notrunc
```

Note: Use the full name of the echo command (/usr/bin/echo) to avoid invoking any of the shell echo subcommands by mistake.

The echo string generates the binary value 0x80000000. The dd command seeks to the proper offset in the executable file and modifies the o_maxdata field. Do not use the dd command on nonexecutable object files, loadable modules, or shared libraries.

3.9.5 Executing Programs with Large Data Areas

When a program attempts to execute a program with large data areas, the system recognizes the requirement for large data and attempts to modify the soft limit on data size to accommodate that requirement. However, if it does not have permission to modify the soft limit, the program ends.

In addition, it is also possible that the data size specified in the o_maxdata field may be too small to accommodate the amount of space required for initialized or uninitialized data. In this case, the process ends, and an error is reported.

The attempt is also unsuccessful if the new soft limit is above the hard limit for the process. For example, the login process usually sets the hard limit to infinity. However, if the calling process has modified its hard limit using either the ulimit command in the Bourne shell or the limit command in the C shell, the newly modified soft limit may be above the hard limit for the process. In this case, the process will be killed during exec processing. In this situation, the only message you receive is killed which informs you that the process was killed.

For more information on the ulimit command in the Bourne shell, see *Bourne Shell Special Commands in AIX Version 4 System User's Guide: Operating System and Devices*. For more information about the limit command in the C shell, see *Command Substitution in the C Shell and Filename Substitution in the C Shell* in the same publication.

After placing the program's initialized and uninitialized data in segment 3 and beyond, the system computes the break value. The break value defines the end

of the process's static data and the beginning of its dynamically allocatable data. Using the malloc(), brk() or sbrk() subroutines, the process is free to move the break value toward the end of the segment identified by the maxdata field in the a.out header file.

For example, if the value specified in the maxdata field in the a.out header file is 0x80000000, then the maximum break value is up to the end of segment 10 or 0xafffffff. The brk() subroutine extends the break across segment boundaries, but not beyond the point specified in the maxdata field.

The majority of subroutines are unaffected by large data programs. The semantics of the fork subroutine remain unchanged. Large data programs can run other large or small programs, as well as load and unload other modules.

The setrlimit() subroutine allows the soft data limit to be set to any value that does not exceed the hard limit. However, because of the inherent limitation of the address space model used by the process, it may not be able to increase its size to the value that is set.

3.9.6 Special Considerations

Programs with large data spaces require a large amount of paging space. For example, if a program with a 2 GB address space tries to access every page in its address space, the system must have 2 GB of paging space. The operating system page-space monitor terminates processes when paging space runs low. Programs with large data spaces are terminated first because they typically consume a large amount of paging space.

Debugging programs with large data is similar to debugging other programs. The dbx command can debug these large programs actively or from a core dump. A full core dump should not be performed because programs with large data areas produce large core dumps, which consume large amounts of filesystem space.

Some application programs may be written in such a way that they rely on characteristics of the address-space model. Programs in which the large address-space is enabled use a different address-space model than programs without the large address-space enabled. This could cause problems for applications which make assumptions about the address-space model they are running in. In general, it is not a good idea for an application program to make any assumptions about the address-space model.

3.10 Full ulimit Control for Administrators

In previous versions of AIX, system administrators were only allowed to set the soft `ulimit` values of the user resources which the user could then change using the `ulimit` command. The hard `ulimit` values were set to some undocumented values.

In AIX V4.2, the system administrator is able to specify both the hard and soft `ulimits` of user resources. With the `ulimit` command, users can change their soft `ulimit` values but not beyond the maximum set by the hard `ulimit` values.

Note: Setting the default limits in the /etc/security/limits file sets system wide limits, not just limits taken on by a user when that user is created.

3.10.1 Limits

The /etc/security/limits file defines process resource limits for users. This file is an ASCII file that contains stanzas that specify the process resource limits for each user. These limits are set by individual attributes within a stanza.

Each stanza is identified by a user name followed by a colon, and contains attributes in the Attribute=Value form. Each attribute is ended by a new-line character, and each stanza is ended by an additional new-line character. If you do not define an attribute for a user, the system applies default values.

If the hard values are not explicitly defined in the /etc/security/limits file but the soft values are, the system substitutes the following values for the hard limits:

Resource	**Hard Value**
File Size	fsize
Core Size	unlimited
CPU Time	cpu
Data Size	unlimited
Memory Size	unlimited
Stack Size	unlimited

Note: Use a value of -1 to set a resource to unlimited.

If the hard values are explicitly defined but the soft values are not, the system sets the soft values equal to the hard values.

You can set the following limits on a user:

fsize Identifies the soft limit for the largest file a user's process can create or extend. The minimum value for this attribute is 8192.

core Specifies the soft limit for the largest core file a user's process can create.

cpu Sets the soft limit for the largest amount of system unit time (in seconds) that a user's process can use. The default value is -1 which turns off restrictions.

data Identifies the soft limit for the largest process data segment for a user's process. The minimum allowable value for this attribute is 1272.

stack Specifies the soft limit for the largest process stack segment for a user's process. The minimum allowable value for this attribute is 49.

rss Sets the soft limit for the largest amount of physical memory a user's process can allocate. This limit is not enforced by the system.

fsize_hard Identifies the largest file a user's process can create or extend. The minimum value for this attribute is 8192.

core_hard Specifies the largest core file a user's process can create.

cpu_hard Sets the largest amount of system unit time (in seconds) that a user's process can use. The default value is -1 which turns off restrictions.

data_hard Identifies the largest process data segment for a user's process. The minimum allowable value for this attribute is 1272.

stack_hard Specifies the largest process stack segment for a user's process. The minimum allowable value for this attribute is 49.

rss_hard Sets the largest amount of physical memory a user's process can allocate. This limit is not enforced by the system.

Except for the cpu attribute, each attribute must be a decimal integer string representing the number of 512-byte blocks allotted to the user. The cpu attribute is a decimal integer string representing the amount of system unit time in seconds. For an example of a limits file stanza, see 3.10.2, "Examples" on page 62.

When you create a user with the mkuser command, the system adds a stanza for the user to the limits file. Once the stanza exists, you can use the chuser command to change the user's limits. To display the current limits for a user, use the lsuser command. To remove users and their stanzas, use the rmuser command.

Note: Access to the user database files should be through the system commands and subroutines defined for this purpose. Access through other commands or subroutines may not be supported in future releases. The /etc/security/limits file should have read access for the root user and members of the security group, and write access for the root user only. Access for other users and groups depends upon the security policy for the system.

3.10.2 Examples

An example of records in the /etc/security/limits file:

```
* Sizes are in multiples of 512 byte blocks, CPU time is in seconds
* fsize      - soft file size in blocks
* core       - soft core file size in blocks
* cpu        - soft per process CPU time limit in seconds
* data       - soft data segment size in blocks
* stack      - soft stack segment size in blocks
* rss        - soft real memory usage in blocks
* fsize_hard - hard file size in blocks
* core_hard  - hard core file size in blocks
* cpu_hard   - hard per process CPU time limit in seconds
* data_hard  - hard data segment size in blocks
* stack_hard - hard stack segment size in blocks
* rss_hard   - hard real memory usage in blocks
*
* The following table contains the default hard values if the
* hard values are not explicitly defined:
*
*   Attribute        Value
*   ==========    ============
*   fsize_hard    set to fsize
*   cpu_hard      set to cpu
*   core_hard         -1
*   data_hard         -1
*   stack_hard        -1
*   rss_hard          -1
*
* NOTE:  A value of -1 implies "unlimited"
default:
        fsize = 2097151
        core = 2048
        cpu = -1
        data = 262144
        rss = 65536
        stack = 65536

root:

daemon:

bin:

sys:

adm:

uucp:

guest:

nobody:

lpd:

aix42sm:
        fsize = 8192
        cpu = 3600
        data = 1272
        stack = 1024
        core = 4096
```

SMIT modifications:

```
                    Change / Show Characteristics of a User

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

                                                      [Entry Fields]
* User NAME                                           aix42sm
  User ID                                            [204]                #
  ADMINISTRATIVE USER?                                false               +
  Primary GROUP                                      [staff]              +
  Group SET                                          [staff]              +
  ADMINISTRATIVE GROUPS                              []                   +
  Another user can SU TO USER?                        true                +
  SU GROUPS                                          [ALL]                +
  HOME directory                                     [home/aix42sm]
  Initial PROGRAM                                    [usr/bin/ksh]
  User INFORMATION                                   []
  EXPIRATION date (MMDDhhmmyy)                       [0]
  Is this user ACCOUNT LOCKED?                        false               +
  User can LOGIN?                                     true                +
  User can LOGIN REMOTELY?                            true                +
  Allowed LOGIN TIMES                                []
  Number of FAILED LOGINS before                     [0]                  #
       user account is locked
  Login AUTHENTICATION GRAMMAR                       [compat]
  Valid TTYs                                         [ALL]
  Days to WARN USER before password expires          [0]                  #
  Password CHECK METHODS                             []
  Password DICTIONARY FILES                          []
  NUMBER OF PASSWORDS before reuse                   [0]                  #
  WEEKS before password reuse                        [0]                  #
  Weeks between password EXPIRATION and LOCKOUT      [1]
  Password MAX. AGE                                  [0]                  #
  Password MIN. AGE                                  [0]                  #
  Password MIN. LENGTH                               [0]                  #
  Password MIN. ALPHA characters                     [0]                  #
  Password MIN. OTHER characters                     [0]                  #
  Password MAX. REPEATED characters                  [8]                  #
  Password MIN. DIFFERENT characters                 [0]                  #
  Password REGISTRY                                  []
  Soft FILE size                                     [8192]               #
  Soft CPU time                                      [3600]
  Soft DATA segment                                  [1272]               #
  Soft STACK size                                    [1024]               #
  Soft CORE file size                                [4096]               #
  Hard FILE size                                     []                   #
  Hard CPU time                                      []
  Hard DATA segment                                  []                   #
  Hard STACK size                                    []                   #
  Hard CORE file size                                []                   #
  File creation UMASK                                [22]
  AUDIT classes                                      []                   +
  TRUSTED PATH?                                       nosak               +
  PRIMARY authentication method                      [SYSTEM]
  SECONDARY authentication method                    [NONE]



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

3.11 mksysb and savevg Support for Striped Logical Volumes

In previous versions of AIX, when restoring from a mksysb/savevg media, striped logical volumes in the volume group were not recreated as striped.

In AIX V4.2, support for striped logical volumes in mksysb/savevg has been added. When a backup is created using the mksysb/savevg command, the mkszfile command is called to record the current attributes of each logical volume in the lv_data stanzas of the image.data file. There is an lv_data stanza for each logical volume in the volume group. The additional stanzas that are recorded are **STRIPE_WIDTH** and **STRIPE_SIZE** which are used for recreating striped logical volumes.

During BosInstall or restvg operation, the logical volume attributes are read from the image.data file. The logical volumes are then recreated using these attributes.

3.11.1 lv_data stanza in the image.data file for a Striped Logical Volume

:

```
lv_data:
        VOLUME_GROUP= rootvg
        LV_SOURCE_DISK_LIST= hdisk0 hdisk1 hdisk2
        LV_IDENTIFIER= 00fffffb84907a4.9
        LOGICAL_VOLUME= stripelv
        VG_STAT= active/complete
        TYPE= jfs
        MAX_LPS= 1024
        COPIES= 1
        LPs= 900
        STALE_PPs= 0
        INTER_POLICY= maximum
        INTRA_POLICY= middle
        MOUNT_POINT= /stripefs
        MIRROR_WRITE_CONSISTENCY= on
        LV_SEPARATE_PV= yes
        PERMISSION= read/write
        LV_STATE= opened/syncd
        WRITE_VERIFY= off
        PP_SIZE= 4
        SCHED_POLICY= striped
        PP= 900
        BB_POLICY= relocatable
        RELOCATABLE= no
        UPPER_BOUND= 3
        LABEL= /stripefs
        MAPFILE=
        LV_MIN_LPS= 4
        STRIPE_WIDTH= 3
        STRIPE_SIZE= 128K
```

3.11.2 Support for Large File Enabled Journaled Filesystems and AG Size

In AIX V4.2 the mkszfile command also records information for journaled filesystems that are Large File Enabled. The **FS_BF** and the **FS_AGSIZE** attributes have been introduced in this AIX version.

During BosInstall or restvg operation, the filesystem attributes are read from the image.data file. The filesystems are recreated using these attributes.

3.11.3 fs_data stanza in the image.data file for a Large File Enabled JFS

```
fs_data:
        FS_NAME= /stripefs
        FS_SIZE= 7372800
        FS_MIN_SIZE= 29872
        FS_LV= /dev/stripelv
        FS_FS= 4096
        FS_NBPI= 32768
        FS_COMPRESS= no
        FS_BF= true
        FS_AGSIZE= 64
```

3.11.4 Support for Large Files

In AIX V4.2 the mksysb and savevg commands use the dd command for the system boot image and backup for the rootvg and rootvg/othervg images respectively. Both commands support files greater than 2 GB.

3.11.5 New mksysb Options

mksysb creates an installable image of the root volume group either in a file or onto a bootable tape.

Note: The mksysb command will create a bootable tape on an ISA-bus machine but *not* every ISA-bus machine will boot from tape. To determine if your system will boot from tape, run the following command:

```
bootinfo -e
```

If a 1 is returned, it will boot from tape, a 0 means it will not boot from tape, and you must boot from CD-ROM to install your mksysb tape.

Two new options have been added to enhance the functionality of the mksysb command in AIX Version 4.2. They are the -v option which will display progress messages during each of the stages and the -p option which gives you the option of packing files.

Usage:

mksysb [-X] [-i] [-m] [-e] [-b blocks] [-p] device

-X Expand /tmp if needed.

-i Create the image.data file.

-m Create the image.data file and physical partition maps.

-e Exclude the files/directories.

-v **List files as they are backed up.**

-p **Do not pack files as they are backed up.**

-b blocks Number of 512-byte blocks to write in a single output operation.

device Name of device to receive the backup information. Example: *mksysb* /dev/rmt0

3.11.6 SMIT mksysb

The two new options have also been added to the SMIT panels for mksysb.

```
# smitty mksysb
```

```
                                  Back Up the System

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

                                                        [Entry Fields]
    WARNING:  Execution of the mksysb command will
              result in the loss of all material
              previously stored on the selected
              output medium. This command backs
              up only rootvg volume group.

* Backup DEVICE or FILE                                 []                      +/
  Create MAP files?                                      no                      +
  EXCLUDE files?                                         no                      +
  List files as they are backed up?                      no            +
  Generate new image.data file?                          yes                     +
  EXPAND /tmp if needed?                                 no                      +
  Disable software packing of backup?                    no            +
  Number of BLOCKS to write in a single output           []                      #
     (Leave blank to use a system default)

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

3.12 Merge of LVM and CLVM

AIX extends traditional UNIX disk management facilities through the *logical volume manager* (LVM). The logical volume manager provides sophisticated disk management services that allow system administrators to establish and manage their storage environment without significant effort or extensive experience.

Here LVM refers to the AIX standard logical volume manager. CLVM refers to the concurrent logical volume manager used in HACMP concurrent mode.

In AIX V4.2, the LVM and CLVM have been merged. The CLVM is therefore included in the base operating system.

3.12.1 Introduction

The basic objective of the CLVM merge is to have the base AIX LVM organization take on responsibility for the CLVM function and make it available as part of the base LVM. It avoids switching between standard LVM and CLVM and greatly simplifies HACMP concurrent management. It is also entirely transparent to the user and applications.

The current HACMP product uses a modified LVM called the concurrent LVM (CLVM). The LVM was modified because the AIX standard LVM assumes that only a single node is in control of any device, but with the concurrent access feature of HACMP, more than one node may be accessing the same device.

There are some special considerations for users performing regular system management tasks in a concurrent cluster environment running on previous AIX releases which are:

- The typical LVM control functions, such as adding or deleting logical volumes, resizing or adding mirrors, can be done only if the target volume group is varied on in non-concurrent mode.
- When using the `mksysb` command to perform system backups on cluster nodes, it is essential that the standard AIX LVM (non-concurrent) is the active LVM. Restoring from a `mksysb` tape that was created while the CLVM was the active LVM will fail. The reason is that the `mode3` command used by CLVM is not included by `mksysb` in the bootable image on tape.
- When installing fixes or maintenace updates (PTF′s) the installation must be done with the standard AIX LVM as the active LVM, otherwise it is possible for the wrong files to be modified.

This means that the user must switch back to the standard AIX LVM before performing any of the above administration tasks. This results in annoying operational problems for the user and can sometimes cause incompatibilities between the levels of LVM and CLVM. The CLVM and LVM merge eliminates or greatly simplifies these problems.

As a result of the merge of LVM and CLVM some commands now support additional options:

- varyonvg
- importvg
- mkvg
- chvg

These new options are discussed in greater detail in the following sections.

3.12.2 varyonvg Command

The purpose of varyonvg command is to activate a volume group.

Syntax:

```
varyonvg [ -f ] [ -n ] [ -p Number ] [ -s ] [ -c ] [ -b ] [ -u ] VolumeGroup
```

The -c, -b, and -u options are new options to the varyonvg command in AIX V4.2.

3.12.2.1 Support for varyonvg -b and varyonvg -u

The varyonvg command activates the volume group specified by the VolumeGroup parameter and all the associated logical volumes. When a volume group is activated, physical partitions are synchronized if they are not current.

The options -b and -u are provided for users who use n-tailed DASD systems. The base design of LVM assumes that only one initiator can access a volume group. The HACMP product is a product that works in conjunction with LVM in order to synchronize multi-node accesses to a shared volume group. However, it is not necessary to have the HACMP software installed to make volume groups accessible to multi-initiator nodes. The -b and -u options will make a volume group easily accessible to multi-initiator nodes without the use of HACMP software. It is imperative though that the user understands the conflicts that can occur when more than one CPU is accessing the same set of disks.

varyonvg -b breaks disk reservations on disks locked as a result of a normal varyonvg command. It unlocks all disks in a given volume group so that other systems can varyon the volume group simultaneously. Use this flag on a volume group that is already varied on.

The varyonvg command in AIX 4.1.x accepted the -b flag but it was not officially supported nor documented.

varyonvg -u varies on a volume group, but leaves the disk that makes up the volume group in an unlocked state. Use this flag as part of the initial varyon of a dormant volume group.

Important!

You must be aware that volume group status information may be compromised or inexplicably altered as a result of disk protect (locking) being bypassed with these two flags. If you use the -b and -u flags, data and status output cannot be guaranteed to be consistent.

3.12.2.2 Support for varyonvg -c

varyonvg -c varies on a volume group in concurrent mode. Use this option only if the volume group is concurrent capable and the HACMP product is loaded and available on the system. It was previously a part of the HACMP CLVM. It is now included in AIX V4.2 bos.

The SMIT panel for varyonvg now looks like:

```
                    Activate a Volume Group

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

                                                    [Entry Fields]
*VOLUME GROUP name                                      []
 RESYNCHRONIZE stale physical partitions?               yes
 Activate volume group in SYSTEM                        no
   MANAGEMENT mode?
 FORCE activation of the volume group?                  no
  Warning--this may cause loss of data
  integrity.
 Varyon VG in Concurrent Mode?                          no



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

The Varyon VG in Concurrent Mode field relates to the -c option.

3.12.3 importvg Command

The purpose of the importvg command is to import a new volume group definition from a set of physical volumes.

Syntax:

importvg [-c] [-f] [-V MajorNumber] [-x] [-y VolumeGroup] PhysicalVolume

The -c and -x options are new options added to the importvg command in AIX V4.2. Their functionality is as follows:

-c Imports the volume group and creates it as a Concurrent Capable volume group. Only use the -c flag with HACMP. It has no effect on volume groups and systems not using the HACMP product.

-x When used with the -c flag, sets the Concurrent Capable volume to be autovaried on in concurrent mode. When used without the -c flag, does nothing.

The SMIT panel for importvg now looks like:

```
                        Import a Volume Group

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

                                                    [Entry Fields]
* VOLUME GROUP name                                  []
  PHYSICAL VOLUME name                               []
  Volume Group MAJOR NUMBER                          []
  Make this VG Concurrent Capable?                    no
  Make default varyon of VG Concurrent?               no


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

The Make this VG Concurrent Capable? and Make default varyon of VG Concurrent? fields relate to the -c and -x options .

3.12.4 mkvg Command

The purpose of the mkvg command is to create a volume group.

Syntax:

```
mkvg [ -d MaximumPhysicalVolumes ] [ -f ] [ -i ] [ -c ] [ -x ] [ -m MaxPvSize
]  [ -n ] [ -s Size ] [ -V MajorNumber ] [ -y VolumeGroup ] PhysicalVolume ...
```

The -c and -x options are new options added to the mkvg command in AIX V4.2. Their functionality is as follows:

-c Creates a Concurrent Capable volume group. The -c option should only be used with the HACMP product.

-x Specifies to varyon automatically in Concurrent mode a Concurrent Capable volume group set to varyon automatically during a system restart. This flag has no meaning on systems which do not have the HACMP product installed.

The SMIT panel for mkvg now looks like:

```
                  Add a Volume Group

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

                                                  [Entry Fields]
  VOLUME GROUP name                                    []
* Physical partition SIZE in megabytes                  4
  PHYSICAL VOLUME names                                []
  Activate volume group AUTOMATICALLY                   yes
    at system restart?
  Volume Group MAJOR NUMBER                            []
  Create VG Concurrent Capable?                         no
  Auto-varyon in Concurrent Mode?                       no



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

The Create VG Concurrent Capable? and Auto-varyon in Concurrent Mode? fields relate to the -c and -x options .

3.12.5 chvg Command

The purpose of chvg command is to set the characteristics of a volume group.

Syntax:

chvg [-a AutoOn { n | y }] [-c] [-Q { n | y }] [-u] [-x { n | y }] VolumeGroup

The -c and -x options are new options added to the chvg command in AIX V4.2. Their functionality is as follows:

-c Changes the volume group into a Concurrent Capable volume group. However, the volume group must be varied on in non-concurrent mode for this command to take effect.

-x Changes the mode in which the Concurrent Capable volume group is varied on. The volume group must be varied on in non-concurrent mode for this command to take effect.

The SMIT interface for chvg now looks like:

```
                        Change a Volume Group
Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* VOLUME GROUP name                                      nonrootvg
* Activate volume group AUTOMATICALLY                    yes
    at system restart?
* A QUORUM of disks required to keep the volume          yes
    group on-line ?
Convert this VG to Concurrent Capable?                   no
* Auto-varyon VG in Concurrent Mode?                     no



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

The Convert this VG to Concurrent Capable? and Auto-varyon in Concurrent Mode? fields relate to the -c and -x options .

3.13 Enhancements to Shared Libraries

In AIX 4.2 there have been two major enhancements to the AIX linker and system loader:

- System V dynamic load interfaces - libdl.a routines
- Runtime linking - also known as hookable symbols

This implies a shared library and dynamic loader environment in which symbol names are resolved just as if the libraries were all linked statically.

3.13.1 Definitions Used

The following definitions are used throughout this section:

Binding This word is sometimes used as a synonym for *link-editing* or *linking*. Here binding means the association of a symbol name with a particular definition.

Module The smallest separately loadable and reloadable unit. On AIX, a module is either a program, a shared object that another module depends on, or an object file that is loaded with the load() systemcall. Shared objects can be archived, and in some cases, the loader can load them without extracting them from the archive. A module must be an XCOFF file with a *.loader* section.

Dependent Module A module that defines symbols required by another module. At link time, a list of dependent modules is saved in the output module.

Shared Object A module that can be loaded and shared by multiple processes at run-time. When a shared object is specified during linking, information about the object is recorded in the output file, but no code from the shared object is actually included in the output file.

Relocation Relocation means two things: assigning new addresses to symbols or sections and altering instructions and address constants to reflect the new addresses. The appropriate meaning can usually be deduced from context.

Symbol resolution The process of binding symbol names to definitions. Symbol resolution occurs in the compiler, the linker and the loader.

Exec time The *instant* when the loader creates a new process image. The module being executed is loaded at exec() time. Whether other modules are loaded at the same time depends on the implementation.

Run time The period when a program is actually running. Additional modules can be loaded and symbols resolved at run time, either explicitly or implicitly, depending on the implementation. On AIX, programs call load() to load additional modules and loadbind() to resolve deferred imports. On other platforms, dlopen() is used to load modules and dlsym() is used to resolve symbols by name.

Load time The *instant* when a module is being loaded. Modules are loaded as the result of an exec() call or at run time, either because of a load() or dlopen() call, or a reference to an unresolved symbol.

Dynamic Loading A technique by which references from a module to other modules are not resolved until a symbol is first used. On most platforms that implement this technique, only function references can be linked dynamically. Variables must all be linked at load time. This technique allows faster process startup, but can cause arbitrary delays when the first reference to a symbol is made. In addition, if a symbol definition cannot be found when it is first referenced, the process exits.

Dynamic Linking Dynamic linking refers to linking with a shared object. A dynamically-linked module is not self-contained, because it requires definitions from dependent modules. The *loader* (or *run-time linker* on some platforms) loads these dependent modules and resolves inter-module references.

Static Linking Static linking refers to linking with static libraries instead of shared objects. A statically linked program is self-contained. That is, it has no dependent modules. (On AIX, statically-linked programs appear to have a dependency on /unix, but this pathname is used to designate the running kernel, including kernel extensions).

Object File A general term for a file containing executable code and relocation information. Depending on the contents of an object file, the file can be used as a module, used as input to the linker, and so on.

Loader The system loader is a part of the kernel. Some platforms have a user-space loader, which can also be called a run-time linker.

3.13.2 Support for System V.4 Dynamic Loading

The support for the system V.4 dynamic loading makes it easy for porting applications that were written for some other UNIX platform. Most independent software vendors require this function.

The libdl.a routines (dynamic load interfaces) allow binding of a name that is not known until run time. Such bindings are made symbolically, and multiple bindings for a given symbol name can be created and used (using function pointers in the C language).

The new dynamic load interfaces include:

- dlopen()
- dlsym()
- dlclose()
- dlerror()

3.13.2.1 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, which can be modified during execution with 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.

RETURN VALUES

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().

3.13.2.2 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.

RETURN VALUE

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

3.13.2.3 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.

SYNOPSIS

```
#include <dlfcn.h>

int dlclose(void *handle);
```

RETURN VALUE

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

3.13.2.4 dlerror()

This function is used to return error information about the most recent call to 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.

3.13.3 Run-time Linking - Hookable Symbols

Run-time linking (also known as *hookable symbols*) provides the capability to defer most symbol bindings until load time. Here, *binding* means the association of a symbol name with a particular definition.

The AIX shared library hookable symbols facility is designed to implement the ability to replace symbol definitions that reside within shared objects at run time. Symbol definition precedence is a function of the command line ordering used when linking an application and applies to both functions and data structures. The symbols within the application are *hooked*, thereby allowing a run time module to preempt symbol reference definitions upon application start-up, but before the main subroutine begins execution.

By default, references to symbols in shared objects are bound at link time. That is, the output module associates an imported symbol with a definition in a specific shared object. At load time, the definition in the specified shared object is used even if other shared objects use the same symbol.

You can cause your program to use the run-time linker, allowing some symbols to be rebound at load time. To create a program that uses the run-time linker, link the program with the `-brtl` option. The way that shared modules are linked affects the rebinding of symbols.

3.13.3.1 Run-time Linking at Exec Time

The main executable is loaded and resolved by the system loader in the usual way. If the executable cannot be loaded for any reason, the exec() call fails and the run-time linker is not invoked at all. If the executable loads successfully, control passes to the run-time linker. When the run-time linker completes, main() is called.

The run-time linker processes modules in breadth-first search order, starting with the main executable and continuing with the direct dependents of the main executable, following the order of dependent modules listed in each module′s loader section. This order is also used when searching for the defining instance of a symbol. The *defining instance* of a symbol is usually the first instance of a symbol.

The loader section of each module lists imported symbols, which are usually defined in another specified module, and exported symbols, which are usually defined in the module itself.

Symbols can also be marked as *deferred imports*. Reference to deferred import symbols are never rebound by the run-time linker. Resolution of these symbols must be performed by the system loader, either by calling loadbind() or by loading a new module explicitly.

References to imported symbols (other than deferred imports) can always be rebound. The system loader will have already resolved all imports, except deferred imports and imports from the symbolic module DOT-DOT (using two

periods as the defining module for an imported symbol means the symbol will be resolved by the run-time linker, not the system loader).

References to each imported symbol are rebound to the symbol's defining instance. If no defining instance exists, an error message will be printed to standard error. In addition, if the type checking hash string of an imported symbol does not match the hash string of the defining symbol, an error message is printed.

References to exported symbols are also rebound to their defining instances, as long as the references appear in the relocation table of the loader section.

Once all modules have been processed, the run-time linker calls exit(). If any run time linking errors occurred, it passes a return code of 144(0x90). Otherwise, execution continues by calling other initialization routines or main().

3.13.3.2 Run time Linking After an Explicit Load

When an application is enabled for run time linking, the run-time linker receives control when new modules are loaded with load() or dlopen(), as long as the module was loaded successfully. Symbol resolution proceeds as it does at exec time with a few differences:

- The first difference is that only newly loaded modules need to be rebound. Previously loaded modules may be searched to find defining instances, but they are not modified.
- The second difference is that not all modules are searched to find a defining instance. Modules are searched as follows. First, the main executable and its original dependents are searched in breadth-first order. Then, most explicitly loaded modules are searched in the order they were loaded, as long as they haven't been unloaded. For each explicitly loaded module being searched, the module and its original dependents are searched in breadth-first search order. To prevent a module loaded with dlopen() from being searched, the RTLD_GLOBAL flag must not be used. Modules loaded with load() will always be searched.

  If run time linking is successful, initialization routines are called for each newly loaded module. If any errors occur either in the system loader or the run-time linker, relevant information is saved, to be returned to subsequent calls to dlerror() or loadquert().

3.13.4 Linker Changes

No new application programming interfaces (APIs) are added to support run time linking. Run time linking is enabled by using new linker options. The modified linker works in the same way as before unless new options are specified.

The purpose of `ld` command is to link object files.

Syntax:

```
ld [-DNumber ] [ -eLabel ] [ -G ] [ -HNumber ] [ -K ] [ -m ] [ -M ] [ -oName ]
[ -r ] [ -s ] [ -SNumber ] [ -TNumber ] [ -u Name ] ... [ -v ] [ -z ] [
-ZString ] ... [ -bOption ] ...  [ -LDirectory ] ... { -fFileID ... -lName ...
InputFile ...}
```

Several new options and features are added to the `ld` command.

New -b Options:

- -brtl and -bnortl (the default)
- -bdynamic (the default) and -bstatic
- -bsymbolic, -bnosymbolic and -bsymbolic- (the default)
- -binitfini
- -bautoexport (the default) and -bnoautoexport
- -bipath (the default) and -bnoipath
- -bexpall and -bnoexpall (the default)
- -brtllib and -bnortllib (the default)

Except for the -bautoexport option, the default behavior of the linker is unchanged.

The general technique used to create a shared object will be to link with ld, specifying the -G flag. Programs using run time linking should be linked with the cc command and the -brtl option.

The new values for the Options variable of the -b flag are described below. You can list more than one option after the -b flag, separating them with a single blank:

1. -brtl and -bnortl (the default)

   rtl Enables run time linking for the output file. This option implies the rtllib and symbolic options.

   When dynamic mode is in effect (see the dynamic and static options), the rtl option allows input files specified with the -l flag to end in .so as well as in .a.

   All input files that are shared objects are listed as dependents of your program in the output files loader section. The shared objects are listed in the same order as they were specified on the command line. A shared object contained in an archive is only listed if the archive specifies automatic loading for the shared object member. You can specify automatic loading for an archive member foo.o by creating a file with the following lines:

   ```
   # autoload
   #! (foo.o)
   ```

   and adding the file as a member to the archive.

   nortl Disables run time linking for the output file. This option implies the nortllib and nosymbolic- options. Furthermore, additional actions described under the rtl option are not taken.

2. -bdynamic (the default) and -bstatic

   dynamic or shared Causes the linker to process subsequent shared objects in dynamic mode. In dynamic mode, shared objects are not statically included in the output file. Instead, the shared objects are listed in the loader

section of the output file. When you specify the rtl option and dynamic mode is in effect, files ending in .so as well as .a satisfy searches for libraries specified with -l flag.

static Causes the linker to process subsequent shared objects in static mode. In static mode, shared objects are statically linked in the output file. Furthermore, files ending in .so are not found when searching for libraries specified with the -l flag.

3. -bsymbolic, -bnosymbolic and -bsymbolic- (the default)

   symbolic Assigns the symbolic attribute to most symbols exported without an explicit attribute.

   nosymbolic Assigns the nosymbolic attribute to most symbols exported without an explicit attribute.

   nosymbolic- Assigns the nosymbolic- attribute to most symbols exported without an explicit attribute.

4. -binitfini

   initfini:[**Initial**][**:Termination**][**:Priority**]

   Specifies a module initialization and termination function for a module, where Initial is an initialization routine, Termination is a termination routine, and Priority is a signed integer, with values from -2,147,483,648 to 2,147,483,647. You must specify at least one of Initial and Termination, and if you omit both Termination and Priority, you must omit the colon after Initial as well. If you do not specify Priority, 0 is the default. This option may be repeated.

5. -bautoexport (the default) and -bnoautoexport

   autoexport Automatically exports some symbols from the output module without having to list them in an export file. (This option does not export all symbols from the output module. Use the -bexpall option to export all symbols). Use this option when linking a main program. The linker assumes that you are linking a main program when you do not specify a module type (with the M or modtype option) beginning with S and you do not use the noentry option.

   When you use the autoexp option, if any shared object listed on the command-line imports a symbol from the special file . (dot), and the module being linked contains a local definition of the symbol, the symbol is exported automatically.

   Other symbols are also exported automatically when you link with the -rtl option. If a symbol defined in the module being linked has one or more additional definitions exported from a shared object listed on the command-line, and if any of the definitions is a BSS symbol, the symbol is exported automatically. If the definition in the module being linked is a BSS symbol, the symbol is exported with the nosymbolic attribute.

Otherwise, the symbol is exported with the symbolic attribute. If the symbol is listed in an export file with another export attribute, the explicit attribute is used.

If the autoexp option would automatically export a symbol, but the symbol is listed in an export file with the list attribute, the symbol is not exported.

noautoexp Prevents automatic exportation of any symbols.

6. -bipath (the default) and -bnoipath

   ipath For shared objects listed on the command-line, rather than specified with the -l flag, use the path component when listing the shared object in the loader section of the output file.

   noipath For shared objects listed on the command-line, rather than specified with the -l flag, use a null path component when listing the shared object in the loader section of the output file. A null path component is always used for shared objects specified with the -l flag. This option does not affect the specification of a path component by using a line beginning with #! in an import file.

7. -bexpall and -bnoexpall (the default)

   expall Exports all global symbols from the output module without having to list them in an export file. Imported symbols, unreferenced symbols defined in archive members, and symbols beginning with an underscore (_) are not automatically exported. You may export additional symbols by listing them in an export file. This option does not affect symbols exported by the autoexp option.

   This option was added to make it easier to build shared modules that need to simply export every symbol.

   When the executable is generated, all remaining external symbols will be exported and written to the loader section.

   The option can replace the -bE: (-bexport:) option, depending upon the desired API for a module.

   noexpall Does not export symbols unless you list them in an export file or you export them with the autoexp option.

8. -brtllib and -bnortllib (the default)

   rtllib Includes a reference to the run-time linker. The run-time linker is defined in librtl.a, and an implicit -lrtl flag is added automatically to the command line. This option (implied by the rtl option) must be used when linking a main program or no run time linking will occur. Shared objects do not have to be linked with this option.

nortllib Does not include a reference to the run-time linker. If a main program is linked with this option, no run time linking will take place in the program, regardless of the way any shared modules were linked that are used by the program.

New flag:

-G Produces a shared object enabled for use with the run-time linker. The -G flag is equivalent to specifying the erok, rtl, nortllib, nosymbolic, noautoexp, and M:SRE options with the -b flag. Subsequent options can override these options.

3.13.5 rtl_enable Command

rtl_enable relinks shared objects to enable the run-time linker to use them.

Syntax:

```
rtl_enable [ -R | -o Name ] [ -l ] [ -s ] File
           [ ldFlag ] [ -F ObjsLibs ]
```

The rtl_enable command relinks a module, or an archive containing modules, with the -G flag, to enable run-time linking. A module is an XCOFF file containing a loader section. A shared object is a module with the F_SHROBJ flag set in the XCOFF header.

In its simplest form, the rtl_enable command creates a new file with the name File.new. If File is a module, File.new. will be the same kind of module. If File is an archive, File.new. will be an archive whose members have the same names as the members of File. The rtl_enable command relinks the modules in the new archive to enable run-time linking. The rtl_enable command archives other members unchanged into the output file.

The rtl_enable command uses the loader section in File (or its members) to create import and export files, to determine the libpath information, and to determine the entry point.

Flags

-F ObjsLibs Adds ObjsLibs to the beginning of the generated ld command. The ObjsLibs parameter is either an object file or a library (specified with the ld command's -l (lowercase L) flag). If you are enabling an archive, it adds the ObjsLibs to the ld command for all shared objects in the archive.

-l Leaves the import and export files in the current directory instead of deleting them. Import files have the suffix .imp and export files, the suffix .exp. The rtl_enable command adds the suffixes to the input file name if File is a module. It adds the suffixes to the names of members that are modules if File is an archive.

-o Name Specifies an alternate output file name instead of File.new. Do not use this flag with the -R flag.

-R Replaces the input file instead of creating a new file. It will not overwrite the input file if any errors occur. Do not use this flag with the -o flag.

-s Generates a script of commands in the current directory that you can use to create a new output file or archive, but does not relink anything. It names the script `Base.sh`, where `Base` is the basename of the input file with any suffix stripped off. It writes generated import and export files in the current directory as well. You can modify the script and the import and export files to customize the output objects.

Parameters

File Specifies the input file.

ldFlag Copies the specified `ld` command flags to the end of the generated `ld` command, overriding default options.

Note: Do not use the `-o` flag in the `ldFlag` parameter to name the output file. To specify an alternate output filename, use the `rtl_enable` command's `-o` Name flag.

Examples

To create a new version of `libc.a` with run-time linking enabled, enter:

1. Create a new directory for the runtime version by entering:

   ```
   mkdir /tmp/rtllibs
   ```

2. Make `/tmp/rtllibs` your current directory by entering:

   ```
   cd /tmp/rtllibs
   ```

3. To create the run-time version of `libc.a` with the same name, enter:

   ```
   rtl_enable -o libc.a /lib/libc.a
   ```

4. To use this version of `libc.a` when linking programs, use `-L /tmp/rtllibs` with the `ld` command.

3.13.6 Binary Compatibility and Performance

Binary compatibility has been maintained in AIX 4.2, the default action of the compiler, linker, and loader provides behavior identical to that in AIX 4.1. Also, programs that are recompiled or relinked will not be affected by the enhancements, unless new options are used. Existing executables and libraries that work on AIX 4.1 will still work in exactly the same way on AIX 4.2. Run time linking is only enabled for new binaries which have been generated with the new linker flags.

On the other hand, for run time linking to be fully effective, existing shared objects may have to be recompiled or relinked when used with applications that make use of run time linking. Otherwise, the ability of the run-time linker to rebind symbols will be limited.

For applications that do not use the run time linking features performance is not affected. Both new and existing binaries should load and execute with the same performance as in AIX 4.1. The loader changes required to support run time linking do not affect the performance of loading modules that do not use this feature.

3.13.7 Installing the New Linker

The run-time linker is a part of the BOS runtime environment and replaces the existing linker. Nothing further needs to be installed.

The libdl.a dynamic routines are always installed as part of the bos.rte fileset.

3.13.8 Resource Utilization

When the run-time linker is used, additional memory will be used, both in the loader's kernel heap and in the process's heap. When the run-time linker is not used, the loader will not use additional memory (additional fields have been added to the existing loader structures, so there is a small increase in the amount of memory used by the loader even if run time linking is not enabled).

The dynamic loading routines use malloc() to allocate memory to support these routines.

The modified linker does require slightly more memory since internal data structures have been expanded to accommodate the new options. This extra memory is allocated even if the new options are not used.

3.13.9 Application Binary Interface

Calls to load(), unload() and loadquery() from existing programs continue to work without change.

New calls dlopen(), dlsym(), dlclose() and dlerror() are added to a new library.

3.13.10 Packaging

Some changes to the packaging have been made to accomodate the new linker.

- A new library /usr/ccs/lib/libdl.a has been added for the new dynamic load interfaces: dlopen(), dlsym(), dlclose() and dlerror(). This library is shipped as part of bos.rte.bind_cmds fileset. It contains a single shared object shr.o.

  There is a symbolic link from /usr/lib/libdl.a to /usr/ccs/lib/libdl.a.

- A new header file dlfcn.h is installed in the /usr/include directory. It is supplied as part of the bos.adt.include fileset.

- The run-time linker is added to libc.a(shr.o). No new crt0.o files are needed, since the existing files were modified.

3.14 64-Bit Development Hooks

The 64-bit development hooks provide the enabling changes needed in AIX 4.2 in order to support 64-bit applications in later releases. Changes have been made in the following areas:

- Hooks (enabling code) and branch table entries have been changed to allow 64-bit context saving to be added via extensions.

- Package definitions have been changed to allow 64-bit application support extensions to be built on top of AIX V4.2.

- Hooks and table entries have been changed to allow 64-bit binary object format to be added as system extensions.

Hook is just a term used to describe the *enablement* for 64-bit development work. For example, there are *hooks* to recognize new 64-bit specific system calls, support for saving and restoring the context of a 64-bit process and for management of the 32/64 mode bit of the processor.

Inclusion of the 64-bit hooks allows support for 64-bit applications to be installed as extensions and incremental changes on top of an AIX V4.2 system at a later date. The extra kernel extensions and scaffolding (64-bit loader, 64-bit debuggers, 64-bit tools) are not part of the AIX V4.2 product.

3.15 Linker Support for Large Branch Offsets

AIX V4.2 can now handle relative branch displacements greater than 26 bits (+/- 32 MB) for use in large user applications. More specifically, the linker is able to link and relocate modules containing branch instructions that are more than 32 MB away from their targets. Applications with large executables (.text sections) require this functionality.

Some executables can already be linked successfully even though their resulting text sections are larger than 32 MB. This is because relocation overflow only occurs when branch instructions are more than 32 MB from their targets.

The Power and PowerPC processors have a branch instruction that allows branches within 32 MB of the current location. That is, the instruction contains a 26-bit, signed offset that is added to the current location counter to compute the destination address. When the destination is more than 32 MB away from the branch instruction, the destination is unreachable. Before AIX V4.2, this situation would cause the linker to issue a *relocation overflow* message, and the link would fail. On AIX V4.2 the branch is modified, so it points to another branch, and this second branch instruction branches to the destination. That is, two consecutive branches are required to reach a far away destination.

Branches to far away targets is fixed up at link time so that they can branch to their targets with two or more branch instructions. There is no limit for the branch displacement. On the other hand, the AIX V4.2 linker will not write an object file larger than 256 MB, so this restriction effectively limits the maximum branch displacement.

The approach the binder takes in handling Large Branch Offsets is as follows:

1. Insert a *hop branch table* at every 32 MB boundary within the executable to handle out of range branches across those boundaries.
2. Move any binder generated *fixup* code adjacent to the csect that requires the fixup (rather than to the end of the *.text* section).

Branches to targets that are too far away to be reached with a single branch instruction will be replaced with a branch to an intermediate branch instruction. These intermediate branch instructions are called *trampoline* branches. A restriction is that the target of the branch must be a csect (an XTY_SD symbol) or a label (an XTY_LD symbol). For example, a branch to a symbol *foo* will be fixed up. A branch to *foo+4* will not be fixed up if it results in relocation overflow. AIX compilers never generate a branch to a symbol plus an offset, so this restriction only affects assembly language programs and third party compilers.

No new options are needed to enable the generation of trampoline branches.

Existing modules that link successfully with the current linker will continue to link successfully with the new linker.

3.16 exec/fork Enhancement for Graphics Processes

AIX V4.2 adds a new resource handler during exec() processing that will enable the graphics subsystem (RCM) to clean up. RCM stands for the Rendering Context Manager that is central to graphics support.

This new feature is used to resolve an existing problem involving segment registers and fork()/exec().

If a parent process is a graphics process (GP), this means that at least one of the segment registers has been used for an adapter address. With shared memory, it is also possible that one or two other registers are in use for graphics specific functions.

If the parent fork()′s and exec()′s a child, the child′s address space is copied from the parents, which includes all these memory objects. The child actually doesn′t need (shouldn′t have) access to the graphics objects until it becomes a graphics process.

In the problem situation, the child is not a graphics process. Instead, it wants to use the large memory model which asks for lots of segment registers. The problem is that several registers appear to be in use for graphics when they don′t need to be, so an insufficient number are available for the large memory model process.

The reason is because graphics processes are not cleaned up during fork() and exec() so their graphics segment registers remain allocated. This causes a subsequent exec() of a large memory program such as the Fortran compiler to fail for lack of sufficient memory. There is now a new resource handler callout during exec() processing that enables the graphics subsystem (RCM) to clean up.

3.17 New SMIT Menu for System Backup

System backup utilities are now accessible from the `smit install` fastpath menu. In AIX V4.2 an entry has been added to the Software Installation and Maintenance SMIT panel so that it is possible to backup the system from the same SMIT panel that is used for installing and updating software. This makes it more obvious and convenient to backup the system before and after an installation or changes have been made.

The new SMIT panel looks like the following:

```
                  Software Installation and Maintenance

Move cursor to desired item and press Enter.

Install and Update Software
List Software and Related Information
Software Maintenance and Utilities
Network Installation Management
System Backup Manager



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

The last option System Backup Manager is the new entry.

3.18 Restoring mksysb Backups to Different Hardware Platforms

It is now possible to create system backups with mksysb that can be installed on target systems with different hardware configurations to the source system.

For example, you can create a system backup image that can be installed on both uniprocessor and SMP machines.

Additionally, the device and kernel support required on the target system does not necessarily have to be installed on the source system. For example, if the target system requires PCI device support that was not installed on the source system, the required filesets for PCI support can be retrieved from the product CDROM for the target system without ever being installed on the source system.

To restore the mksysb tape, you can:

- Boot from product media CD

  This uses the product media CD and the mksysb backup tape

- Boot from product media tape

  This uses the product tape, mksysb backup tape and a diskette containing two files: bosinst.data and signature

Note

When booting from tape, there must be a diskette in the diskette drive containing a ./signature file consisting of the four characters *data* and a ./bosinst.data file containing the SWITCH_TO_PRODUCT_TAPE=yes statement in the control_flow stanza.

The following is an example of how to restore a mksysb tape from a uni-processor machine to an SMP machine.

First, insert the product media CD or tape and boot the system into maintenance mode. When the system is booted the following screen will be displayed.

```
           Welcome to Base Operating System
           Installation and Maintenance
Type the number of your choice and press Enter
Choice is indicated by >>>.

>>> 1. Start Install Now with Default Settings
    2. Change/Show Installation Settings and Install
    3. Start Maintenance Mode for System Recovery




    88. Help?
    99. Previous Menu
>>> Choice[1]:
```

Selecting option **3** will display the screen below.

```
                 Maintenance
Type the number of your choice and press Enter.

>>> 1. Access a Root Volume Group
    2. Copy a system Dump to Removable Media
    3. Access Advanced Maintenance Functions
    4. Install from a System Backup




    88. Help?
    99. Previous Menu

>>> Choice[1]:
```

Enter **4** to install from a mksysb tape and then choose the tape drive that contains the backup from the next screen.

```
                Choose Tape Drive
Type the number of the tape drive containing the system backup to be
installed and press Enter.

       Tape Drive                   Path Name
>>> 1. tape/scsi/8mm               /dev/rmt0




    88. Help?
    99. Previous Menu

>>> Choice[1]
```

If you booted from a product tape, you must remove the product tape and insert the mksysb tape.

If you booted from a product CD, leave the CD in the CD drive and insert the mksysb tape into the tape drive. The mksysb procedure in BosInstall (bi_main.sh) will detect that the install media is available and then :

- Determine whether the UP or MP version of the kernel is needed (based on the output of the bootinfo -z command) and call the devinstall command for the required fileset (bos.up or bos.mp).

  Note: bootinfo -z specifies whether the machine hardware is MP-capable (capable of running the multi-processor kernel and supporting more than one processor). The return value indicates:

  0 The machine is not MP-capable.
  1 The machine is MP-capable.

- Call devinstall -f /tmp/devices.pkgs to install all the required device support available on the install media. The /tmp/devices.pkgs file is created by a previous call to the cfgmgr.
- Call the cfgmgr, configuration manager to install any device support required by the target system but not included in the mksysb image. The cfgmgr -i flag references the install media.

This processing must occur before BosInstall calls bosboot to create the new boot image for the target machine.

In the case of tape, if SWITCH_TO_PRODUCT_TAPE is set to yes, the user is prompted when to remove the mksysb tape and insert the product tape.

These steps will ensure that the correct kernel and device support are installed on the target machine even though these filesets are not available in the mksysb image.

Since this processing takes place whenever install media is detected during the mksysb procedure of BosInstall, the NIM mksysb case is also supported.

3.19 Currently Unsupported LPP′s

There are hundreds of IBM licensed programs available for AIX Version 4. all the licensed programs sold today for AIX V4.1 will also run on AIX V4.2 except those that are in the process of redesign, development of an enhanced version/release, or a replacement product. Those exceptions are:

- IBM AIX Continuous Speech Runtime, Version 1
- IBM AIX Continuous Speech Developers Toolkit, Version 1
- IBM High Availability Cluster Multi-Processing(HACMP) for AIX, Version 4.1.1
- IBM CommonPoint Application System for AIX, Version 1.1
- IBM CommonPoint Application Development Toolkit for AIX, Version 1.1
- IBM Client Input Output/Sockets, Version 2.1
- IBM InterMix for AIX and UNIX Environments, Release 1.0
- Internet Connection Secured Network Gateway for AIX, Version 2.1 Program Package
- IBM LoadLeveler, Version 1.2
- IBM Multimedia Server for AIX, Version 1.1
- IBM Network Tape Access and Control System for AIX(NetTAPE), Version 1.1
- IBM NetTAPE Tape Library Connection, Version 1.1

- IBM NetView for AIX, Version 3
- IBM Nways Family for AIX
- IBM Parallel Environment for AIX, Version 2 release 1
- IBM Parallel OSL, Version 1.1
- Time and Place/6000, Version 1.1
- IBM 3270 Emulator for the X Window System, Version 1.2.2(X3270)

The RISC System/6000 SP systems are not currently supported by this initial release of AIX Version 4.2.

The following licensed programs for AIX Version 4.2 should be used with AIX Version 4.2 instead of a product or products (listed in parentheses) which were available for AIX Version 4.1 before this announcement:

- OpenGL and GL 3.2 Version 4.2 for AIX (Instead of OpenGL and GL 3.2 Version 4.1 for AIX)
- PEX and PHIGS Version 4.2 (Instead of PEX and PHIGS Version 4.1 for AIX)
- IBM Soft5080 for AIX Version 2.3 (Instead of Soft5080 for AIX Version 1.1.1)
- Performance Toolbox Version 2.2 for AIX (Instead of Performance Toolbox Version 3)
- Communications Server for AIX (Instead of IBM SNA Server for AIX Version 3)
- Transaction Server for AIX (Instead of CICS for AIX Version 2.1, CICS System Manager Version 1.1 and the Encina Family Version 2.1 products)

3.20 Migration

AIX Version 4.2 supports system migration installation for systems previously installed with any level of AIX Version 3.2 or AIX Version 4.1.

Many of the restrictions placed on the migration from AIX Version 3.2.5 to AIX Version 4.1 are not present in the move from AIX Version 4.1 to AIX Version 4.2. Device drivers and kernel extensions that presently operate in an AIX Version 4.1 environment should migrate without problem to AIX Version 4.2. The compatibility installation options available in AIX Version 4.1 continue to be available in AIX Version 4.2.

3.21 Power-Off Facilities

In previous versions of AIX the `shutdown` and `halt` commands automatically issue a write() to the power control register that causes the racks and newer desksides to turn the power off after the `halt` is completed.

If we `shutdown` these models (examples are 570/580s), the only way to re-ipl the machine is to switch off main power for about 10 to 15 seconds and power on again.

In AIX V4.2, a **-p** option provides the ability for `halt` and `shutdown` to halt the system and leave it powered up.

The shutdown command will accept the **-p** flag, and pass it internally to the halt command and perform a halt without power down.

The **-c** flag for shutdown which was used to disable checking the file systems upon restarting is now obsolete and it performs the same function as the **-r** option.

3.21.1 shutdown Command

Syntax:

```
shutdown [ -c][ -d][ -F][ - i][ -k][ -m][ -p][ -r][ -hv | -t mmddHHMM[yy]]
         [ + Time:]:[ Message:]
```

Flags

-c Obsolete; same as **-r**

-d Brings the system down from a distributed mode to a multiuser mode.

-F Does a fast shutdown, bypassing the messages to other users and bringing the system down as quickly as possible.

-i Specifies interactive mode. Displays interactive messages to guide the user through the shutdown.

-k Avoids shutting down the system.

-m Brings the system down to maintenance (single user) mode.

-p **Do not power the system down.**

Note: If -p is specified in combination with flags not requiring a permanent halt, it will have no effect. In particular, power will still be turned off if other operands request a delayed power on reboot.

-r Restarts the system after being shutdown with the reboot command.

-h Halts the operating system completely; same as the -v flag.

-v Halts the operating system completely.

-t mmddHHMM [**yy**] Restarts the system on the date specified by mmddHHMM [yy] where:

- mm Specifies the month.
- dd Specifies the day.
- HH Specifies the hour.
- MM Specifies the minute.
- yy Specifies the year.

The shutdown -t flag cannot be used with the -v or -h option.

Note: This option is only supported on systems that have a power supply which automatically turns power off at shutdown and an alarm to allow reboot at a later time. Systems without this capability may hang or may reboot immediately after shutdown.

Parameters

+Time Specifies the time at which the shutdown command stops the system. An immediate shutdown is indicated by the word now displayed on the screen. A future time can be specified in one of two formats: +number or hour : minute. The first form brings the system down in

the specified number of minutes and the second brings the system down at the time of day indicated (as a 24-hour clock). If the Message parameter is specified, the Time parameter must also be specified.

Message Specifies the message to be sent.

3.21.2 halt Command

Syntax:

`halt [-p][-y][-q][-l][-n]`

Flags

-p **Do not power the system down**.

-l Does not log the `halt` in the accounting file. The -l flag does notsuppress accounting file update. The -n and -q flags imply the -l flag.

-n Prevents the sync before stopping.

-q Causes a quick halt.

-y Halts the system from a dial-up operation.

3.22 Removal of Support for bootinfo Command

This command determines and displays various boot information, including boot device type and boot device name. In AIX V4.2, **-a, -L, -M** and **-p** flags were added.

Important

This command is not a user-level command and is not supported in AIX Version 4.2 or later.

Syntax:

```
bootinfo { -a | -b device | -B | -c | -e | -k | -L | -M | -m | -o disk |
           -p | -q device | -r | -s diskname [ -P 0 ] | -t | -T | -z }
```

Flags

-a Displays the model architecture.

- 1 is RS6K
- 2 is RSPC

-b Displays name of the device that was last booted from.

-B Disk Displays an indication as to whether or not the firmware supports booting from the named device:

- 0 Not bootable
- 1 Bootable
- 2 Indeterminate

-c Displays bootp daemon reply packet information stored with IPL control block.

-e Displays an indication as to whether or not the model supports tape boot:

- 0 Tape boot not supported
- 1 Tape boot supported

-k Specifies the key position. The return value indicates:

- 1 Key is in Secure position
- 2 Key is in Service position
- 3 Key is in Normal position

-L Displays an indication as to whether or not the model has LEDs:

- 0 Model does not have LEDs
- 1 Model has LEDs

-m Displays the model code from the sys0 device′s modelcode attribute.

-M Displays the model code and has the following exit codes:

- 255 Model type is indeterminate
- 100 Entry server type system, no diagnostics
- 101 Entry server type system, diagnostics supported
- 102 Not an entry server type, no diagnostics
- 103 Not an entry server type, diagnostics supported

-o Disk Displays either the disk device name or the location depending upon the value of disk.

-p Displays the package descriptor or platform name.

-P Ppsize Size of disk physical partitions to be used for calculating defaults.

-q Adapter Displays an indication as to wether or not the firmware supports booting from the named Adapter.

- 0 Not bootable
- 1 Bootable
- 2 Indeterminate

-r Displays amount of real memory in kilobytes.

-s Disk Displays disk size in megabytes.

-t Specifies the type of boot. The return values include:

- 1 Disk boot
- 3 CD-ROM boot
- 4 Tape boot
- 5 Network boot

-T Displays one of the following strings:

- rspc - for a model whose model architecture is RSPC
- rs6k - for a uni-processor model whose model architecture is RS6K
- rs6ksmp - for a multi-processor model whose model architecture is RS6K This option is provided only for compatibility with earlier versions of AIX V4.1.

-z Specifies whether the machine hardware is MP-capable (capable of running the multi-processor kernel and supporting more than one processor). The return value indicates:

- 0 The machine is not MP-capable.
- 1 The machine is MP-capable.

Description

The `bootinfo` command is used during the boot and bos install to gather and display information. During boot it is used to determine the boot device type and which device the machine has booted from. During a network boot, the `bootinfo` command displays the contents of the client's bootpd daemon REPLY packet. This information is used by the boot programs to contact the server to obtain the client's file systems.

The `bootinfo` command uses the device configuration databases in information searches. For some information, the `bootinfo` command uses NVRAM (nonvolatile random access memory).

When booting a system across a network, use the -c flag to display the following information:

- Client IP address
- Server IP address
- Gateway IP address
- Network type of boot, where:
  - 68 (ASCII "D") Ethernet
  - 79 (ASCII "O") Token Ring
  - 80 (ASCII "P") FDDI

- Slot number of the network adapter
- 802.3 indicator
- Boot file
- Vendor tag information
- Hardware attribute value

For token-ring:

- 4 Specifies 4 megabit-per-second token ring
- 5 Specifies 16 megabit-per-second token Ring

For MCA Ethernet:

- 7 Specifies thin cable
- 8 Specifies thick cable

All of the items except the hardware attribute value are part of the bootp daemon REPLY packet information. The `bootinfo` command determines the hardware attribute value by looking in NVRAM.

3.23 Reading bootlist Information

The bootlist command alters the list of boot devices (or the ordering of devices on the list) available to the system. In AIX V4.2, a **-o** flag was included to enable the device list specified by the **-m** flag to be displayed.

Syntax:

```
bootlist -m Mode [ -i | [ -r] [ -o] [ -f File]
                 [ Device [ Attr=Value ... ] ... ]
```

Flags

Device Provides the names of the specific or generic devices to include in the boot list.

-f File Indicates that the device information is to be read from the specified file name.

-i Indicates that the device list specified by the -m flag should be invalidated.

-m Mode Specifies which boot list to alter. Possible values for the mode variable are normal, service, both, or prevboot.

-r Indicates that a hexadecimal dump of the specified device list in nonvolatile RAM should be output after any specified alteration is performed. (This is normally used for problem determination.)

-o Indicates that the device list specified by the -m flag should be displayed.

3.23.1 bootlist Command

The bootlist command allows the user to alter the list of boot devices scanned by read-only-storage (ROS) when the system is booted. This command can alter the contents of the two battery backed-up RAM (NVRAM) boot device lists and the choice of boot device used on the next and subsequent system boots. This command supports the updating of the following:

- Service boot list. The service list designates possible boot devices when the front panel keylock is in the service position.
- Normal boot list. The normal list is used when the keylock is in the normal position.
- Previous boot device entry. Retained in battery-backed-up RAM on the system unit.

Each list contains a maximum of 84 bytes. When searching for a boot device, the ROS system selects the first device in the list and determines if it is bootable. If no boot file system is detected on the first device, ROS moves on to the next device in the list. As a result, the ordering of devices in the device list is extremely important.

If no device list has been supplied, or if it was empty, the ROS system attempts to boot from the boot device used on a previous boot. (This assumes that the previous boot device was not a diskette drive.)

If this boot device is unavailable or not bootable, the ROS system starts searching the I/O bus for the first device from which it can boot.

The bootlist command supports the specification of generic device types as well as specific devices for boot candidates. Possible device names are listed either on the command line or in a file. Devices in the boot device list occur in the same order as devices listed on the invocation of this command.

It is strongly recommended that if more than one device is to be entered into the device list, the -f File flag be used. This makes an alterable record of the boot devices available for reference or future update. When the -f File flag is used, the list of devices is taken from the file specified by the File variable. Devices from this list are then placed in the boot list in the order found in the file.

The selection of the boot device list to alter is made with the -m Mode option, where the mode variable is one of the keywords service, normal, both, or prevboot. If the prevboot keyword is specified, only the -i (invalidate) flag may be specified. If the both keyword is specified, then the service list and the normal list will contain exactly the same information. The -i flag invalidates the device list specified by the -m flag. The -f File flag can be used to specify device names in a file. Use of this option allows for future querying and updating of the devices listed in the file.

Device Choices

The device name specified on the command line (or in a file) can occur in one of two different forms:

- It can indicate a specific device by its device logical name.
- It can indicate a generic or special device type by keyword.

The following generic device keywords are supported:

fd Any standard I/O-attached diskette drive

scdisk Any SCSI-attached disk (including serial-link disk drives)

badisk Any direct bus-attached disk (Model 320 only)

cd Any SCSI-attached CD-ROM

rmt Any SCSI-attached tape device

ent Any Ethernet adapter

tok Any Token-Ring adapter

fddi Any Fiber Distributed Data Interface adapter

When a specific device is to be included in the device list, the device's logical name (used with system management commands) must be specified. This logical name is made up of a prefix and a suffix. The suffix is generally a number and designates the specific device. The specified device must be in the Available state. If it is not, the update to the device list is rejected and this command fails. The following devices and their associated logical names are supported (where the prefix is the device type and the **xx** variable is the device-specific suffix):

fdxx Diskette-drive device logical names

hdiskxx Physical-volume device logical names

cdxx SCSI CD-ROM device logical names

rmtxx Magnetic-tape device logical names

entxx Ethernet-adapter logical names

tokxx Token-ring adapter logical names

fddixx Fiber Distributed Data Interface adapter logical names

Attribute Choices

Attributes are extra pieces of information about a device the user supplies on the command line. Since this information is specific to a particular device, generic devices do not have attributes. Attributes apply to the device that immediately precedes them on the command line, which allows attributes to be interspersed among devices on the command line. Currently, only network devices have attributes. These are:

bserver IP address of the BOOTP server

gateway IP address of the gateway

client IP address of the client

hardware Hardware address

These attributes can be combined in the following ways:

- The hardware attribute cannot be specified alone; it must be specified with the bserver or gateway attribute. When specified with bserver or gateway, it applies to the server or gateway, respectively; when both bserver and gateway are specified, hardware will apply to gateway.
- The bserver attribute can be specified alone, with hardware, and/or gateway.
- If the gateway attribute is specified, bserver and client must also be specified.
- The client attribute can only be specified with gateway and bserver.

The syntax for specifying an attribute is attr=value, where attr is the attribute name, value is the value, and there are no spaces before or after the =.

File Format When Using the -f Flag

Attention

Care must be taken when specifying the possible boot devices. A future reboot in Normal mode may fail if the devices specified in the device list become unbootable. A diskette boot is always available when the keylock is in the Service position. The system must not be powered off or reset during the operation of the `bootlist` command. If the system is reset, or power fails at a critical point in the execution of this command, a checksum error can cause the system setup information in battery-backed-up RAM to be lost.

The file specified by the file variable should contain device names separated by white space:

```
hdisk0 hdisk1 cd1
```

or one device per line:

```
hdisk0
hdisk1
cd1
```

Error Handling

If this command returns with an error, the device lists are not altered. The following device list errors are possible:

- If the user attempts to add too many devices to the boot list, the command fails, indicating that too many devices were requested. The number of devices supported varies depending on the device selection.
- If an invalid keyword, invalid flag, or unknown device is specified, the command fails with the appropriate error message.
- If a specified device is not in the Available state, the command fails with the appropriate error message.

Security

Only the root user and members of the security group should have execute permission for this command.

3.24 bosboot Command

Creates a boot image. In AIX V4.2, flags **-v** and **-T** were added.

Syntax:

To Create a Device Boot Image:

```
bosboot {-a|-v} [-d Device] [-p Proto]
            [-k kernel] [-U] [-I|-D]
            [-l LVdev] [-L] [-M {Norm|Serv|Both} ] [ -O Number ]
            [-T Type] [-b FileName] [-q]
```

To Copy a Device Boot Image:

```
bosboot -w file [-d device] [-q]
```

To Create a ROS Emulation Boot Image:

```
bosboot -r file [-d device] [-l lvdev] [-M serv|norm|both] [-q]
```

Flags:

-a Create boot image and write to device or file.

-w File Copy given boot image file to device.

-r File Create ROS emulation network boot image.

-v **Verify, but do not build boot image.**

-d Device Device for which to create the boot image.

-U Create uncompressed boot image.

-p Proto Use the given Proto file for RAM disk file system.

-k kernel Use the given kernel file for boot image.

-l lvdev Target boot logical volume for boot image.

-b File Use the given file name for boot image name.

-D Load Low Level Debugger.

-I Load and Invoke Low Level Debugger.

-M norm|serv|both Specifies the boot mode.

-O Offset Boot image offset for CDROM file system.

-T Platform Specifies the hardware platform type.

-q Query disk space required to create boot image.

-L Enable MP locks instrumentation.

Description

The bosboot command creates the boot image that interfaces with the machine boot ROS (Read-Only Storage) EPROM (Erasable Programmable Read-Only Memory).

The bosboot command creates a boot file (boot image) from a RAM (Random Access Memory) disk file system and a kernel. This boot image is transferred to a particular media that the ROS boot code recognizes. When the machine is powered on or rebooted, the ROS boot code loads the boot image from the media into memory. ROS then transfers control to the loaded image's kernel.

The associated RAM disk file system contains device configuration routines that make the machine's devices and file systems available. The RAM disk file system contains differing configuration files depending upon the boot device. A mkfs prototype file is supplied for each type of device, (see note below). Currently supported devices are:

- CD-ROM
- Disk
- Tape
- Network

A network device may be a token ring, Ethernet, or Fiber-Distributed Data Interface (FDDI) used to boot from a network boot server over a local area network (LAN).

The boot image varies for each type of device booted and is compressed by default to fit on certain media and to lessen real memory requirements. The boot image can be left uncompressed by specifying the -U flag. The logical volume must be large enough for the uncompressed boot image.

In addition to creating a boot image, the bosboot command always saves device configuration data for disk. It does not update the list of boot devices in the NVRAM (nonvolatile random access memory). You can modify the list with the bootlist command.

The bosboot command is normally called during the Base Operating System installation and by the updatep command when the operating system is upgraded.

Notes:

1. You must have root user authority to use the bosboot command.
2. Do not reboot the machine if the bosboot command is unsuccessful with a message not to do so while creating a boot disk. The problem should be resolved and the bosboot command run to successful completion.
3. The bosboot command requires some space in the /tmp file system and the file system where the target image is to reside, if there is such an image.
4. The bosboot command requires that the specified physical disk contain the boot logical volume. To determine which disk device to specify, issue the following command: lsvg -M rootvg. This command displays a map of all logical volumes. The default boot logical volume is hd5. Use the disk device that contains the boot logical volume.
5. When the device is not specified with the -d flag, the bosboot command assumes the default device is the disk the system is booted from. However, if the prototype file is specified with a -p flag, the device must also be specified with a -d flag.
6. The prototype file used by the bosboot command to build the RAM disk file system depends on the boot device and the hardware platform (sys0) type of the machine the boot image will run on. The hardware platform type is an abstraction which allows machines to be grouped according to fundamental configuration characteristics such as number of processors and/or I/O bus structure. Machines with different hardware platform types will have basic differences in the way their devices are dynamically configured at boot time.

The hardware platform type rs6k applies to all Micro Channel-based uni-processor models. The type rs6ksmp applies to all Micro Channel-based symmetric multi-processor models. The type rspc applies to all ISA-bus models. As new models are developed, their hardware platform types will either be one of the aforementioned types or, if fundamental configuration differences exist, new types will be defined. Boot images for a given boot device type will generally be different for machines with different hardware platform types.

The prototype file used by bosbootis constructed by starting with a copy of the base prototype file for the platform type and boot device (for example /usr/lib/boot/rs6k.disk.proto). Next the bosboot command looks at the pcfg file for the platform type being used (for example /usr/lib/boot/rs6k.pcfg). The pcfg file contains entries which uses in a template to search for proto extension files. These files, located in the directory /usr/lib/boot/protoext, provide extensions to the prototype file under construction. As an example, if the platform type is rs6k and the boot device is disk, and the file /usr/lib/boot/protoext/rs6k.pcfg contains the following lines:

```
mca.
scsi.
```

The bosboot command will start with the base prototype file /usr/lib/boot/rs6k.disk.proto and search the directory /usr/lib/boot/protoext for any files that match the template disk.proto.ext.mca.*. The contents of these files are added to the prototype file under construction. Next, the contents of files matching the template /usr/lib/boot/protoext/disk.proto.ext.scsi.* are added to the prototype file under construction. This continues until all lines in the pcfg file have been processed. At this point the prototype file under construction is complete. The bosboot command passes this prototype file to the mkfs command which builds the RAM disk file system.

7. In AIX Version 3.2.5, the prototype files used by the bosboot command to build boot images were dependent on the boot device. This is still true in AIX Version 4.1.3, but in addition, the prototype files are dependent on the system device type (sys0) of the machine for which the boot image is built. This is reflected in the names of the prototype files:

   ```
   /usr/lib/boot/rs6k.disk.proto
   /usr/lib/boot/rs6ksmp.disk.proto
   /usr/lib/boot/rs6k.tape.proto
   /usr/lib/boot/rs6ksmp.tape.proto
   /usr/lib/boot/rs6k.cd.proto
   /usr/lib/boot/rs6ksmp.cd.proto
   /usr/lib/boot/network/rs6k.tok.proto
   /usr/lib/boot/network/rs6ksmp.tok.proto
   /usr/lib/boot/network/rs6k.ent.proto
   /usr/lib/boot/network/rs6ksmp.ent.proto
   /usr/lib/boot/network/rs6k.fddi.proto
   /usr/lib/boot/rspc.disk.proto
   /usr/lib/boot/rspc.cd.proto
   /usr/lib/boot/network/rspc.tok.proto
   /usr/lib/boot/network/rspc.ent.proto
   ```

   The system device type is an abstraction that allows machines to be grouped according to fundamental configuration characteristics, such as number of processors and I/O bus structure. The system device is the highest-level

device in the system node, which consists of all physical devices in the system.

Machines with different system device types have basic differences in the way their devices are dynamically configured at boot time. The system device type RS6K applies to all of the RISC System/6000 models supported by AIX versions prior to 4.1. The type RS6KSMP applies to symmetric multiprocessor models.

Note: The `bootinfo` command does not apply to AIX Version 4.2 or later.

Security

Only the root user should have read and execute permission for this command.

3.25 Additional Printers Support

Additional printers supported in AIX V4.2.

- IBM
  - IBM 3130 LaserPrinter
  - IBM 4247 Printer
  - IBM 6400 Printer
- Hewlett Packard
  - Hewlett Packard LaserJet 5Si/5Si MX
- Lexmark
  - Lexmark Optra Plus LaserPrinter
  - Lexmark Optra C Color LaserPrinter
  - Lexmark Optra E LaserPrinter
  - Lexmark Optra N LaserPrinter
  - Lexmark ExecJet IIc
  - Lexmark ValueWriter 600
  - Lexmark 4039 Plus LaserPrinter
  - Lexmark 4079 Color JetPrinter Plus

Chapter 4. Binary Compatibility

Ensuring binary compatibility between AIX Version 4.1 and Version 4.2 was one of the main priorities during development of AIX V4.2.

The design goal during development was: *With few exceptions - If the application installs and runs on AIX Version 4.1, it will install and run unchanged on AIX Version 4.2*.

4.1 Known Exceptions to Compatibility

Binaries which contain the following features will not be binary compatible with AIX Version 4.2.

- Non-shared compiles of AIX shared libraries

  This means that if you use the -bnso option to the linker, there is no guarantee that the binary will run on an AIX Version 4.2 system. This is not a global interdiction, it is possible that staticly linked binaries will run on an AIX Version 4.2 system.
- Features explicitly described as non-portable by IBM in the AIX Version 4.1 reference manuals.
- Non-documented AIX internal features for example API's and/or data structures.

Another area where problems may occur which is not directly related to AIX 4.2 is the use of hardware specific compiler options. Binary compatibility does not necessarily exist between applications compiled using POWER2 or PowerPC specific compiler options but executed on models other than POWER2 or PowerPC platforms respectively.

Important

Any program that must run in all environments -- POWER, POWER2 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 PowerPC-based technology must be run on PowerPC-based processors. Existing binaries need not be recompiled to operate on the target processors.

4.2 Backward Compatibility between AIX V4.2 and AIX V4.1

There is no global backward compatibility between Version 4.2 and Version 4.1. If you have a mixed environment with systems running both versions of the operating system, you should compile your application on an AIX Version 4.1 platform, this will allow it to run without problems in both environments.

If for any reason, you have no other option but to develop your application on a Version 4.2 system and be able to run it on a 4.1 system there are a number of common problems you are likely to encounter.

© Copyright IBM Corp. 1996

You have compiled and linked your application on an AIX Version 4.2 system. Everything runs fine on your AIX V4.2 system. When you try to execute it on an AIX Version 4.1 system, you receive a message similar to the following :

```
/home/ausres9/BINARY.itsorus> ./foo
     0509-037 System error - error data is: ./foo
     0509-023 Symbol __mod_init in ksh is not defined.
     0509-023 Symbol __crt0v in ksh is not defined.
     0509-026 System error: Cannot run a file that does not have a valid
     format.
```

The problem here is that the symbol __crt0v appears in the /lib/crt0. o file in AIX Version 4.2 but not in AIX Version 4.1. To get around this you can do the following :

Create a directory on your AIX Version 4.2 system, /usr/local/env41 for example and copy /lib/crt0.o and similar files (/lib/crt0.o, /lib/gcrt0.o, /lib/mcrt0_r.o, /lib/crt0_r.o, /lib/gcrt0_r.o, /lib/mcrt0.o), to this directory. You can then modify the compiler configuration file to create an AIX Version 4.1 environment.

To do this you will need to create an entry in the /etc/xlC.cfg file for a new compiler, that you can call cc41. The xlC.cfg file defines what the default options are for a particular compiler depending upon the name by which it is called. To define this new cc41 compiler, do the following:

- Link the C compiler to the cc41 command

  ```
  # ln -s /bin/cc /bin/cc41
  ```

- Add a new entry for cc41 in the /etc/xlC.cfg file

  Copy the stanza for the cc compiler:

  ```
  cc:     use        = DEFLT
    crt        = /lib/crt0.
    mcrt       = /lib/mcrt0.o
    gcrt       = /lib/gcrt0.
          libraries  = -lc
          proflibs   = -L/lib/profiled,-L/usr/lib/profiled
          options    = -qlanglvl=extended,-qnoro,-qnoroconst
  ```

  and modify it to:

  ```
  cc41:     use        = DEFLT
    crt        = /usr/local/env41/crt0.o
    mcrt       = /usr/local/env41/mcrt0.o
    gcrt       = /usr/local/env41/gcrt0.o
          libraries  = -lc
          proflibs   = -L/lib/profiled,-L/usr/lib/profiled
          options    = -qlanglvl=extended,-qnoro,-qnoroconst
  ```

You can then compile using cc41 as the name of you compiler. Note that the executable you produce should also run on your AIX Version 4.2 system. If you use cc you will produce strictly AIX Version 4.2 code.

The above workaround will only work in certain cases. In other cases you will discover that some libraries in Version 4.2 have had the names of their modules changed since AIX Version 4.1. This is true of the curses library. If you compile

a program that uses the curses library, and then try to execute it on an AIX Version 4.1 system you will receive a message similar to the following:

```
/home/ausres9/BINARY.itsorus> ./foo
     0509-037 System error - error data is: ./foo
     0509-027 Member shr42.o is not found or file is not an archive.
     0509-027 Member shr42.o is not found or file is not an archive.
     0509-022 Cannot load library libcurses.a·shr42.o".
     0509-026 System error: A file or directory in the path name does
     not exist.
```

The workaround in this case is to copy the curses library from the AIX V4.1 system to the AIX V4.2 system, into /usr/local/env41 for example. Then compile with a command similar to:

```
cc41 -o foo foo.c -L/usr/local/env41 -lcurses <any other options>
```

This program should then run on the AIX Version 4.1 system.

Libraries that have been found to be incompatible between AIX Version 4.2 and AIX Version 4.1 are:

- The C++ library (libC.a)
- The curses library (libcurses.a)
- The extended curses library libxcurses.a)

Important

Please note that these are *not* IBM supported solutions. This information is provided purely as a suggestion for *last resort* situations and users will need to test each resulting executable on an individual basis to measure its compatibility in their own particular environment.

Chapter 5. NIM Enhancements

Enhancements to the NIM environment in AIX Version 4.2 have concentrated on ease of use. The SMIT panels have changed, there is a new quick setup facility and a utility to allow easy definition of multiple client systems has been added.

5.1 NIM Quick Setup

In AIX Version 4.1, configuring a NIM environment was a time consuming and laborious process, requiring considerable expertise. This was true even for a simple network install environment where the flexibility offered was not always needed.

For AIX Version 4.2 the new NIM Easy Startup facility hides much of the complexity of NIM configuration by providing a method for simple configuration of a sensible default environment. This allows a less experienced systems administrator to set up and configure a NIM environment much more easily than before.

The new SMIT Easy Startup panel has a fastpath of `nim_config_env`, and is shown in Figure 6 on page 110. It allows the systems administrator to set up a basic NIM environment by supplying a minimum of two pieces of information, namely:

- Input device for installation images
- Primary network interface

Default values, which can be overridden, are provided for the remaining options.

On successful completion of this one SMIT panel, the following actions will have been completed:

- NIM master initialized on the primary network interface
- NIM daemons running
- lpp_source resource created and available
- SPOT resource created and available

When the user selects the default action of creating filesystems for the lpp_source and SPOT resources, the new filesystems are created with the specified sizes. During creation of the resources, the filesystems will be expanded automatically as required for the resources to fit.

Creating a filesystem for each resource makes system storage management easier, particularly in environments where resources are added and removed frequently.

Setting up resources for diskless and dataless machines, and configuring the list of NIM clients from a stanza file can also be carried out at the same time from this SMIT panel.

When you want to perform either of these tasks after you have already used the SMIT Easy Startup panel, they should be performed from the SMIT Advanced Configuration panel. This is because the SMIT Easy Startup panel configures the NIM master, and a second attempt to configure the master will result in an error.

© Copyright IBM Corp. 1996

When the primary network is token ring, and the machine that is to be the NIM master has a PCI bus token-ring card, ensure that the ring speed of the adapter is set to the correct value for the network. Although the PCI token ring adapter supports *autosense* as a valid ring speed, NIM only recognizes *16* and *4* as valid ring speeds. If the ring speed is set to *autosense*, an error will occur when trying to configure the NIM master.

```
                Configure a Basic NIM Environment (Easy Startup)

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

                                                      [Entry Fields]
  Initialize the NIM Master:
* Primary Network Interface for the NIM Master       []                        +

  Basic Installation Resources:
* Input device for installation images               []                        +
* LPP_SOURCE Name                                    [lpp_source1]
* LPP_SOURCE Directory                               [/export/lpp_source]      +
    Create new filesystem for LPP_SOURCE?            [yes]                     +
    Filesystem SIZE (MB)                             [200]                      #
    VOLUME GROUP for new filesystem                  [rootvg]                  +
* SPOT Name                                          [spot1]
* SPOT Directory                                     [/export/spot]            +
    Create new filesystem for SPOT?                  [yes]                     +
    Filesystem SIZE (MB)                             [200]                      #
    VOLUME GROUP for new filesystem                  [rootvg]                  +

  Create Diskless/Dataless Machine Resources?        [no]                      +
  Specify Resource Name to Define:
    ROOT   (required for diskless and dataless)      [root1]
    DUMP   (required for diskless and dataless)      [dump1]
    PAGING (required for diskless)                   [paging1]
    HOME        (optional)                           [home1]
    SHARED_HOME (optional)                           [shared_home1]
    TMP         (optional)                           [tmp1]
  Diskless/Dataless resource directory               [/export/dd_resource]
    Create new filesystem for resources?             [yes]                     +
    Filesystem SIZE (MB)                             [60]                       #
    VOLUME GROUP for new filesystem                  [rootvg]                  +

  Define NIM System Bundles?                         [yes]                     +

  Add Machines from a Definition File?               [no]                      +
  Specify Filename                                   []

* Remove all newly added NIM definitions            [no]                      +
  and filesystems if any part of this
  operation fails?

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

Figure 6. NIM, Easy Startup SMIT Panel

Each of the tasks featured in the SMIT Easy Startup panel can be performed individually from the SMIT Advanced Configuration panel. This panel is shown in Figure 7 on page 111, and has a fastpath of nim_config_adv. This method of configuration is similar to the method used in AIX Version 4.1 but has been enhanced to group related tasks together. Examples of these enhancements

include creating a SPOT and lpp_source at the same time and defining multiple NIM clients in one operation by processing a stanza file.

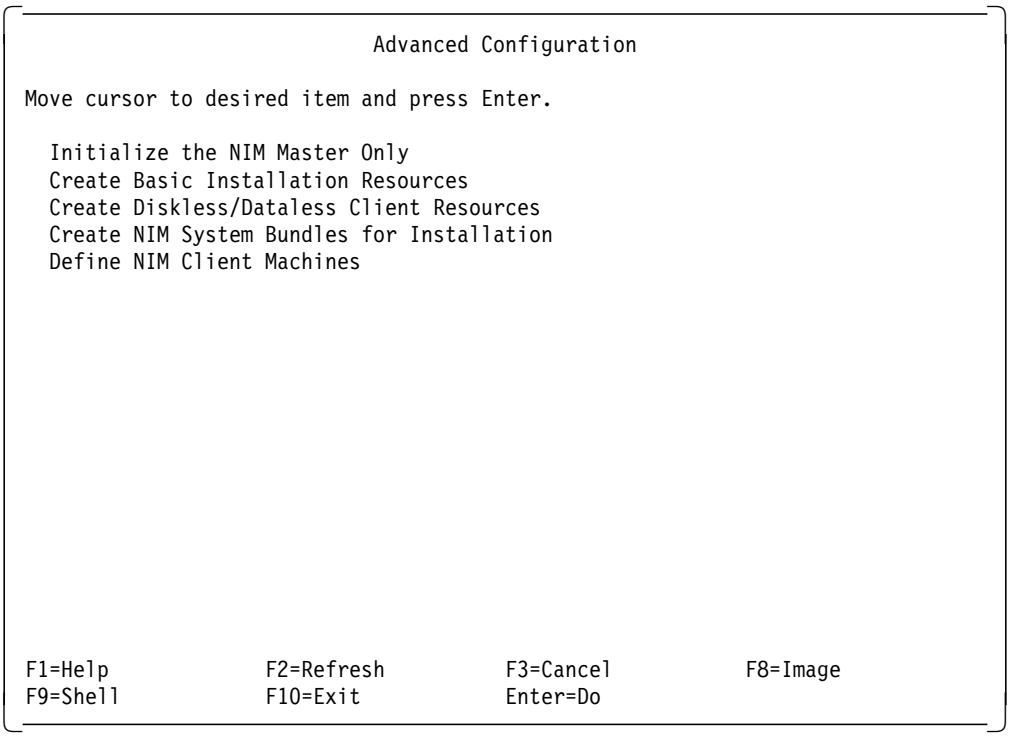


Figure 7. NIM, Advanced Configuration SMIT Panel

5.2 NIM Client Definition

In AIX Version 4.1 each NIM client machine had to be defined in the NIM environment in a separate operation. In a network with a large number of NIM client systems, this would be a very time consuming and laborious task.

AIX Version 4.2 has added the facility to define multiple NIM client machines in a single operation. The `nimdef` command parses a definition stanza file to build the list of commands required to add NIM client definitions to the NIM environment. The `nimdef` command can be instructed to parse the definition file and report errors, invoke the commands it generates, or it can display the commands as output which can be redirected to a file and invoked at a later time.

A client definition file consists of a number of machine definitions. Each machine definition consists of a stanza header, which is the machine hostname delimited by a colon, followed by a number of attribute lines. If the keyword `default` is used in a stanza header, it indicates that the stanza is defining default values for the given attributes.

The required attributes for a machine definition are:

machine_type The type of the machine. (standalone,diskless, dataless)

network_type The type of the machines network adapter.

ring_speed Required if machine uses token ring.

cable_type Required if machine uses Ethernet.

gateway Hostname or IP address of default gateway used by the machine.

subnet_mask The subnet mask used by the machine.

Other keywords are recognized as optional attributes. The optional attributes are:

nim_name The NIM name that should be used by a machine. This is required to avoid a name conflict if non-unique names are used in different domains. By default the NIM name is the hostname of the machine with any domain information stripped off.

platform The machine hardware platform. Defaults to rs6k if not specified.

net_adptr_name
The name of the network adapter used by the machine. For example, tok0.

netboot_kernel The type of kernel to use when booting the kernel. Possible values are up or mp.

ipl_rom_emulation
The device to use for IPL ROM emulation.

primary_interface
The hostname that this machine is initially defined under, if this stanza definition is being used to specify an additional interface.

master_gateway
The gateway the NIM master uses to reach this machine if this machine is on a different network. This attribute is not necessary if this machine is defined on a network that is already defined in the NIM environment, or if the NIM master network has a default gateway specified.

machine_group The NIM group or groups the machine should be added to. The NIM group will be defined if it does not already exist.

comment A comment to be included in the machine definition. The comment string should be in double quotes.

While parsing the definition file, if a machine is defined on a network which already exists in the NIM environment, the machine will be defined on that network. If a machine is defined on a network which does not exist in the NIM environment the NIM network object will be defined assuming that the required information is contained in the definition file.

A sample machine definition file is shown in Figure 8 on page 113.

```
# This is an example definition file
# set the defaults
default:
  machine_type=standalone
  subnet_mask=255.255.255.0
  gateway=itsorus
  network_type=tok
  ring_speed=16
  platform=rspc

# define the machine aix42hw
# not much needed, as most things picked up from defaults
aix42hw.itsc.austin.ibm.com:
    platform=rs6k

# define the machine vera
vera.itsc.austin.ibm.com:
    platform=rs6k
    ipl_rom_emulation=/dev/fd0
```

Figure 8. NIM, Sample Machine Definition File

5.3 Network Topology Enhancements

Enhancements to networking within NIM have removed the restriction whereby each client and server had to have a static route defined before communication was possible. The SMIT panels for defining client systems have also been improved with certain parameters being determined automatically whenever possible.

5.3.1 Default Routes

In AIX Version 4.1 all NIM routing was static. For each client system and each server it was necessary to specify which machines were gateways in order to establish a route for the client to communicate with the server. AIX Version 4.2 adds the capability to define default routes. Default routes more closely model the network configuration of common network environments. The NIM server is configured with a default route attribute for each NIM network object. It no longer needs to have details of a static route to each client, although this capability is provided for compatibility purposes.

Default routes are not required if all networks in a NIM environment are associated with interfaces defined on the NIM master, and if all resources are defined on the NIM master.

5.3.2 Automatic Selection of NIM Networks

The procedure for defining a NIM client machine in AIX Version 4.1 was as follows:

1. Define the NIM network the client is connected to, if it has not already been defined.
2. Define the route between the client network and a network interface defined on the master, if it has not already been defined.
3. Define the client, which includes selecting the correct NIM network.

This required the system administrator to use up to three SMIT panels to define one client machine.

The process of defining a NIM client machine has been enhanced in AIX Version 4.2. One SMIT panel is used to define a NIM client machine. The initial dialogue prompts for the fully qualified hostname of the client machine. The NIM system resolves the name to an IP address which is used to determine whether a NIM network has been defined for the subnet the client is connected to. If the NIM network has been defined, the second dialogue panel of the SMIT screen will already contain details of the network. This is shown in Figure 9.

```
                                Define a Machine

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

                                                      [Entry Fields]
* NIM Machine Name                                   [aix42inst]
* Machine Type                                       [standalone]            +
* Hardware Platform Type                             [rs6k]                  +
  Kernel to use for Network Boot                     [up]                    +
  Primary Network Install Interface
*   Ring Speed                                       []                       #
*   NIM Network                                       network1
*   Host Name                                         aix42inst
    Network Adapter Hardware Address                 [0]
    Network Adapter Logical Device Name              []
  IPL ROM Emulation Device                           []                      +/
  CPU Id                                             []
  Machine Group                                      []                      +
  Comments                                           []


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

Figure 9. NIM, Define a Machine SMIT Panel (when NIM network exists)

If the NIM network has not been defined, the user will be prompted for the type of network the client is connected to. The dialogue box is shown in Figure 10.

```
      Type of Network Attached to Primary Network Install Interface

 Move cursor to desired item and press Enter.

   tok  = token ring network
   ent  = ethernet network
   fddi = FDDI network

 F1=Help                 F2=Refresh              F3=Cancel
 F8=Image                F10=Exit                Enter=Do
 /=Find                  n=Find Next
```

Figure 10. NIM, SMIT Dialogue Box for Choosing Network Interface

The next part of the dialogue will prompt for details of the default route and subnet mask of the NIM network in addition to details of the NIM client. This is

so that the NIM network can be defined automatically as part of the process of defining the NIM client. This is shown in Figure 11 on page 115.

```
                                Define a Machine

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

                                                      [Entry Fields]
* NIM Machine Name                                   [www]
* Machine Type                                       [standalone]             +
* Hardware Platform Type                             [rs6k]                   +
  Kernel to use for Network Boot                     [up]                     +
  Primary Network Install Interface
*   Ring Speed                                       []                        #
*   NIM Network                                      [tok-Network3]
*     Network Type                                    tok
*     Subnetmask                                     []
*     Default Gateway Used by Machine                []
*     Default Gateway Used by Master                 [9.3.1.74]
*   Host Name                                         www.austin.ibm.com
    Network Adapter Hardware Address                 [0]
    Network Adapter Logical Device Name              []
  IPL ROM Emulation Device                           []                       +/
  CPU Id                                             []
  Machine Group                                      []                       +
  Comments                                           []



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

Figure 11. NIM, Define a Machine SMIT Panel (when NIM network does not exist)

The panel prompts for information on the default gateway used by the machine being defined. This is so the NIM network can be defined with a default route, which applies to all machines on this network. See 5.3.1, “Default Routes” on page 113 for more information.

5.3.3 New resolv_conf Resource

AIX Version 4.2 has added the resolv_conf resource. This resource represents the location of a file that contains valid /etc/resolv.conf entries which define Domain Name Protocol nameserver information. If a resolv_conf resource is allocated to a standalone machine before a bos_inst operation, upon successful installation and reboot, the machine will be configured to use the domain name services defined by the resource.

The resolv_conf resource is only supported when allocated to a standalone client prior to a bos_inst of AIX Version 4.2 or above. Manual customization is still necessary when configuring Doman Name Services for AIX Version 4.1 NIM clients.

5.4 NIM Client Groups

In AIX Version 4.1, it was necessary to configure the environment for each NIM client separately, and then perform the NIM operations on each client in turn.

In AIX Version 4.2, Client Groups can be set up for groups of client systems with similar hardware configurations, on which identical operations need to be performed at the same time.

The concept of Client Groups was originally envisaged for groups of identical diskless workstations, to allow them to be managed more easily and treated as a single entity. AIX Version 4.2 implements this concept for all NIM client systems, not just diskless systems.

```
                              Define a Machine Group

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

                                                      [Entry Fields]
* Group Name                                         [itso_lab_machines]
* Group Type                                          mac_group
  Member Type                                         standalone
* Group Members                                      [aix42hw aix42inst]      +
  Comments                                           [AIX 4.2 Lab machines]

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

Figure 12. NIM, Define a Machine Group SMIT Panel

Once a machine group has been defined, NIM operations can be initiated on all members of the group at the same time. The list of available options will depend upon the type of machines in the group.

For example, Figure 13 on page 117 shows the operations which can be performed on a group of standalone machines, while Figure 14 on page 117 shows the operations which can be performed on dataless machines.

```
                          Operation to Perform

Move cursor to desired item and press Enter.

     diag       = enable a machine to boot a diagnostic image
     cust       = perform software customization
     bos_inst   = perform a BOS installation
     maint      = perform software maintenance
     reset      = reset an object's NIM state
     fix_query  = perform queries on installed fixes
     check      = check the status of a NIM object
     reboot     = reboot specified machines
     maint_boot = enable a machine to boot in maintenance mode
     showlog    = display a log in the NIM environment
     showres    = show contents of a resource
     lppchk     = verify installed filesets
     update_all = update all currently installed filesets

F1=Help                 F2=Refresh              F3=Cancel
F8=Image                F10=Exit                Enter=Do
/=Find                  n=Find Next
```

Figure 13. NIM, Standalone Machine Group Operations

```
                          Operation to Perform

Move cursor to desired item and press Enter.

     dtls_init  = initialize a dataless machine's environment
     diag       = enable a machine to boot a diagnostic image
     reset      = reset an object's NIM state
     check      = check the status of a NIM object
     showlog    = display a log in the NIM environment




F1=Help                 F2=Refresh              F3=Cancel
F8=Image                F10=Exit                Enter=Do
/=Find                  n=Find Next
```

Figure 14. NIM, Dataless Machine Group Operations

You do not have to perform a NIM operation on all members of a machine group. Machines can be *excluded* from NIM operations, yet remain in the group.

For example, you may wish to reboot all standalone machines, apart from those in a particular room. This can be achieved by excluding the desired machines from the machine group before performing the NIM operation. The machines are still part of the group, but are excluded from NIM operations.

A machine which has been *excluded* from a machine group will not be affected by NIM operations on the group until it has been *included* in the group again.

5.5 NIM Resource Groups

Resource Groups are similar in concept to Client Groups. Instead of grouping machines, they are used to group together a set of resources which logically belong together. The group of resources can then be handled as a single entity and assigned to client systems or groups for NIM operations.

As an example, a resource group for installing AIX Version 4.2 on machines in a software development environment could consist of the following resources:

- spot42 - A SPOT resource.
- lpp_source42 - The location of optional product images.
- SW_dev - An installp_bundle resource, which defines the additional filesets to be installed on the machine.
- bosinst42 - A bosinst_data resource to enable unprompted installation.
- resolv_conf_itso - A resolv_conf resource to configure the name resolution of the machine after successful installation.

Once this resource group has been defined, it can be used to perform NIM operations on NIM clients. When a NIM client is to be installed with a software development environment, it can be installed using the resource group, rather than having to explicitly list each component resource.

```
                          Define a Resource Group

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

                                                      [Entry Fields]
* Group Name                                         []
  SPOT       (Shared Product Object Tree)            []                      +
  LPP_SOURCE (source for optional product images)    []                      +
  INSTALLP_BUNDLE (an installp bundle file)          []                      +
  SCRIPT       (file which is executed on clients)   []                      +
  BOSINST_DATA (config file for bos_inst operation)  []                      +
  IMAGE_DATA   (config file for bos_inst operation)  []                      +
  RESOLV_CONF  (config file for name-server info.)   []                      +
  MKSYSB       (an AIX mksysb image)                 []                      +
  FIX_BUNDLE   (fix keyword input file)              []                      +
  ROOT   (parent dir. for client / (root) dirs.)     []                      +
  PAGING (parent dir. for client paging files)       []                      +
  DUMP   (parent dir. for client dump files)         []                      +
  HOME   (parent dir. for client /home dirs.)        []                      +
  SHARED_HOME (/home dir. shared by clients)         []                      +
  TMP    (parent dir. for client /tmp dir)           []                      +
  Use this Group for Default Allocation?             []                      +
  Comments                                           []


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

Figure 15. NIM, Define a Resource Group SMIT Panel

Similarly, if a number of machines in a machine group are to be installed with a software development environment, the NIM operation can be set up by allocating the resource group to the machine group. In AIX Version 4.1, this type

of operation would require each individual resource to be allocated to each individual machine.

For example, allocating five resources to five machines would require 25 allocations. In AIX Version 4.2, after the initial setup of the machine group and resource group had been performed, this task would require one allocation.

It is possible to define a resource group to be the default resource group. This allows NIM operations which require resources to be performed on NIM clients which have not had resources allocated to them.

5.6 Task Oriented Interface

In AIX Version 4.1 the SMIT panels for NIM software installation and maintenance are very different to the SMIT interface for local software installation. NIM adopted an object oriented style interface, whereas the standard SMIT interface is task oriented.

For AIX Version 4.2 the SMIT panels for NIM software installation have been changed to mimic the standard SMIT interfaces as closely as possible and for the equivalent NIM operation to produce the same results as the standard install operation. Figure 16 on page 120 shows the top level SMIT panel for NIM software installation and maintenance. It can be distinguished from the SMIT panel for local software installation by the title of the panel and the fact that the last menu option mentions diskless and dataless machines.

Other software installation and maintenance tasks which can be performed on NIM components include installing bundles of software, updating systems to the latest level via `update_all` and updating software by fix (APAR). The SMIT panel for this NIM task is shown in Figure 17 on page 120. It can be distinguished from the SMIT panel for local software installation by the fact that the first menu option is to install software on standalone clients.

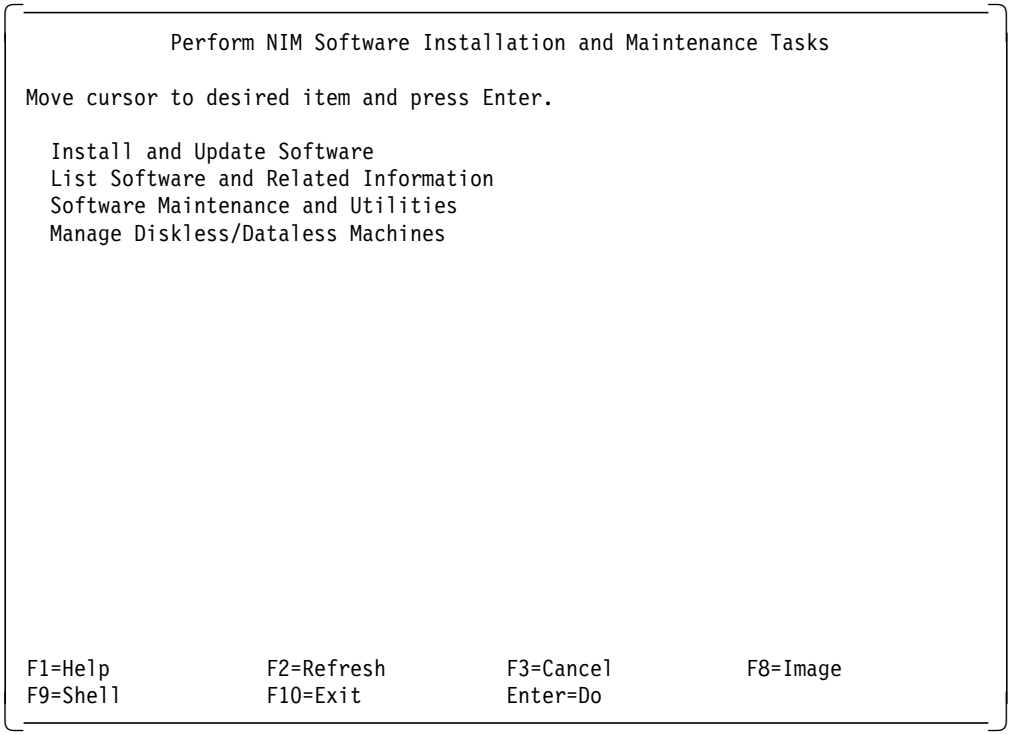


Figure 16. NIM, Perform NIM Software Installation SMIT Panel

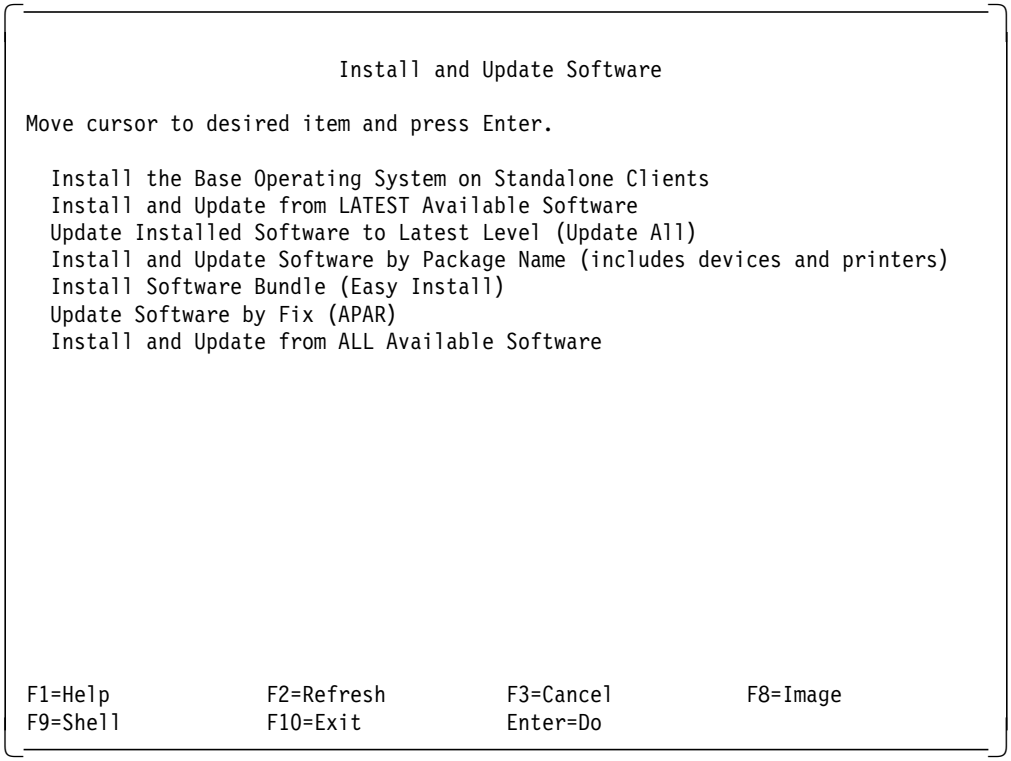


Figure 17. NIM, Install and Update Software SMIT Panel

5.7 NIM Remote Backup

In AIX Version 4.1 it was possible to define a mksysb resource to NIM but the mksysb image must have been previously created for the machine being backed up. It then had to be copied to the desired server.

The process of defining a mksysb resource to NIM has been enhanced to allow creation of the mksysb image at the same time.

The procedure for creating the mksysb resource can be used as a means of making a backup of a machine which does not have a tape drive.

Once a mksysb resource has been created, it can be treated like any other NIM resource. A mksysb resource can be restored on the machine on which the mksysb was taken, or it can be restored on another machine to clone the setup of the original machine. When a mksysb resource is used to clone a system, the missing device software is automatically installed by NIM.

5.8 Software Verification

AIX Version 4.2 NIM has added the facility to run the lppchk command on the software in a SPOT. The following lppchk options are supported:

-f Fast check (file existence, file size)

-c Checksum verification

-v Fileset version consistency check

-l File link verification

-u Update inventory (only valid with -c or -l)

-m [**1|2|3**] Controls level of information

The lppchk operation can be performed with the nim **-**o lppchk command or from SMIT with the nim_check_files fastpath.

```
                          Verify Installed Filesets

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

                                                      [Entry Fields]
  Target Name                                        spot1
  Fileset                                           []                      +
  Verification Mode: (Select one of the following)
     Fileset Versions and Requisites?                yes                    +
     File existence and length (fast check)?         no                     +
     File Checksums?                                 no                     +
     File Links?                                     no                     +

  Verbose Output?                                    no                     +
  Update the Software Database?                      no                     +
     (Checksums and Links Only)




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

Figure 18. NIM, lppchk SMIT Panel

5.9 NIM Port Numbers

In AIX Version 4.2 the default port numbers used by NIM are now registered in the /etc/services file. On AIX Version 4.1 the port numbers were not reserved so it was possible for systems administrators to inadvertently use the same port numbers for other functions.

5.10 NIM Executables

The NIM component of AIX Version 4.2 has been redesigned in such a way that most of the executables now use the new shared library libnim.a. This has considerably reduced the amount of disk space required by the NIM fileset. In addition to the disk space savings, it will also result in smaller fixes for the NIM fileset.

5.11 Network Boot in Maintenance Mode

One of the many differences between AIX Version 4.1 and AIX Version 3 was the removal of the ability to create bosboot diskettes. This introduced a dependency on external CD-ROM or tape drive to enable many RS/6000 machines to boot in maintenance mode. In AIX Version 4.2 NIM has been enhanced to allow both push and pull initiated maintenance mode boots. This function was previously performed by issuing the bos_inst operation on a target. The new maint_boot operation allows the target to bypass the install menus to perform maintenance operations.

5.12 Command Line Interface

The `nim` and `nimclient` commands have been enhanced to allow resource allocation and NIM operation invocation at the same time. It is no longer necessary to allocate resources in a separate step prior to performing the NIM operation although this is supported for compatibility with AIX Version 4.1.

5.13 IPL ROM Emulation Diskette

Certain older models of RS/6000 machines do not have a BOOTP-enabled IPL ROM, and so require IPL ROM emulation media. The instructions given in the NIM documentation on the creation of IPL ROM emulation media contain an error. The error will only be noticed when the NIM master is not a Micro Channel-based machine. The `bosboot` command used to create the emulation media requires an additional option, `-T rs6k`, in order to create an image that will work correctly.

The IPL ROM emulation media will be created correctly if the procedure is carried out from the SMIT panel Create IPL ROM Emulation Media.

Chapter 6. Graphic Enhancements

AIX Version 4.2 provides two new extensions to the Xserver. The first enhancement, the XTEST extension, allows developers to create their own test suites for interaction with graphical programs. The second, called DBE, enables the use of double buffering for X-Window animation. First though, let′s review some of the basic X-Window System components.

6.1.1 X-Windows Architecture Review

The X-Windows architecture consists of three basic components: The client, the protocol and the server.

6.1.1.1 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 him to open the 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 which perform complex operations by calling several elementary subroutines. The most elaborate toolkits such as Athena or Motif, also contain widgets which are program objects with a predefined appearance and behavior - a menu pushbutton for example. The standard toolkits provided with AIX are:

Xt The basic X toolkit with hundreds of standard functions or intrinsics as they are also called

Xaw The Athena Widget set from MIT

Xmu Miscellaneous utilities

Xext The extension library

Xi The input extension library, manages the peripherals that is, keyboard and mouse

Xm The Motif library with 3D-look widgets

It is not necessary for a client to be executing on the same system as the server, the Xlibrary subroutines use the DISPLAY environment variable to determine whether the requests have to be submitted locally or via the network. This is completely transparent to both the application developer and end-user.

6.1.1.2 Protocol

The protocol is a formal description of the conversation which is carried out between the client and the server. The server and clients exchange information using messages. These messages are contained in packets which can be transported over a network if necessary. There are four packet types.

- The Request packet, used by the client to ask the server to perform an action.
- The Reply packet, used by the server to answer the client after a request has been received.

© Copyright IBM Corp. 1996

- The Event packet, used by the server to inform the client that something has happened which requires action by the client
- The Error packet, indicates that something went wrong.

6.1.1.3 Server

The server is the program that runs on the workstation to which a physical graphic display is attached. The Server program, called X, consists of two parts. One part that is device independent (dix) that interprets requests from the 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 and renders the 2D graphic operations defined by the X Window System for the specific display adapter. The loadddx GAI (Graphic Adapter Interface) load module implements this interface. The server modules have strictly-defined functionality and peripheral support, the only way to increase the capability of the X server is via 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 with no extensions) only supports keyboard and mouse as peripherals, if you intend to use other peripherals such as Spaceball or dials or LPFKeys you need to load a specific extension that is capable of handling these devices. The command `xdpyinfo` lists the extensions available for the specified server.

Here is a typical example of the output from the xdpyinfo command:

```
$ xdpyinfo
name of display:    tx7.itsc.austin.ibm.com:0.0
version number:    11.0
vendor string:    International Business Machines Corp. / AGE Logic Inc.
vendor release number:    6000
maximum request size:  65535 longwords (262140 bytes)
motion buffer size:  256
bitmap unit, bit order, padding:    32, MSBFirst, 32
image byte order:    MSBFirst
number of supported pixmap formats:    2
supported pixmap formats:
    depth 1, bits_per_pixel 1, scanline_pad 32
    depth 8, bits_per_pixel 8, scanline_pad 32
keycode range:    minimum 9, maximum 141
number of extensions:    11
    XTestExtension1
    SHAPE
    Multi-Buffering
    XInputExtension
    XTEST
    BIG-REQUESTS
    MIT-SUNDRY-NONSTANDARD
    MIT-SCREEN-SAVER
    SCREEN-SAVER
    SYNC
    XC-MISC
default screen number:    0
number of screens:    1

screen #0:
  dimensions:    1280x1024 pixels (325x260 millimeters)
  resolution:    100x100 dots per inch
  depths (1):    8
  root window id:    0x2d
  depth of root window:    8 planes
  number of colormaps:    minimum 1, maximum 1
  default colormap:    0x2a
  default number of colormap cells:    256
  preallocated pixels:    black 0, white 1
  options:    backing-store NO, save-unders NO
  current input event mask:    0x30003c
    ButtonPressMask          ButtonReleaseMask        EnterWindowMask
    LeaveWindowMask          SubstructureRedirectMask FocusChangeMask
  number of visuals:    1
  default visual id:  0x29
  visual:
    visual id:    0x29
    class:    PseudoColor
    depth:    8 planes
    size of colormap:    256 entries
    red, green, blue masks:    0x0, 0x0, 0x0
    significant bits in color specification:    8 bits
```

Note that after the name of the display and the implementation capabilities, the total number of loaded extensions is displayed, in this case 11. Then all the extensions are listed on seperate lines beginning here with XtestExtension1. In our example we also have the XInputExtension. loaded, which manages peripherals other than the keyboard and mouse, the SHAPE extension that permits rounded windows and SCREEN-SAVER that allows use of a screen-saver or display

lock if the session has been left unattended for more than a certain period of time.

6.1.2 X Server XTEST Extension

The XTEST extension is an API which allows the writing of programs where the interactions between application and user are automated. Some existing commercial products and the xrecorder program found in /usr/lpp/X11/Xamples/aixclients/xrecorder are able to perform some user automation, but this new extension allows a programmer to develop his own test suite in a much easier manner. The capabilities of the API even allow the programmer to manage windows that may have moved between two successive test runs. This provides a much more repeatable and accurate test environment which can be essential for tuning and benchmarking situations.

6.1.2.1 Configuring the XTest Extension

The XTest extension, which is an add-on to the X server, is located in the /usr/lpp/X11/bin/loadXTest file which is part of the X11.base.rte fileset.

To run programs which use the Xtest functions, the Xserver must have been started with the XTest extension loaded. There are a number of ways to configure the X-Windows environment to achieve this.

If you want the extension loaded each time you launch the Xserver:

- Check that the file /usr/lpp/X11/bin/loadXTest exists.
- Add the line:

```
XTEST  /usr/lpp/X11/bin/loadXTest
```

to the /usr/bin/X11/static_ext file.

If the extension only needs to be loaded on demand:

- Check that the file /usr/lpp/X11/bin/loadXTest exists.
- Add the line:

```
XTEST  /usr/lpp/X11/bin/loadXTest
```

to the /usr/bin/X11/dynamic_ext file.

You can then load the XTest extension by adding the `-x XTEST` option to the command line when starting the Xserver:

```
$ xinit -- -x XTEST
```

If you use the CDE environment:

- Check that the file /usr/lpp/X11/bin/loadXTest exists
- Check that the /etc/dt/config/Xservers file exists, if not copy the default file from /usr/dt/config
- Modify the following line in the /etc/dt/config/Xservers file

```
:0   Local local@console /usr/lpp/X11/defaults/xserverrc -T -force :0
```

to:

```
:0   Local local@console /usr/lpp/X11/defaults/xserverrc -x XTEST -T -force :0
```

This extension can also be loaded by modifying the `EXTENSIONS` stanza in the /usr/lpp/X11/defaults/xserverrc file by adding the `-x XTEST` option.

```
EXTENSIONS="-x XTEST"
```

Note that the Xserver will not start if you have specified that the XTest extension should be loaded and the /usr/lpp/X11/bin/loadXTest file does not exist or is corrupt.

Also note, that the Xserver must be restarted if modifications have been made to the above files while the server is running.

Once the Xserver is running, you can use the `xdpyinfo` command to verify that the extension has been loaded correctly. You can also use the API call that checks for the existence of this extension in the server.

6.1.2.2 Programming Concepts

The API is composed of 14 C subroutines:

- XTestQueryExtension()
- XTestCompareCursorWithWindow()
- XTestCompareCurrentCursorWithWindow()
- XTestFakeKeyEvent()
- XTestFakeButtonEvent()
- XTestFakeMotionEvent()
- XTestFakeRelativeMotionEvent()
- XTestSetGContextOfGC()
- XTestSetVisualIDOfVisual()
- XTestDiscard()
- XTestFakeDeviceKeyEvent()
- XTestFakeDeviceButtonEvent()
- XTestFakeProximityEvent()
- XTestFakeDeviceMotionEvent()

All XTest extension functions and procedures, and all manifest constants and macros, start with the string *XTest*. All operations are classified as server/client (Server) or client-only (Client). All routines that have return type status will return non-zero for success and zero for failure. Even if the XTest extension is supported the server may withdraw such facilities arbitrarily; in which case they will subsequently return zero.

Please refer to Info Explorer for a full description of XTest routines.

One good starting point to construct a program using the XTest extension, is to look at the sample provided in /usr/lpp/X11/Xamples/extensions/test. The name of the sample program is xtesttest.c. You may compile it using this command:

```
make xtesttest LDFLAGS="-lXtst -lXi -lXext -lX11" CFLAGS= \
"-I../../../include/X11/extensions -I/usr/include/X11"
```

This sample simulates cursor motion and keystroke manipulations.

6.1.3 X Server DBE Extension

The Double Buffer Extension (DBE) provides a standard way to utilize double buffering within the framework of the X Window system. Double buffering is a technique that is used mainly for animation. If there is only one buffer available for both drawing and displaying an animated scene, called single buffer, then the procedure for successive images is as follows:

1. Load the image into the buffer

2. Display the buffer
3. Clear the buffer
4. Load the new image
5. Display the buffer again
6. And so on

Having to clear the buffer and draw the next frame into the buffer before it can be displayed causes a delay which makes the screen image flicker and any motion to appear jerky. With the introduction of 3D APIs where smooth animation was essential, came the concept of double buffering. When double buffering is used there are two buffers available for the image. While one buffer is being used to display the image, the other buffer can be cleared and then loaded with the next image. When the image has loaded into the second buffer the buffers are switched and the second buffer is displayed while the first is cleared and reloaded. The procedure for successive images then becomes:

1. Draw the image into the first buffer
2. Display the image from the first buffer
3. Clear the second buffer
4. Load the next image into the second buffer
5. Display the image from the second buffer
6. Clear the first buffer
7. Load the next image into the first buffer
8. Display the image from the first buffer
9. And so on

Because the images switch almost instantaneously and the only time consuming operation, filling the buffer, occurs while the user is viewing the previous image, the flicker is eliminated and any motion is much smoother.

Of course, 3D APIs generally take full advantage of the extensive hardware buffering facilities available on the latest 3D capable display adapters. Buffer switching is done automatically by the hardware. With the introduction of AIX Version 4.1, the same double buffering principles were applied to X Windows except that the systems main memory was used to store both buffers. This provided reasonable flicker free animation with the X libraries but was wasteful of memory resources. With AIX Version 4.2, the Double Buffering Extension is able to identify the type of graphic adapter it is addressing and take advantage of any built-in multi-buffering capabilities. This results in reduced memory wasteage and better graphic performance.

6.1.3.1 Configuring the Double Buffer Extension

The DBE extension, is located in the /usr/lpp/X11/bin/loadDBE file which is part of the X11.base.rte fileset.

To run programs using this extension the X server must have been started with the extension loaded. There are a number of ways to configure the X Windows environment to achieve this.

If you want the extension loaded each time you launch the X Server:

- Check that the file /usr/lpp/X11/bin/loadDBE exists.
- Add the line:

```
dbe  /usr/lpp/X11/bin/loadDBE
```

to the /usr/bin/X11/static_ext file.

If the extension only needs to be loaded on demand:

- Check that the file /usr/lpp/X11/bin/loadDBE exists.
- Add the line:

```
dbe /usr/lpp/X11/bin/loadDBE
```

to the /usr/bin/X11/dynamic_ext file.

You can then load the DBE extension by adding the `-x dbe` option to the command line when starting the X Server.

```
$ xinit -- -x dbe
```

If you use the CDE environment:

- Check that the /usr/lpp/X11/bin.loadDBE file exists
- Check that the /etc/dt/config/Xservers file exists, if not
- Modify the following line in the /etc/dt/config/Xservers file:

```
:0   Local local@console /usr/lpp/X11/defaults/xserverrc -T -force :0
```

to:

```
:0   Local local@console /usr/lpp/X11/defaults/xserverrc -x dbe -T -force :0
```

This extension can also be loaded by modifying the `EXTENSIONS` stanza in the /usr/lpp/X11/defaults/xserverrc file by adding the `-x dbe` option.

```
EXTENSIONS="-x dbe"
```

Note the X Server will not start if you have specified the DBE extension should be loaded and the /usr/lpp/X11/bin/loadDBE does not exist or is corrupted.

Also note, the X Server must be restarted if you have made modifications to the above files while the server is running.

Once the X server is loaded, you can use the `xdpyinfo` command to check that the extension was loaded correctly. An API call is also available which checks that the extension is available.

6.1.3.2 Programming concepts

Normally, in an X11 application new windows are created by issuing the core CreateWindow protocol request which allocates a set of window attributes, and, for InpuOutput windows, a buffer into which an image can be drawn. The contents of the buffer are visible whenever the window is displayed.

The DBE extension allows applications to associate a second buffer with the window. This second buffer is commonly referred to as the back buffer, while the first buffer is known as the front buffer. Applications using double buffering do not need to keep track of which physical buffer is currently being displayed since when a buffer is swapped the names associated with the buffers are also swapped. Therefore they need only refer to either the front buffer or back buffer.

Multiple clients and toolkits can all use double buffering on the same window. DBE does not provide a request for querying whether a window has double buffering support, and if so, what the back buffer name is. Given the asynchronous nature of the X Window System, this would cause race conditions. Instead, DBE allows multiple back buffer names to exist for the same window but they all refer to the same physical back buffer. The first time a back buffer name

is allocated for a window, the window becomes double buffered and the buffer name is associated with the window. Subsequently, since the window is already a double-buffered window, nothing about the window changes when a new buffer name is allocated except that the new back buffer name is associated with the window. The window remains double buffered until either the window is destroyed, or until all of the back buffer names for the window are deallocated.

In general, both the front and back buffers are treated the same. In particular, be aware of the following important characteristics:

- Only one buffer per window can be visible at a time (the front buffer).
- Both buffers associated with a window have the same visual type, depth, width, height, and shape as the window.
- Both buffers associated with a window are visible or obscured in the same way. When an Expose event is generated for a window, both buffers should be considered to be damaged in the exposed area. Damage that occurs to either buffer will result in an Expose event on the window. When a double buffered window is exposed, both buffers are tiled with the window background, exactly as stated by the core protocol. Even though the back buffer is not visible, terms such as obscure apply to the back buffer as well as the front buffer.

Some notable features of DBE are:

- DBE adds no new events.
- DBE does not extend the semantics of any existing events with the exception of adding a new DRAWABLE type called BACKBUFFER. If events, replies or errors that contain a DRAWABLE, for example GraphicsExpose, are generated in response to a request, the DRAWABLE returned will be the one specified in the request.
- DBE advertises which visuals support double buffering.
- DBE does not include any timing or synchronization facilities.

These characteristics, as well as the subroutines part of the API are described in detail in Info Explorer.

6.1.4 New X Server Cursor Algorithm

The method the X server uses to service its client requests is by using a round robin loop. The server maintains a queue for each client where it stores the packets for that particular client. It then goes from queue to queue servicing each client in turn. In addition to the client queues the server also maintains another queue where it stores external events such as keystrokes, mouse movements button clicks, and so on. Between execution of client queue packets the X server checks and services this event queue. This algorithm is effective if each client operation takes only a few tenths of a second to be analyzed and executed. Problems however began to occur with the appearance of 3D API′s such as PEX and OpenGL. These API′s allow the developer to first store the the application data into the server, then as a seperate request, ask the server to perform an operation on the data. These operations can be extremely complex and time consuming for large amounts of data. Picture rotation and color computation using heavy shading algorithms can take several seconds or even minutes to complete. The result is that the event queue does not get serviced for this extended period of time and the screen can appear to freeze with no cursor movement. This often causes users to lose patience and attempt to reboot the server believing it to have hung.

A new algorithm for surveying the mouse has been introduced with AIX Version 4.2 whereby the mouse is surveyed at fixed time intervals rather than between completion of client requests.

6.2 CDE Enhancements

The new version of CDE supplied with AIX Version 4.2 contains a number of new features. Specifically, it now includes support for multiple physical displays and introduces two new tools to help users manage their workspaces.

6.2.1 Multiple Screen Support

The X window system is able to manage multiple physical displays as if they were one virtual screen.

The displays can be arranged as a row, a column or even as a matrix. The maximum number of displays is limited by the maximum number of graphics adapters that can be installed in the system. This is because each display must be connected to its own independant graphics adapter. AIX Version 4.2 CDE is now capable of taking advantage of this feature of X windows.

6.2.1.1 Configuring Multiple Display Support

The X server when it starts up will, by default, begin to use every display it finds attached to the system. There is therefore no configuration required to tell the server to use multiple displays. The logical location of the displays however, may not be exactly as desired. Moving the mouse off of the right hand extreme of the display may cause the cursor to appear on the display to the users left or vice versa.

It is always possible of course to physically move the displays into the required order but this is not always convenient. Let's see how we can define the logical location of each display by command or configuration files.

Configuring screen order with the lsdisp and chdisp commands: If multiple displays exist on a machine, the `lsdisp` and `chdisp` commands can be used to list what those displays are and which one of them should be used as the system default display.

The default display will be used as the console for boot messages, as a terminal for ASCII login and for displaying the CDE login window. The X server will also use the default display as the first logical display and will logically locate it as the leftmost display in the group. All the other attached displays are then added to the right of the default display in the order in which they are listed by the `lsdisp` command. The following is an example of the output from the `lsdisp` command:

```
aix42gra> lsdisp


DEV_NAME  SLOT  BUS  ADPT_NAME  DESCRIPTION
========  ====  ===  =========  ===========
ppr0      05    mca  POWER_Gt3  POWER Gt3 Graphics Adapter
gda0      07    mca  colorgda   Color Graphics Display Adapter

   Default display = gda0
```

Figure 19. CDE Enhancements lsdisp Command

This machine has two graphics adapters, one is a POWER Gt3 the other is a Color Graphic Display Adapter which is configured as the default display.

To change the default display the chdisp command is used. It can be changed temporarily with the -d option (for a single session only), in which case the default display will revert to its previous setting after a reboot, or permanently with the -p option. The following shows the ouput of the chdisp -d command:

```
# chdisp -d ppr0
# lsdisp


DEV_NAME  SLOT  BUS  ADPT_NAME  DESCRIPTION
========  ====  ===  =========  ===========
ppr0      05    mca  POWER_Gt3  POWER Gt3 Graphics Adapter
gda0      07    mca  colorgda   Color Graphics Display Adapter

   Default display = gda0
```

Figure 20. CDE Enhancements chdisp -d Command

Note that the setting for the default display has not changed but you are now connected to the other screen. Also that this command must be run as a root user.

```
# chdisp -p ppr0
# lsdisp


DEV_NAME  SLOT  BUS  ADPT_NAME  DESCRIPTION
========  ====  ===  =========  ===========
ppr0      05    mca  POWER_Gt3  POWER Gt3 Graphics Adapter
gda0      07    mca  colorgda   Color Graphics Display Adapter

   Default display = ppr0
```

Figure 21. CDE Enhancements chdisp -p Command

Now the default display has changed. At the next reboot of the X server the first logical display will be the one attached to adapter ppr0.

Configuring screen order using the Xservers file: It is also possible to configure the order of the displays without changing the default display by customizing the /etc/dt/config/Xservers file. If this file does not exist the factory default file, found in the/usr/dt/config directory can be copied to /etc/dt/config before modifying it.

The line which starts the X server is as follows:

```
:0   Local local@console /usr/lpp/X11/defaults/xserverrc -T -force :0
```

To define the logical order of displays this command accepts the -P option with a two digit parameter. The first digit specifies the column number of the display, the second specifies the row number, both relative to the bottom left hand side of the matrix, as shown here:

```
-P[column number][row number]
```

For example lets imagine that we have four graphics adapters and displays attached to our system that we need to arrange in a 2 X 2 matrix.

The lsdisp command lists the names of the devices installed on our system as follows:

```
$lsdisp

DEV_NAME  SLOT  BUS  ADPT_NAME  DESCRIPTION
========  ====  ===  =========  ===========
ppr0      05    mca  POWER_Gt3  POWER Gt3 Graphics Adapter
gda0      06    mca  colorgda   Color Graphics Display Adapter
ppr1      07    mca  POWER_Gt3  POWER Gt3 Graphics Adapter
gda1      08    mca  colorgda   Color Graphics Display Adapter

   Default display = gda0
```

So, by changing the default line:

```
:0   Local local@console /usr/lpp/X11/defaults/xserverrc -T -force :0
```

to:

```
:0   Local local@console /usr/lpp/X11/defaults/xserverrc -P11 ppr0
-P12 gda0 \ -P21 ppr1 -P22 gda1 -T -force :0
```

we get:

- ppr0 bottom left
- gda0 bottom right
- ppr1 top left
- gda1 top right

We must then restart the dtlogin daemon in order to have these changes take effect. There are two ways to do this:

1. /usr/dt/bin/dtconfig -reset

   or

2. kill -1 <dtloginpid> where dtloginpid stands for the Process id of the main dtlogin daemon.

Note that when CDE is started with multiple displays, only the one where the login window appears has a black root window. All the others keep the default gray root window.

In order to change this we can modify the Xsetup file. As with the Xservers file a default version can be found in /usr/dt/config and copied to /etc/dt/config if required.

If we add one call to xsetroot in the following location for each display, the root windows will be changed as desired.

```
if [ "$DTXSERVERLOCATION" != "remote" -a -z "$XSTATION" ]; then
#
# Since X server is local, optimize by checking local desktop
# font directories and making one call to xset.
#

xsetroot -solid black -display :0.1
xsetroot -solid black -display :0.2
xsetroot -solid black -display :0.3
```

6.2.1.2 Working with Multiple Displays

Once logged into CDE a front panel is displayed on each screen. These are independant in that the number of workspaces does not have to be the same on each display; each menu and submenu can be configured seperately and we can choose whether to have the menu displayed or not.

Synchronization: The workspace manager on each front panel can be configured from the style manager menu to select whether each screen should be independant or synchronized. That is, if we click on the button to go to workspace one of screen one, should all screens switch to the first workspace or keep the current workspace.

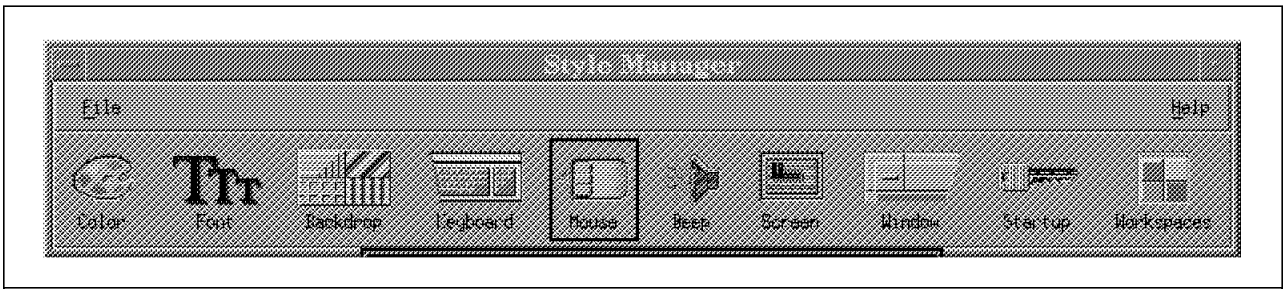


Figure 22. CDE Style Manager

Screen lock and saver: The screen lock and saver timeout period can be changed from any display but the value selected applies to all displays. It is not possible to lock individual displays.

Color palettes and backdrops: These can be different for each screen. One screen may use a monochrome palette while another is using full color. Similarly, the backdrop for each workspace can be different. Workspace one of display two does not have to be the same as workspace one of display one.

6.2.2 Graphical Workspace Manager

The Graphical Workspace Manager is a new built-in function of the DeskTop Window Manager supplied with AIX Version 4.2. Its role is to provide a simple representation of the various applications started in any workspaces. One of the drawbacks of using multiple workspaces is that it is sometimes difficult to remember in which particular workspace an application was started. It is then tedious to have to search through each workspace to find the application you want. The Graphical Workspace Manager is supplied as an aid to users in locating and changing context to the desired application.

6.2.2.1 What Does the Graphical Workspace Manager Do?

The Graphical Window Manager appears as a window with as many subwindows as there are workspaces defined on your screen. If you have an odd number of workspaces, the last subwindow is left black. In each subwindow, there are rectangles that represent the windows of each application located in that workspace. The name of the application, or part of it, is displayed in the rectangle.

By moving the mouse over the subwindows of the workspace manager the name of each workspace appears in the Title Bar. When the mouse is moved over the rectangles within the workspace the name of the application represented by that rectangle is displayed.

Using the left button of the mouse, it is possible to drag applications (rectangles) from one subwindow to another which automatically moves the application to the chosen workspace.

Clicking on a rectangle with the middle-button of the mouse, makes the workspace of the application selected the current one and brings the window of that application to the top of the screen.

By clicking with the middle-button in a free zone (not occupied by rectangles), then you switch the screen to the desired workspace.

Note that the icons are also represented in the Graphical Window Manager, but this method cannot be used to move them. The icon box, if used, can be moved.

Be aware that if there are several physical screens there is only one Graphical Workspace Manager on each screen.

Also note, if the Graphical Window Manager is used to switch context to a specific application in a multithreaded environment, only the workspace of the current screen is changed, other screens are left unchanged, even if you have configured the screens to be synchronized.

The Graphical Window Manager is not represented as an application and has no rectangle associated with it.

6.2.2.2 Starting the Graphical Workspace Manager

The Graphical Workspace Manager is a built-in function of the Desktop Window Manager and not an independent program, it is, therefore, very fast to launch. There are two ways to bring up the Graphical Workspace manager:

- From the frontpanel

A new button has been added to the frontpanel, just to the left of the workspace buttons and below the screenlock button.

- From the mouse menu

  The default mouse menu, that appears when clicking on the right button in the root window contains a call to the Graphical Window Manager.

  If you wish to customize your own menu, you must make a call to f.show_gwm in your .dtwmrc file.

It is also possible to configure dtwn so that the workspace buttons are hidden and the only way to change workspaces is via the Graphical Workspace Manager.

6.2.3 Active Applications List Window

The Active Applications List is also a built-in function of the Destktop Window Manager. Its role is almost identical to the Graphical Window Manager but it does not provide as much funtionality as the latter. It is simply a list of all applications running in each workspace and by selecting an application from the list and clicking on it the context is switched to the workspace containing that application and the application is brought to the top.

Like the Graphical Workspace Manager, the Active Applications List only changes the screen on which you are currently working even if synchronization between displays has been configured.

6.2.3.1 Starting the Active Applications List Window

The Active Applications List is also a built-in function like the Graphical Workspace Manager. It can only be started from the dtwm menu that is displayed when pressing the right mouse button while in the root window. Customization is done via the f.show_app_list function in the .dtwmrc file.

Chapter 7. Communications

AIX Version 4.2 introduces a number of changes to the communication subsystem. These changes affect areas including:

- Communication device drivers
- AIXLink/X.25 Version 1.1.3
- Asynchronous Communication subsystem
- LDTERM - POSIX Line Discipline Module

7.1 Communication Device Drivers

Communication device drivers are software components of the communication I/O subsystem and their primary function is to allow access to hardware devices.

In this release of AIX some device drivers that were previously part of separate LPP's have been ported onto AIX Version 4.2 and are now included as base device drivers.

7.1.1 NTA (Network Terminal Accelerator) Device Driver

This device driver in earlier releases was a seperate LPP. It is now included in AIX 4.2 as a base device driver. Additionally, the SMIT panels have been updated to make configuration of the driver simpler.

The Network Terminal Accelerator adapter is particularly suited to situations where there is a need to reduce the amount of CPU resources consumed by network processing of telnet and rlogin sessions.

7.2 X.25 on Artic960

The X.25 LPP now includes support for the Artic960 adapter. Also, the Artic Portmaster implementation of AIXLink X.25 will be ported to the Artic960 family of adapters with the V.24 daughter board.

The Artic Portmaster is an MCA card and is very appropriate for high end racks and SP2 machines. This release of X.25 will only support the V.24 daughter board for the Artic960 adapter. This feature is an asynchronous RS232C interface with supported speeds up to 20 Kbps but, with short cables, it is possible to operate at up to 115.2 Kbps.

7.3 X.25 on ISA Machines

X.25 on ISA machines has the capability of supporting X.25 on four ports at speeds of up to 64 Kbps. This is significant, especially for servers. It allows these systems to handle more traffic than a single port card and improves their connectivity.

© Copyright IBM Corp. 1996

Important

You should be aware that on ISA machines, X.25 configuration requires ISA interrupt levels, bus I/O address and adapter memory address to be entered. If there are additional communications adapters on the bus a conflict can occur. For this reason you should pay particular attention to the interrupts and addresses being used by the different adapters.

Figure 23 shows the SMIT panel for configuring X.25 on a PCI/ISA machine.

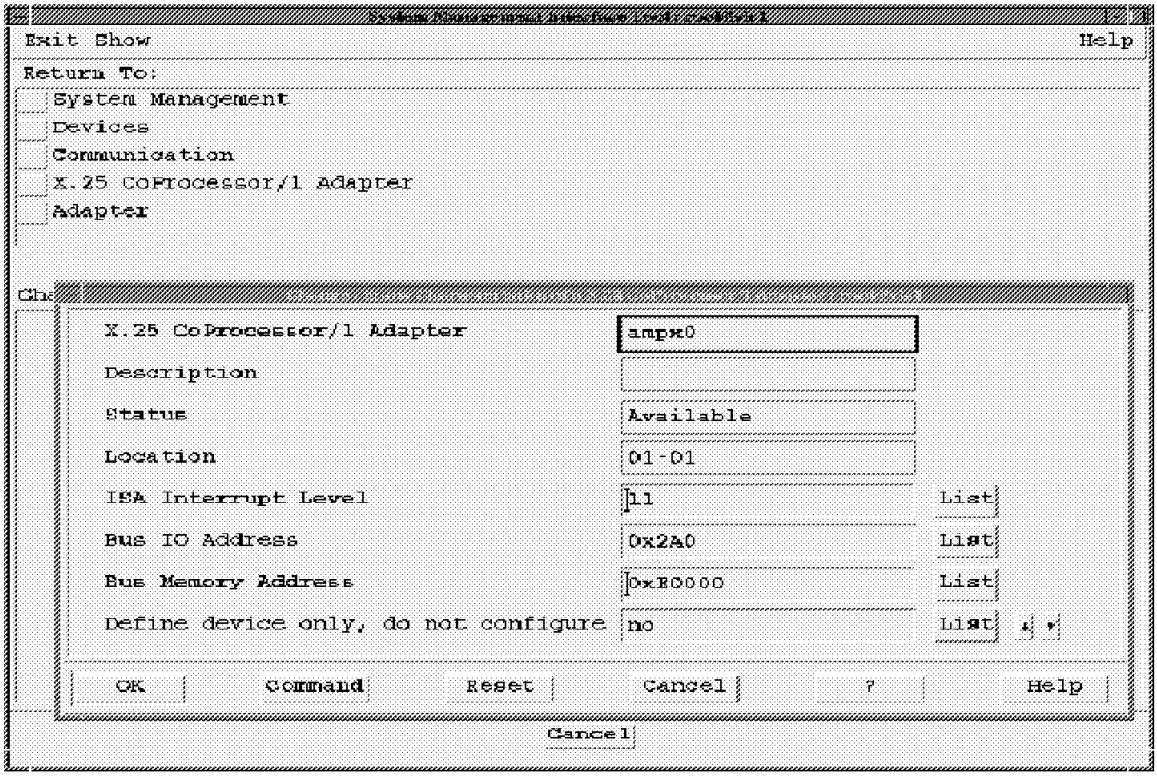


Figure 23. X.25 Adapter on a PCI/ISA machine SMIT Panel

7.4 SNMP Support on the ATM Adapter

Simple Network Management Protocol support has been added to the MCA Turboways 100 Asynchronous Transfer Mode Adapter. This feature allows the adapter to be remotely managed by a Simple Network Management Protocol (SNMP) application such as IBM's NetView/6000.

7.5 ATM 3rd Party CDLI Interface

This function allows 3rd party developers to write applications that interface with ATM directly via CDLI (Common Data Link Interface) without the overhead of intermediate protocol support. It is especially useful for multimedia applications, in particular those that wish to take advantage of ATM's unique synchronization capability for audio and video.

7.6 Full Duplex Software Support for Ethernet PCI Adapter

The PCI Ethernet adapter hardware has always been capable of full duplex operation but the device driver lacked this capability. With this enhancement to the device driver we are now capable of effectively doubling the available LAN bandwidth and have bidirectional frame transmission on 10Base-T network environments.

This performance enhancement is made possible by the separate transmit and receive channels of twisted pair and/or fiber technologies.

Note: To take advantage of this feature the other devices on the 10Base-T network such as hubs, routers and switches must also support full duplex operation.

Figure 24 shows the enablement of the full duplex option on an PCI/ISA Ethernet board.

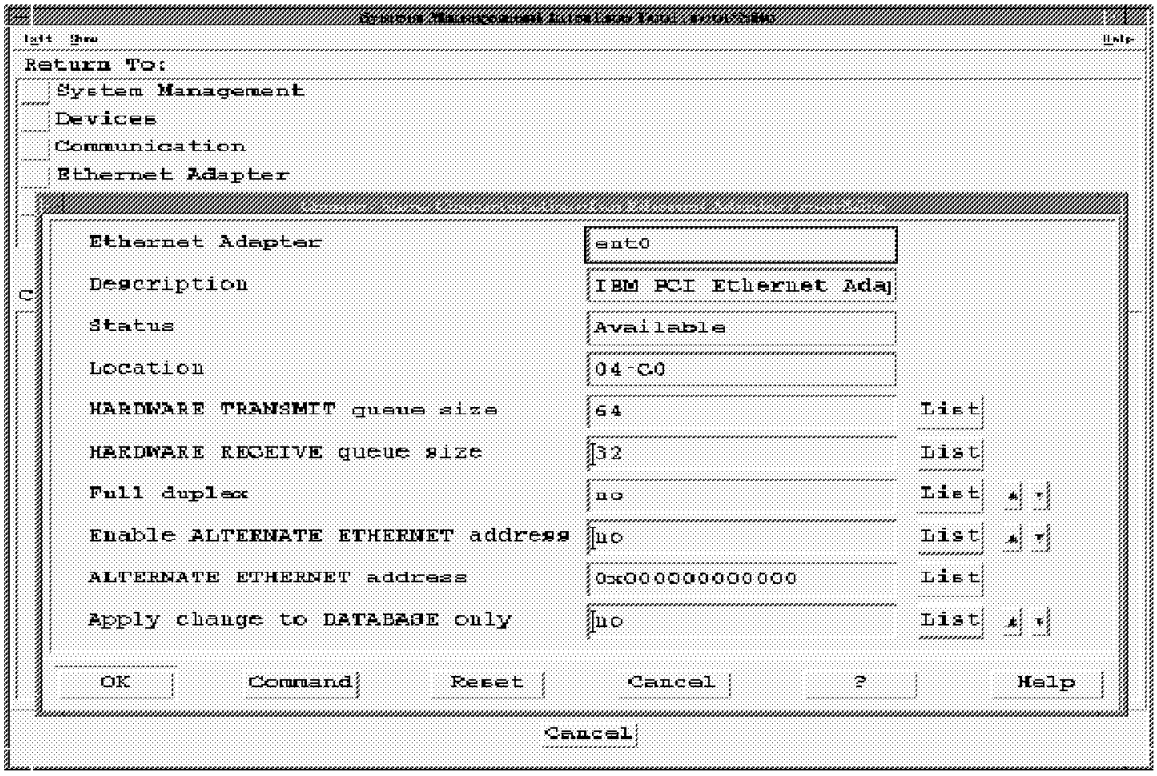


Figure 24. PCI/ISA Machine SMIT Panel for Full Duplex Option

7.7 7318 (Network Terminal Accelerator) Support on SMP Machines

Support for the 7318 models P10 and S20 will be packaged with the base AIX Version 4.2 for both uni-processor and SMP-models.

On the SMP (Symmetric Multi-Processor) machines support for the 7318 is an enhancement which significantly improves the connectivity options for these high-end servers.

Note: This feature is not supported on SMP systems that require AIX Version 4.1.

7.8 AIXLink/X.25 Version 1.1.3

The AIXLink/X.25 Version 1.1.3 Licensed Program Product (5696-926) now supports the high speed Artic960 family of adapters from Boca (FC 2929, 2935 and 2938) up to speeds of 2 Mbps per port.

Feature Code Description for Artic960 Adapters:

- Feature Code 2929 8 ports, EIA 232, 4 MB memory
- Feature Code 2935 6 ports, V.35, 4 MB memory
- Feature Code 2938 8 ports, X.21, 4 MB memory

AIXLink/X.25 can support two adapter types:

ARTIC Portmaster Adapter/A

- 8 ports (V.24 interface) or 6 ports (V.35 or X.21 interface)
- Up to 64 Kbps full duplex concurrently for each port
- Up to 512 virtual circuits per port with a maximum of 1024 per adapter

X.25 Interface Co-Processor

- Single-port support with V.24, V.35, or X.21 interface; either Micro Channel or ISA interface
- Supports up to 512 virtual circuits per port

This capability is included in all uni-processor and SMP-models on AIX V4.1.x and AIX V4.2.

7.9 Asynchronous Communication Subsystem

The native serial ports on the RISC System/6000 in combination with AIX Version 4.2 now have the capability of supporting a wide range of asynchronous speeds. The actual speeds supported will vary according to the RISC System/6000 being used.

This is possible because the native ports for ISA bus machine uses the *rsdd_rspc* driver in /etc/drivers/isa (capable of speeds up to 115.2 Kbps) and the MCA bus native port, Micro Channel 8-port and 16-port adapters use the *rsdd* driver in /etc/drivers which have *High-baud/Odd-baud* capability.

All these serial ports use common drivers, but they are implemented with different hardware, for example: *UART's (Universal Asynchronous Receiver/Transmitter) or Multifunction I/O* chips. These chips are clocked at different speeds determined by the crystal frequency of the local oscillator which may be an integral part of the chip itself or a separate circuit. This crystal frequency determines the range of baud rates a particular port implementation will be able to cover. In addition to defining the absolute range of possible baud rates, different crystal frequencies will be better suited/worse suited to hitting discrete baud rates within that range.

The higher frequency supplied by the oscillator is fed to the I/O chip and typically used for clocking of on-chip circuitry. It is also used to provide the

clock which clocks data in and out on the receive and transmit data lines, but because of its higher frequency it must be divided down to a lower frequency. The amount by which it is divided is programmable and is called the *baud rate divisor*. The device driver writes the divisor to an on-chip register and the chip uses the value to divide the clock down to the required frequency.

In previous versions of AIX the divisors were fixed values pre-selected to match the baud rates that the user was allowed to select via SMIT or the stty command. The legal baud rates were in fixed increments, that is, 2400, 4800, 9600 and so on. In AIX Version 4.2 this limitation has been removed and the user is allowed to select any baud rate they desire between the minimum and maximum rates supported by the particular chip.

When the user selects a particular baud rate the device driver calculates the required *baud rate divisor*. If the incoming clock frequency is not evenly divisible by the calculated divisor and leaves a remainder the driver then calculates the percentage error that would result from using this divisor. If the error would be greater than 2%, then the requested baud rate is rejected and the user must select another.

The following C program makes the same kind of baud rate calculations that are made in the driver. You must supply the crystal frequency. Predefined crystal frequency values can be found in the ODM database of your system, by simply running the following commands:

```
odmget -q "attribute='frequency' and uniquetype like 'adapter/sio*'"
PdAt

odmget -q "attribute='frequency' and uniquetype like
'adapter/isa_sio*'" PdAt

odmget -q "attribute='frequency' and uniquetype like 'adapter/mca*'"
PdAt

/*
 * divisor -
 *
 * Calculates achievable baud rates (and associated baud rate divisors)
 * for National Semiconductor UARTS.  If invoked with no options, tables
 * for the popular baud rates will be generated for xtal frequencies of:
 *
 * 12.288  MHz 8- and 16-port adapters                          (NSC 16552)
 *  8      MHz sio:   adapter/sio/s1a,s2a          id = 0x5fdf  (NSC 16550)
 * 24      MHz sio_1: adapter/sio/s1a_1,s2a_1      id = 0xe6de  (NSC 16553)
 *  8      MHz sio_2: adapter/sio/s1a,s2a          id = 0xfef6  (VLSI 16552)
 *  1.8432 MHz sio_3: adapter/sio/s1a_3,s2a_3,..... id = 0xd9fe  (NSC PC87312)
 *  8      MHz sio_3: adapter/sio/...........,s3a_3 id = 0xd9fe  (NSC 16550)
 *
 * If invoked with a '-x' and specific xtal freq, divisors for the popular
 * baud rates will be calculated.
 *
 * If invoked with a '-b' and specific baud, divisors will be calculated
 * for the aforementioned xtal frequencies.
 *
 * If invoked with a '-x' and a '-b' and specific baud, the baud rate
 * divisor for that xtal frequency and baud rate will be calculated.
 *
 * A '-r' in addition to any of the aforementioned invocations will cause
```

```
 * additional calculations to be performed using the integer based algorithm
 * used in the 'rsdd' driver.
 */

#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
#include <string.h>

char *progname;
int rflag, bflag;
double baud;

double rint(double x);

double bauds[] = {
  50, 75, 110, 134.5, 150, 200, 300, 600, 1200, 1800, 2000, 2400,
  3600, 4800, 7200, 9600, 19200, 38400, 56000, 128000,
  0
};

void usage()
{
  fprintf(stderr, "%s [-r] [-x crystal_frequency_Hz] [-b desired baud]]\n\n",
          progname);
  fprintf(stderr, "where '-r' also shows rsdd-like integral calculations\n\n");
  exit(1);
}

void do_rs(int freq, int baud)
{
  int new, diff, use, actual;

  new = baud * 2;
  if (baud == 134)
    ++new;                    /* new == 134.5 * 2 */
  use = (((2 * freq) / new) + 8) / 16;
  actual = freq / (use * 16);
  if ((diff = new - (2 * (freq / (use * 16)))) < 0)
    diff = - diff;
  printf("desired: %6d     actual: %6d     divisor: %5d error: %3d%%     <- rsdd\n",
         baud, actual, use, ((diff * 100) / new));
}

void do_baud(double freq, double baud)
{
  int use;
  double actual, nuts, slop;

  nuts = freq/(baud * 16);
  use = rint(nuts);
  actual = (freq / (use * 16));
  if ((slop = 1 - (actual / baud)) < 0)
     slop = - slop;

  printf("desired: %10.3lf actual: %10.3lf divisor: %5d error: %7.3lf%%%s\n",
         baud, actual, use, slop * 100, (rflag ? " <- float" : ""));
}
```

```
void do_it (double freq)
{
  int i;

  printf("\nNational Chip at %.4lf MHz\n", freq/1000000);
  if (bflag) {
    if (rflag)
      do_rs(rint(freq), rint(baud));
    do_baud(freq, baud);
  } else {
    for (i = 0; bauds[i]; i++) {
      if (rflag)
 do_rs(rint(freq), rint(bauds[i]));
      do_baud(freq, bauds[i]);
    }
  }
}

main(int argc, char *argv[])
{
  int c, xflag;
  double freq;

  if (progname = strrchr(argv[0], '/'))
    ++progname;
  else
    progname = argv[0];

  xflag = bflag = 0;
  while ((c = getopt(argc, argv, "rx:b:")) != EOF) {
    switch(c) {
    case 'x': xflag++;
                if ((sscanf(optarg, "%lf", &freq)) != 1) {
                   fprintf(stderr, "xtal frequency \"%s\" not valid\n", optarg);
                     usage();
                }
  break;
    case 'b': bflag++;
                if ((sscanf(optarg, "%lf", &baud)) != 1) {
                   fprintf(stderr, "baud rate\"%s\" not valid\n", optarg);
                     usage();
                }
  break;
    case 'r': rflag++;
  break;
    default: usage();
    }
  }

  if (xflag) {
    if (bflag) {
      if (rflag)
 do_rs(rint(freq), rint(baud));
      do_baud(freq, baud);
    } else {
      do_it(freq);
    }
  } else {
    do_it(1843200);
```

```
    do_it(8000000);

    do_it(12288000);

    do_it(24000000);
  }
}
```

Figure 25 shows the output generated by the C program above when compiled and run with the following flags and values: *divisor -b 38400*

```
National Chip at 1.8432 MHz
desired:  38400.000 actual:  38400.000 divisor:     3 error:   0.000%

National Chip at 8.0000 MHz
desired:  38400.000 actual:  38461.538 divisor:    13 error:   0.160%

National Chip at 12.2880 MHz
desired:  38400.000 actual:  38400.000 divisor:    20 error:   0.000%

National Chip at 24.0000 MHz
desired:  38400.000 actual:  38461.538 divisor:    39 error:   0.160%
```

Figure 25. Sample Output from divisor -b 38400

Figure 26 shows the output generated by the C program above when run with the following flags and values: *divisor -b 57600*

```
National Chip at 1.8432 MHz
desired:  57600.000 actual:  57600.000 divisor:     2 error:   0.000%

National Chip at 8.0000 MHz
desired:  57600.000 actual:  55555.556 divisor:     9 error:   3.549%

National Chip at 12.2880 MHz
desired:  57600.000 actual:  59076.923 divisor:    13 error:   2.564%

National Chip at 24.0000 MHz
desired:  57600.000 actual:  57692.308 divisor:    26 error:   0.160%
```

Figure 26. Sample Output from divisor -b 57600

Figure 27 on page 147 shows the output generated by the C program above when run with the following flags and values: *divisor -b 115200*

```
National Chip at 1.8432 MHz
desired: 115200.000 actual: 115200.000 divisor:     1 error:   0.000%

National Chip at 8.0000 MHz
desired: 115200.000 actual: 125000.000 divisor:     4 error:   8.507%

National Chip at 12.2880 MHz
desired: 115200.000 actual: 109714.286 divisor:     7 error:   4.762%

National Chip at 24.0000 MHz
desired: 115200.000 actual: 115384.615 divisor:    13 error:   0.160%
```

Figure 27. Sample Output from divisor -b 115200

Be aware that just because a particular UART crystal combination can hit a particular baud rate, the system may not be powerful enough to handle ports at that speed.

Important

You must always consider the system overhead generated by other hardware and software. This will influence how many CPU cycles are available to service the serial ports, which could cause a data integrity problem on serial port traffic when specifying baud rates that are too high for the system to handle. In this type of situation even with flow control active, the system might not be able to service the serial ports before the UART′s receive buffers are overrun by incoming data. Even if you don′t get overrun errors logged in the system′s errorlog, the system could spend much of its time servicing serial port interrupts causing other applications to run very poorly.

Table 3 shows the asynchronous supported speeds on AIX V4.2:

Table 3. Summary of Asynchronous Speeds Supported on AIX Version 4.2

Type of Port	Architecture	Max Baud Rates	Feature Code
Standard Serial Port	System Planar	38.4	n/a
8-Port EIA-232C	Micro Channel, ISA	38.4, 115.2	2930,2931
8-Port EIA-422A	Micro Channel	38.4	2940
8-Port MIL-STD 188	Micro Channel	38.4	2950
8-Port EIA-232C	ISA	115.2	2931
8-Port EIA-232C/EIA-422A	ISA	115.2	2932
16-Port EIA-232C	Micro Channel	38.4	2955
16-Port EIA-422A	Micro Channel	38.4	2957
16-Port RAN EIA-232C	Micro Channel	38.4	8130,8140
128-Port Controller	Micro Channel,ISA	57.6	8128,2933
7318-P10	Network Attachment	115.2	7318-P10
7318-S20	Network Attachment	115.2	7318-S20

7.9.1 RS232/RS422 Support on ISA-bus Systems

The 8-port ISA adapter device driver has been upgraded to support programming of the asynchronous I/O ports for both RS232 and RS422 operations. All ports are independently programmable to operate in either mode.

SMIT pannels and ODM data base entries have also been upgraded to support RS422 on this adapter. The device driver configuration method has been modified to support RS422 configuration.

Note: EIA-422 is required for configurations where asynchronous devices are located at distances beyond the EIA-232 limit of 200 ft., EIA-422 supports distances up to 4000ft.

7.9.2 LDTERM - POSIX Line Discipline Module

The major enhancement for ldterm is that there is now a single copy of input data, and that data remains in ldterm until read by an application. When the Stream head is entered on a read() system call, the Stream head builds an *M_READ* message block and places it in the write queue of the next module in the stream. ldterm collects characters as they arrive from downstream and when it receives an *M_READ* message block, it processes those characters according to the attributes in force at the time. ldterm determines when the read() should be completed, and sends the characters to satisfy that read() upstream in an *M_DATA* message. The purpose behind this design change is to match standards requirements, either explicitly stated or historically implied, as closely as possible without regard to the performance impact and to improve the support for third party developers writing tty Streams modules.

Throughput performance has been enhanced especially in small read()′s and write()′s in raw I/O mode, this is very common in full-screen character based applications. The changes in the ldterm redesign directly addresses this performance issue and also solves some functional deficiencies.

7.9.3 wantio Concept

Another way to improve performance in cases where there is a large amount of incoming data (for example, file transfer applications), is by utilizing a utility called *wantio*. This is used by a STREAMS module or driver to register I/O (read()/write()/select()) entry points with the Stream head. The Stream head then calls these entry points directly, by-passing all normal STREAMS module processing, when an I/O request is detected. This utility is very useful in situations where normal module processing is not required or where STREAMS processing is to be performed outside of AIX.

STREAMS modules and drivers should precede a wantio() call by sending a high priority M_LETSPLAY message upstream. The M_LETSPLAY message format is a message block containing an integer followed by a pointer to the write queue of the module or driver originating the M_LETSPLAY message. The integer counts the number of modules which can permit direct I/O. Each module passes this message to its neighbor after incrementing the count if direct I/O is possible. When this message reaches the Stream head, the Stream head compares the count field with the number of modules and drivers in the Stream. If the count is not equal to the number of modules it indicates that at least one of the modules is not capable of *wantio* processing. If this is the case, then a M_DONTPLAY message is sent downstream indicating direct I/O will not be permitted on the Stream. If the count is equal then queued messages are cleared by sending

them downstream as M_BACKWASH messages. When all messages are cleared then an M_BACKDONE message is sent downstream. This process starts at the Stream head and is repeated in every module in the Stream. Modules will wait to receive an M_BACKDONE message from upstream. Upon receipt of this message, the module will send all queued data downstream as M_BACKWASH messages. When all data is cleared the module will send an M_BACKDONE message to its downstream neighbor indicating that all data has been cleared from the Stream to this point.

wantio registration is cleared from a Stream by issuing a wantio() call with a NULL pointer to the *wantio* structure.

Note: Multiprocessor serialization is the responsibility of the driver or module requesting direct I/O. The Stream head acquires no STREAMS locks before calling the *wantio* entry point.

Currently, the write entry point of the *wantio* structure is ignored.

Syntax:

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

int wantio(queue_t *q, struct wantio *w)
```

Parameters

q Pointer to the queue structure

w Pointer to the wantio structure

Return Values

Returns 0 always.

Note: The use of this programming utility is currently only supported on 8 and 128 port ISA Asynchronous Controler Adapters, and on the 128 port MCA Asynchronous Controler Adapter.

The following explains the meaning of some of the terms used above:

STREAMS is a kernel mechanism that supports the development of network services and data communication drivers. It defines interface standards for character input and output within the kernel, and between the kernel and the user level.

Stream head is the end of the Stream closest to the user process. The principal functions of the Stream head are processing of STREAMS-related system calls and bidirectional transfer of data and information between a user process and messages in the STREAMS kernel space.

7.9.4 Enhanced Terminal Subsystem Debugging and Tracing Capabilities

In addition to the enhancements covered in the previous sections, the AIX Version 4.2 tty subsystem contains improvements to the tracing and debugging facilities. It effectively brings these facilities up to a similar level to the one provided in the pre-streams AIX Version 3 subsystem.

Chapter 8. TCP/IP Enhancements in AIX V4.2

Enhancements to TCP/IP in AIX Version 4.2 include extensions to Point to Point Protocol to provide authentication support. Sendmail has been updated to Version 8.7 and NTP Version 3 is now included. In addition, the Streams environment now has more tunable parameters and `ifconfig` provides the ability to turn off TCP/IP checksumming.

8.1 Point-to-Point Protocol (PPP)

The implementation of PPP in AIX V4.2 has been upgraded to provide authentication support, a feature not included in previous releases.

8.1.1 Point-to-Point Protocol Authentication

Support for the following authentication protocols as described in RFC1334 has been added.

- Password Authentication Protocol (PAP)
- Challenge-Handshake Authentication Protocol (CHAP)

8.1.1.1 Changes to bos.net.ppp in AIX V4.2

The following changes were made to bos.net.ppp and to SMIT to support PPP authentication:

- /usr/sbin/pppauthd added to bos.net.ppp
- New menu options added to SMIT PPP panel
- Authentication on/off option added to SMIT Link Control Configuration panels
- PAP and CHAP parameters added to SMIT Link Control Configuration panels

Along with the above changes, the following additional flags have been added to the `/etc/ppp/mkppp` command for configuring PAP and CHAP as follows:

-show type_file	Retrieve information for SMIT display (type_file = lcp or ip, pap or chap)
-pap	Indicates PAP authentication (used with -add/-delete)
-chap	Indicates CHAP authentication (used with -add/-delete)
-force	Force on authentication method
-interval number_seconds	Number of seconds for CHAP interval challenge
-user name	User name for authentication
-pw password	Password for authentication
-rh remotehostname	Remote hostname for authentication

The following flag has been replaced in the `/etc/ppp/mkppp` command:

-show conf_file

© Copyright IBM Corp. 1996

8.1.1.2 Changes to SMIT Panels

The main SMIT panel for configuration and control of PPP has been changed to add options for PAP and CHAP authentication as shown in Figure 28.

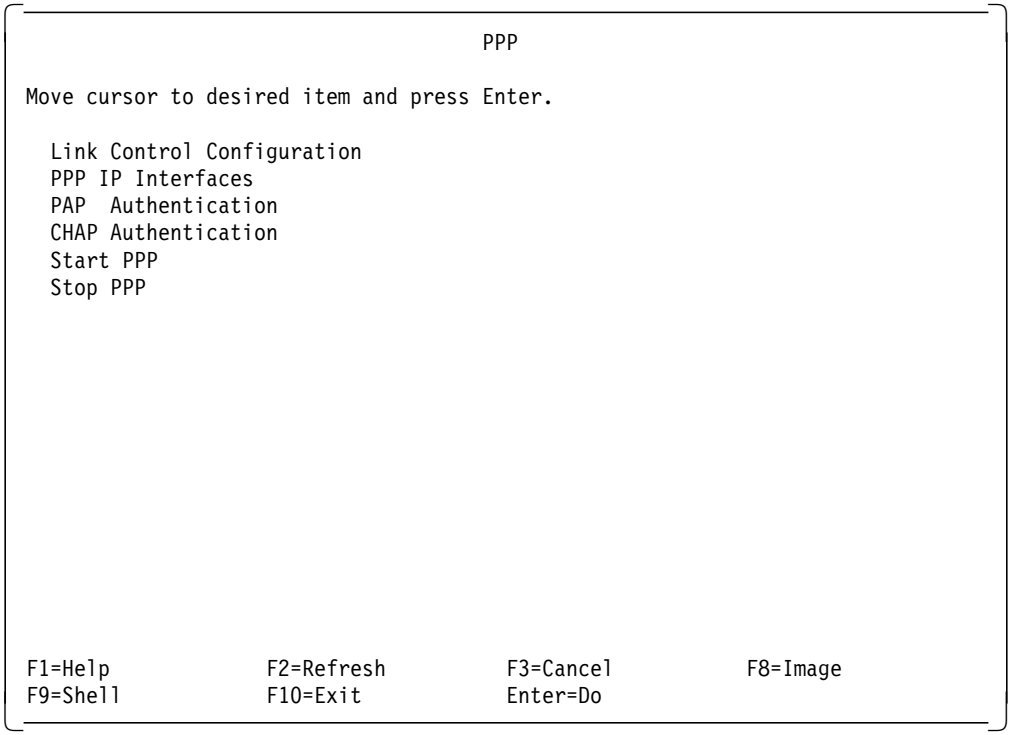


Figure 28. SMIT PPP Panel

The fastpath for this panel is `smit ppp`.

The Link Control Configuration menu has not changed as a result of the SMIT panel changes although the selection panels for Add a Link Configuration and Change/Show a Link Configuration have been changed to provide additional fields for PAP and CHAP authentication. These changes are shown in Figure 29 on page 153 and Figure 30 on page 153.

```
                                LINK Configuration

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

                                                      [Entry Fields]
  PPP subsystem name                                 []
  max server connections                             []                          #
  max client connections                             []                          #
  max demand connections                             []                          #
  max ip interfaces                                  []                          #
  max async hdlc attachments                         []                          #
  mru                                                []                          #
  async character map                                []                          X
  negotiate MRU                                       yes                        +
  negotiate async map                                 yes                        +
  negotiate protocol compression                      yes                        +
  negotiate address control compression               yes                        +
  force authentication                                no                         +
  chap interval                                      []                           #


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

Figure 29. SMIT PPP, Add a Link Configuration Panel

The fastpath for this panel is `smit addlcp`.

```
                                LINK Configuration

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

                                                      [Entry Fields]
  PPP subsystem name                                 [sito]
  max server connections                             [1]                         #
  max client connections                             [1]                         #
  max demand connections                             [1]                         #
  max ip interfaces                                  [3]                         #
  max async hdlc attachments                         [3]                         #
  mru                                                []                          #
  async character map                                []                          X
  negotiate MRU                                       yes                        +
  negotiate async map                                 yes                        +
  negotiate protocol compression                      yes                        +
  negotiate address control compression               yes                        +
  force authentication                                yes                        +
  chap interval                                      [500]                        #


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

Figure 30. SMIT PPP, Change/Show a Link Configuration Panel

The fastpath for this panel is `smit chglcp`.

8.1.2 Password Authentication Protocol

Similar to the standard login procedure, PAP uses an ID/password pair which is sent repeatedly to the Authenticator until they are either authenticated or the connection is terminated.

The peer controls the frequency and timing of the authentication attempts, and the ID/password pair is sent *in the clear*. This makes PAP vulnerable to playback or repeated *trial-and-error* authentication attempts. Also, PAP authentication occurs only once, during initial establishment of the link. Use of stronger authentication methods such as CHAP, if available, must be negotiated prior to PAP.

8.1.2.1 Configuring PAP Using SMIT

By entering the fastpath `smit ppppap`, the panel in Figure 31 is displayed.

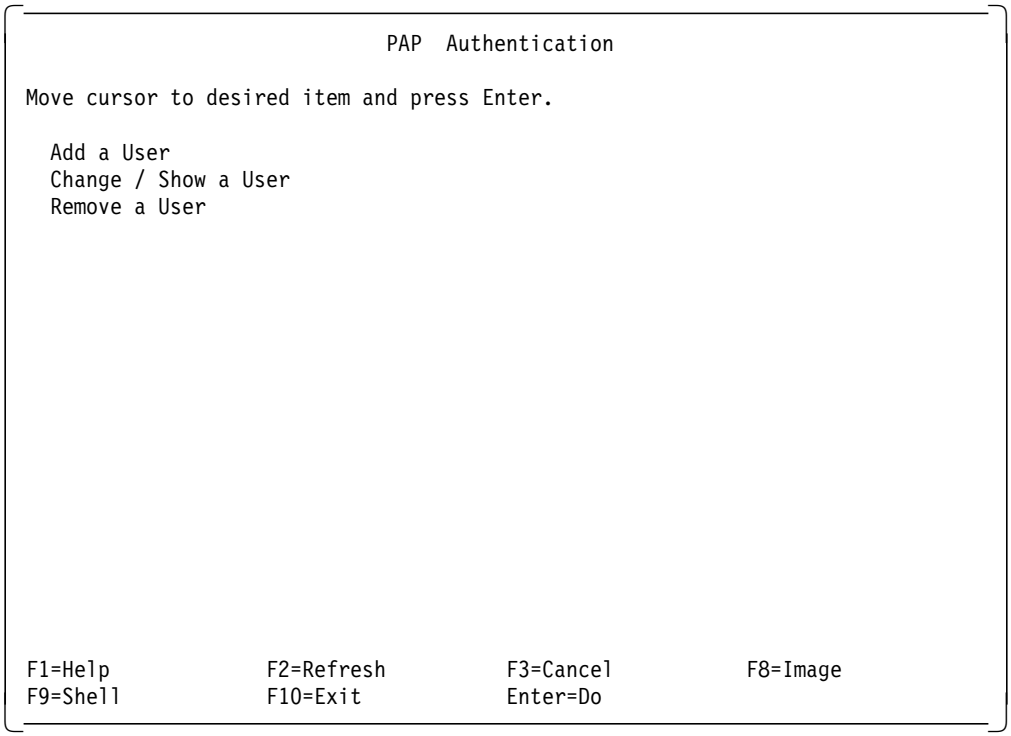


Figure 31. SMIT PAP Authentication Panel

Add a User: The fastpath `smit addpapuser` will display the Add PAP User panel as shown in Figure 32 on page 155.

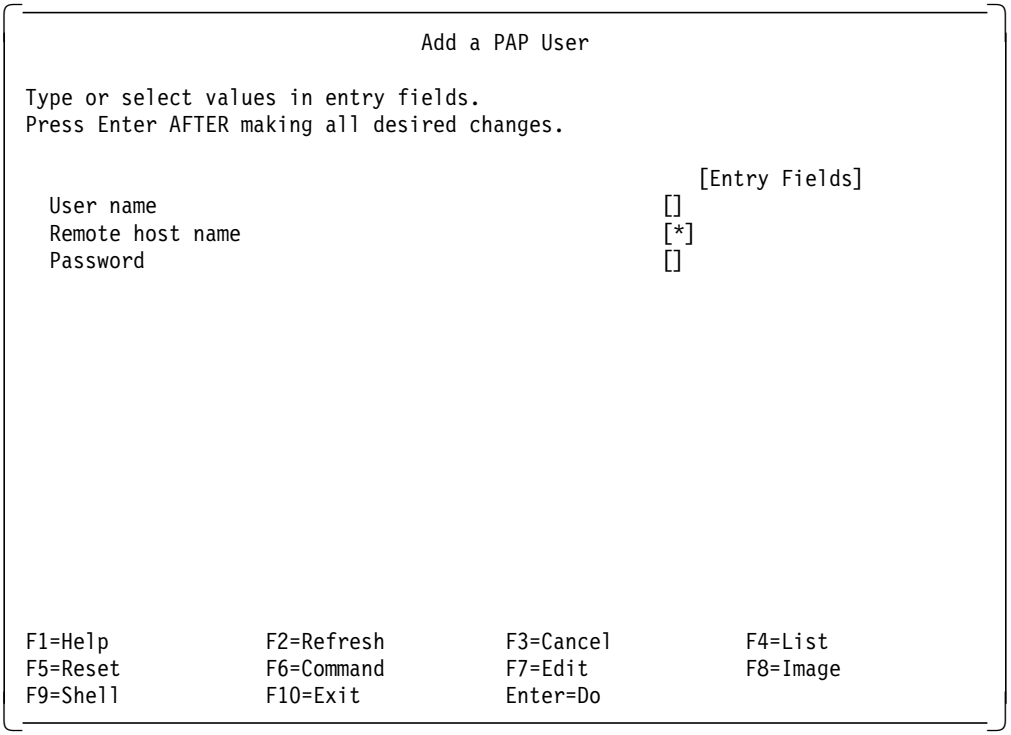


Figure 32. SMIT Add a PAP User Panel

Select **Add a User** to create a /etc/ppp/pap-secrets file if it does not already exist or to add to an existing /etc/ppp/pap-secrets file. The /etc/ppp/pap-secrets file is used to authenticate a peer or to provide the values to be sent to a server for authentication.

User name can be either an alphanumeric string identifying a user, or it may contain the asterisk (*) value to match any user. The asterisk (*) may only be specified for peer authentication.

Remote host name is the name of the remote system that is recognized by the PPP subsystem. This is not necessarily the host name that is recognized by TCP/IP. This is an optional field for PAP authentication, with the default value being asterisk (*) to match any remote host.

The User name/Remote host name pair must be unique within the /etc/ppp/pap-secrets file.

Password is an alphanumeric string. If the password begins with an @, the text that follows must be the file name containing the password.

Change/Show a User: The fastpath `smit listpapuser` will display the Change/Show a PAP User panels as shown in Figure 33 and Figure 34.

The first panel displays a list of existing PAP users whose details can be listed or updated.

```
                              PAP User List

   Move cursor to desired item and press Enter.

     user tdm        remote host *   password password
     user root       remote host sito        password @/etc/ppp/passwd.root
     user *  remote host *   password hello

   F1=Help                 F2=Refresh              F3=Cancel
   F8=Image                F10=Exit                Enter=Do
   /=Find                  n=Find Next
```

Figure 33. SMIT PAP User List Panel

Use the arrow keys to highlight the user whose details you wish to change, and press the **Enter key**. The Change/Show panel will be displayed. Any information associated with this entry can then be updated. If the entry is changed, the original entry is deleted, and a new entry is appended to the /etc/ppp/pap-secrets file.

```
                          Change / Show a PAP User

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

                                                      [Entry Fields]
  User name                                          [tdm]
  Remote host name                                   [*]
  Password                                           [password]









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

Figure 34. SMIT Change/Show a PAP User Panel

Remove a User: The fastpath `smit rmpapuser` displays the same list panel as shown in Figure 33 on page 156 to display a list of existing PAP users who can be removed. Use the arrow keys to highlight the user whose details you wish to remove, and press the **Enter** key. The Remove a PAP User panel is similar to the panel in Figure 34 on page 156. Press the **Enter** key **twice** to remove the entry from the /etc/ppp/pap-secrets file.

Sample /etc/ppp/pap-secrets File: Figure 35 is an example of the /etc/ppp/pap-secrets file that is created or modified by using the above SMIT panels.

```
tdm * password
root sito @/etc/ppp/passwd.root
* * hello
```

Figure 35. /etc/ppp/pap-secrets File

8.1.3 Challenge-Handshake Authentication Protocol

CHAP sends a challenge to the peer with an initial value. The peer must then respond with a different value calculated using a *one way hash* function (using an MD5 algorithm). Using the response value, the Authenticator compares it with what it expects the peer to respond with; if the values match, the authentication is acknowledged. The password is not transmitted over the link, rather it, is used as one of the inputs to the hash algorithm.

Unlike PAP, a CHAP Authenticator can send challenges at any time, even after the link is established and the peer must respond to those challenges.

CHAP provides protection against playback attacks through the use of an incrementally changing identifier and a variable challenge value. The use of repeated challenges is intended to limit the exposure to any single attack, and the Authenticator is in control of the frequency and timing of the challenges.

The AIX V4.2 implementation of CHAP currently supports only the MD5 encryption algorithm for authentication.

8.1.3.1 Configuring CHAP Using SMIT

By entering the fastpath `smit pppchap`, the panel in Figure 36 is displayed.

```
                              CHAP Authentication

Move cursor to desired item and press Enter.

  Add a User
  Change / Show a User
  Remove a User

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

Figure 36. SMIT CHAP Authentication Panel

Add a User: The fastpath `smit addchapuser` will display the Add CHAP User panel as shown in Figure 37 on page 159.

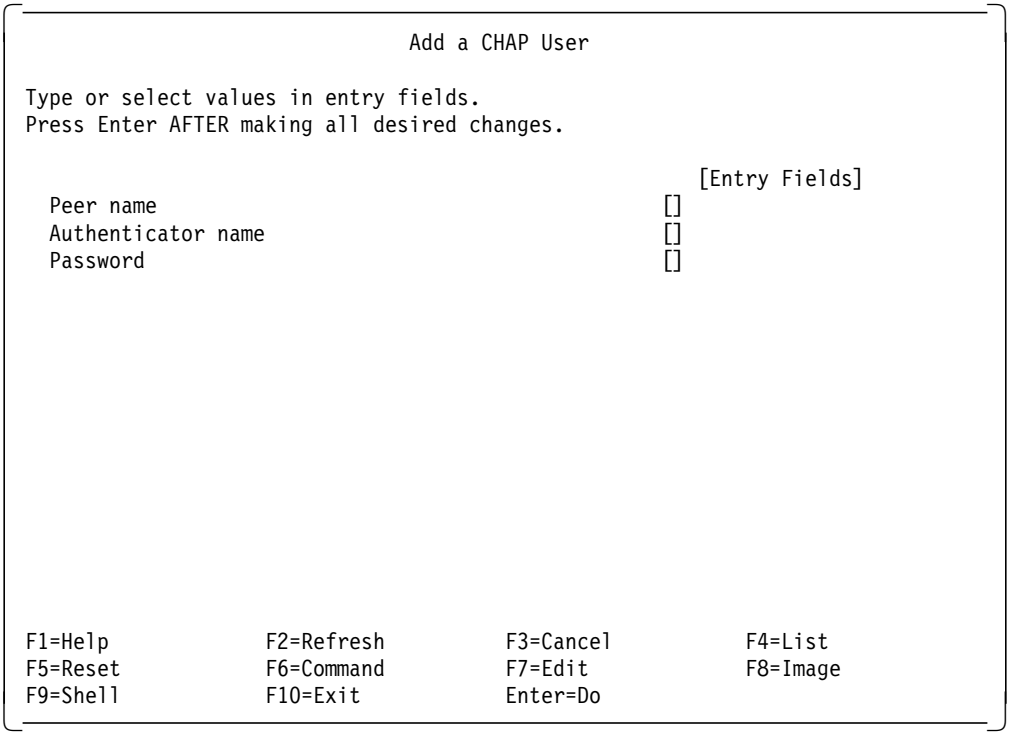


Figure 37. SMIT Add a CHAP User Panel

Select **Add a User** to create a /etc/ppp/chap-secrets file if it does not already exist, or to add to an existing /etc/ppp/pap-secrets file. The /etc/ppp/pap-secrets file is used to authenticate a peer or to provide the values to be sent to a server for authentication.

Peer name is the PPP subsystem name of the peer host being authenticated. A value of asterisk (*) can be used to match any peer host.

Authenticator name is the PPP subsystem name of the authenticating host. A value of asterisk (*) can be used to match any peer host. Both the authenticating host and the peer host must have entries in the /etc/ppp/chap-secrets file for successful CHAP authentication.

Password is an alphanumeric string. If the password begins with an @, the text that follows must be the file name containing the password.

Change/Show a User: The fastpath `smit listchapuser` will display the Change/Show a CHAP User panels as shown in Figure 38 and Figure 39.

The first panel displays a list of existing CHAP users whose details can be listed or updated.

```
                                CHAP User List

   Move cursor to desired item and press Enter.

     peer kira       auth. sito      password deepspace9

   F1=Help                 F2=Refresh              F3=Cancel
   F8=Image                F10=Exit                Enter=Do
   /=Find                  n=Find Next
```

Figure 38. SMIT CHAP User List Panel

Use the arrow keys to highlight the user whose details you wish to change, and press the **Enter** key. The Change/Show panel will be displayed, and you can update any information associated with this entry. If the entry is changed, the original entry is deleted, and a new entry is appended to the /etc/ppp/chap-secrets file.

```
                            Change / Show a CHAP User

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

                                                       [Entry Fields]
  Peer name                                           [kira]
  Authenticator name                                  [sito]
  Password                                            [deepspace9]









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

Figure 39. SMIT Change/Show a CHAP User Panel

Remove a User: The fastpath `smit rmchapuser` displays the same list panel as shown in Figure 38 on page 160 to display a list of existing CHAP users who can be removed. Use the arrow keys to highlight the user whose details you wish to remove, and press the **Enter** key. The Remove a CHAP User panel is similar to the panel in Figure 39 on page 160. Press the **Enter** key **twice** to remove the entry from the /etc/ppp/chap-secrets file.

Sample /etc/ppp/chap-secrets File: Figure 40 is an example of the /etc/ppp/chap-secrets file that is created or modified by using the above SMIT panels.

```
kira sito stds9
cisco odo @/etc/ppp/odo.password
```

Figure 40. /etc/ppp/chap-secrets File

8.2 Trouble Shooting PPP

The following sections will assist you in troubleshooting your PPP configuration

8.2.1.1 syslog

All PPP subsystem errors are logged using the syslog facilities of the operating system. To enable these messages, one must have enabled logging of errors in the /etc/syslog.conf file. By default the system does not do this. Refer to the InfoExplorer online documentation for configuring syslog.

8.2.1.2 pppcontrold Error Information

The pppcontrold logs its error information to syslog. If configuration or loading errors occur, relevant message are logged and the execution of the control daemon is terminated.

Errors in both the LCP configuration and IP configuration are also logged via syslog. In addition, there is debug information that is logged if the debug logging level is turned on.

Below is a sample syslog output for a system which has server IP interfaces configured, but no server connections defined in the LCP configuration.

```
Oct 24 10:54:58 r11OOrs /usr/sbin/pppcontrold[4870]: sysname r11 OOrsppp argv r11OOrsppp
Oct 24 10:54:59 r11OOrs /usr/sbin/pppcontrold[4870]: INVALID NUMBER of Servers configured
```

8.2.1.3 pppattachd Error Information

Error information logged by the attachment process (pppattachd) is as follows:

- Invalid options, parsing problems, option syntax errors
- Specifying a device that is invalid
- Device access problems
- PPP subsystem not available (Failure to access the device /dev/slcp when binding to the subsystem)
- Problems with process control (becoming a daemon)
- Failure of the connection program
- Errors binding to the LCP multiplexor

8.2.1.4 Verifying that the Subsystem is Running

To verify that the subsystem is running the following steps should be performed.

1. **lssrc -s pppcontrold** Lists the status of the pppcontrold with respect to the System Resource Controller.
2. **ps -ef | grep ppp** Search the list of running processes for the PPP subsystem.

If the subsystem is not running, check the syslog for error messages from pppcontrold.

8.2.2 PPP Trace Hooks

The following trace hooks are defined for the streams modules and drivers which make up the PPP subsystem.

- **2AB** TCP/IP interface layer traces
- **2AC** HDLC Encapsulation tracing
- **2AD** LCP multiplexor
- **2AE** LCP and IPCP protocol data

8.2.2.1 In the Event of a Problem

Following is a list of data that should be collected that will assist your support person in determining and resolving your problem.

1. Output of `lssrc -a`
2. Content of the syslog output showing startup of the subsystem as well as the attempted attachment
3. The exact command issued to bind with the PPP subsystem
4. The files /etc/ppp/lcp_config and /etc/ppp/if_conf
5. Traces with the hooks `2AB, 2AC, 2AD` and `2AE` of the start of the subsystem, and any failed connection
6. Information on the system that is being connected to (What type is the peer)
7. Whether or not the local system is acting as a client or a server

8.2.2.2 pppdial Interaction Problems

Often a connection will not be fully established because of a problem with the `pppdial` dialogue. Problems interacting with modems are not considered PPP problems. The following steps should be performed to ensure that the system to modem communications are in good shape:

- Use the `cu` command to attach to the tty and verify that the modem is responding to commands. Program the modem correctly.
- Using the `cu` command send each command to the modem that the dialogue would be performing automatically. Ensure that each of the expect-send pairs are understood and unique.

One common problem is, many modems indicate a connection and if a new line is sent too fast the modem drops the connection. Delays can be input into the dialogue for send strings by putting `d` for each one second of delay that is required before the data in the string. For example this script was used to connect overseas and the long distance required significant delays before sending data.

`File overseas.dial`

```
''
at
OK
atdt9,0119997989
CONNECT
\d\d\d\d\d\d\d\n
''
\d\d\d\d\d\d\d\d\n
TIMEOUT 10
User--User--User--User--User
userid
Pass
password
ibm82 ppp
```

The command that was run is as follows:

```
#/usr/sbin/pppattachd client /dev/tty0 connect
"/usr/sbin/pppdial -t 300 -v -f overseas.dial"
#
```

Note in the command for the dial there were several options specified before dialog input file. They are:

-t 300 Set the timeout value for the expect string to the number of seconds specified.

-v Verbose output. All sent data and input data are logged to syslog as debug output. In order for this option to work, the syslog subsystem must have configured for debug output. Refer to the InfoExplorer documentation on the syslog subsystem.

8.2.2.3 Connection Established, but PPP LCP Negotiation does not Start

This is usually caused by one of two things.

1. If the syslog error output for the attachment process indicates that a failure accessing the device /dev/slcp occurred, then the pppcontrold has not been started. Start the subsystem and reconnect.
2. If the local side successfully bound to the LCP mux, then the remote side may not have started PPP. Many implementations require that a command be executed to start PPP (similar to the attachment process in AIX). The traces are invaluable for analyzing this condition. Before taking traces, connect to the system with cu. If PPP is actually started on that system, after the login, binary data will be sent and the display will show what appears to be garbage. This is a good sign, and at this time one must collect the information specified, including the traces.

8.3 UCB Sendmail, Version 8.7

Sendmail in AIX Version 3.2.5 and AIX Version 4.1 is based on UCB Sendmail, Version 5.64. AIX V4.2 contains an implementation of UCB Sendmail Version 8.7 which includes the service extensions to the Simple Mail Transfer Protocol (SMTP) as outlined in the following RFCs:

RFC1651 SMTP Service Extensions

RFC1652 SMTP Service Extension for 8bit-MIMEtransport

RFC1653 SMTP Service Extension for Message Size Declaration

8.3.1 SMTP Service Extensions

RFC1651 describes the addition of a new command, along with a registry of service extensions.

8.3.1.1 EHLO Command

A system using SMTP would normally start all dialogues with the HELO command. An SMTP client supporting the SMTP Service Extensions should start SMTP sessions by issuing the EHLO command instead of the HELO command. If the SMTP server does not recognize the EHLO command, the server should stay in its current state, and the client should issue either a HELO or QUIT command.

8.3.1.2 Registry of SMTP Service Extensions

The IANAs (Internet Assigned Numbers Authority) initial registry of SMTP Service Extensions consists of the entries listed in Table 4.

Table 4. Registry of SMTP Service Extensions

Service Ext	EHLO Keyword	Parameters	Verb
Send	SEND	none	SEND
Send or Mail	SOML	none	SOML
Send and Mail	SAML	none	SAML
Expand	EXPN	none	EXPN
Help	HELP	none	HELP
Turn	TURN	none	TURN

The mandatory SMTP commands are HELO, MAIL, RCPT, DATA, RSET, VRFY, NOOP, and QUIT.

8.3.2 8bit-MIMEtransport

MIME (Multipurpose Internet Mail Extensions) is described in part in RFC1521 to provide mechanisms for specifying and describing the format of Internet message bodies. RFC1521 redefines the format of message bodies to allow multi-part textual and non-textual message bodies to be represented and exchanged without loss of information. It also provides facilities to, include multiple objects in a single message, represent body text in character sets other than US-ASCII, represent formatted multi-font text messages, represent non-textual material such as images and audio fragments, and generally to facilitate later extensions defining new types of Internet mail for use by cooperating mail agents.

If an SMTP client wishes to send mail using the extended MAIL command, it first issues the EHLO command to the SMTP server. If the SMTP server responds with code 250 to the EHLO command, and the response includes the EHLO keyword value 8BITMIME, the server is indicating that it supports the extended MAIL command and will accept 8-bit MIME messages.

The syntax of the extended MAIL command issued by the SMTP client is identical to the MAIL command, except that a BODY parameter must appear after the address. Only one BODY parameter may be used in a single MAIL command.

The syntax of the BODY parameter is:

```
BODY=[7BIT|8BITMIME]
```

8.3.3 Message Size Declaration

If an SMTP client wishes to send mail using the extended MAIL command, it first issues the EHLO command to the SMTP server. If the SMTP server responds with code 250 to the EHLO command, and the response includes the EHLO keyword value SIZE, then the Message Size Declaration extension is supported. If a numeric parameter follows the SIZE keyword value of the EHLO response, it indicates the size of the largest message that the server is willing to accept.

The extended MAIL command is issued by a client when it wishes to inform a server of the size of the message to be sent. The syntax of the extended MAIL

command issued by the SMTP client is identical to the MAIL command, except that a SIZE parameter appears after the address. Only one SIZE parameter may be used in a single MAIL command. The value associated with the SIZE parameter is a decimal representation of the declared message size in octets. This number should include the message header, body, and the CR-LF sequences between lines, but not the SMTP DATA command's terminating dot or doubled quoting dots.

8.3.4 Sample Dialogues Using SMTP Service Extensions

```
S: <wait for connection on TCP port 25>
C: <open connection to server>
S: 220 kira.itso.ibm.com SMTP service ready
C: EHLO sito.itso.ibm.com
S: 250 kira.itso.ibm.com says hello
 ...
```

Figure 41. SMTP Initial Dialogue

The example in Figure 41 shows that the server supports service extensions but only implements those SMTP commands that are defined as mandatory.

```
S: <wait for connection on TCP port 25>
C: <open connection to server>
S: 220 kira.itso.ibm.com SMTP service ready
C: EHLO sito.itso.ibm.com
S: 250 - kira.itso.ibm.com says hello
S: 250 - EXPN
S: 250 - HELP
S: 250 - 8BITMIME
S: 250 - XONE
S: 250 - XVRB
 ...
```

Figure 42. SMTP Initial Dialogue with Extensions

The example in Figure 42 shows that the server also implements the SMTP EXPN and HELP commands, one standard service extension (8BITMIME), and two non-standard service extensions (XONE and XVRB).

```
S: <wait for connection on TCP port 25>
C: <open connection to server>
S: 220 kira.itso.ibm.com SMTP service ready
C: EHLO sito.itso.ibm.com
S: 250 kira.itso.ibm.com says hello
S: 250 8BITMIME
C: MAIL FROM:<tdm@sito.itso.ibm.com> BODY=8BITMIME
S: 250 <tdm@sito.itso.ibm.com>... Sender and 8BITMIME ok
C: RCPT TO:<gonzo@kira.itso.ibm.com>
S: 250 <gonzo@kira.itso.ibm.com>... Recipient ok
C: DATA
S: 354 Send 8BITMIME message, ending in CRLF.CRLF.
 ...
C: .
S: 250 OK
C: QUIT
S: 250 Goodbye
```

Figure 43. SMTP Dialogue with 8bit-MIME

The example in Figure 43 shows the use of the 8bit-MIMEtransport service extension.

```
S: <wait for connection on TCP port 25>
C: <open connection to server>
S: 220 kira.itso.ibm.com SMTP service ready
C: EHLO sito.itso.ibm.com
S: 250 kira.itso.ibm.com says hello
S: 250 EXPN
S: 250 HELP
S: 250 SIZE 1000000
C: MAIL FROM:<tdm@sito.itso.ibm.com> SIZE=500000
S: 250 Address Ok.
C: RCPT TO:<gonzo@kira.itso.ibm.com>
S: 250 gonzo@kira.itso.ibm.com OK, can accommodate 500000 byte message
C: RCPT TO:<gonzo@ro.itso.ibm.com>
S: 552 Channel size limit exceeded: gonzo@RO.ITSO.IBM.COM
C: RCPT TO:<gonzo@vaatrik.itso.ibm.com>
S: 452 Insufficient channel storage: vaatrik.itso.ibm.com
C: DATA
S: 354 Send 8BITMIME message, ending in CRLF.CRLF.
 ...
C: .
S: 250 some recipients OK
C: QUIT
S: 250 Goodbye
```

Figure 44. SMTP Dialogue with Message Size Declaration

The example in Figure 44 shows the use of the Message Size Declaration service extension.

8.4 Network Time Protocol, Version 3 (NTP)

NTP Version 3 has been implemented in AIX V4.2 as a series of tools that provide client/server network clock synchronization. This implementation conforms to RFC1305 (Network Time Protocol Version 3). NTP provides for accurate network time by synchronizing a set of network clocks using distributed clients and servers. NTP can be used to synchronize systems in NCS cells, and services that interact with NCS, or in services that require accurate time synchronizing such as DCE and NIM.

NTP is built on the Internet Protocol (IP) and User Datagram Protocol (UDP) which provides a connectionless transport mechanism. It is specifically designed to maintain accuracy and robustness, even when used over typical Internet paths involving multiple gateways, highly dispersive delays and unreliable networks.

The system clock is currently the only supported reference clock in AIX V4.2.

8.4.1 AIX 4.2 NTP Files and Commands

The following files and commands are associated with this implementation of NTP:

ntpq	Starts the NTP query program.
xntpd	Set and maintains a UNIX system time-of-day in compliance with Internet standard time servers. System Resource Contol (SRC) support has been added for starting, stopping and listing the status of the xntpd daemon.
xntpdc	Starts the query/control program for the Network Time Protocol Daemon (xntpd).
ntpdate	Sets the local time and date by polling specified NTP servers.
ntptrace	Traces a chain of NTP hosts back to the master time source.
ntp.conf file	Controls how the xntp daemon operates and behaves.
ntp.drift file	Records current value of intrinsic frequency error for each invocation of the NTP daemon. The NTP daemon reinitializes based on the current value recorded here.
ntp.keys file	Optional file for configuring additional NTP security.

8.4.2 NTP Configuration

As noted in 8.4.1, “AIX 4.2 NTP Files and Commands” on page 168, the ntp.conf file controls how the xntp daemon operates and behaves. A default ntp.conf file can be found in the /etc directory and will look similar to Figure 45. The default file configures the local host as a broadcast client, which specifies that the local server listen for broadcast messages on the local network in order to discover other servers on the same subnet. When the local server hears a broadcast message for the first time, it measures the nominal network delay using a brief client/server exchange with the remote server, then enters the broadcastclient mode, where it listens for and synchronizes to succeeding broadcast messages

```
# @(#)48  1.2  src/tcpip/etc/ntp.conf, ntp, tcpip420 2/16/96 10:16:34
#
#   COMPONENT_NAME: ntp
#
#   FUNCTIONS: none
#
#   ORIGINS: 27,176
#
#
#   (C) COPYRIGHT International Business Machines Corp. 1996
#   All Rights Reserved
#   Licensed Materials - Property of IBM
#   US Government Users Restricted Rights - Use, duplication or
#   disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
#
#
#
#
# Default NTP configuration file.
#
#   Broadcast client, no authentication.
#
broadcastclient
driftfile /etc/ntp.drift
tracefile /etc/ntp.trace
```

Figure 45. Default /etc/ntp.conf File

The /etc/ntp.conf file shown in Figure 46 configures the local server to act as a client.

```
#  If you are configuring a client, use "server" entries to
#  synchronize to NTP servers.  For example, server1, server2,
#  and server3.
#
server         kira.itsc.ibm.com
server         ro.itsc.ibm.com
server         vaatrik.itsc.ibm.com
```

Figure 46. /etc/ntp.conf File Client Configuration

The server keyword specifies that the local server operate in client mode with the remote server specified by HostAddress. In this mode, the local server can be synchronized to the remote server, but the remote server can never be synchronized to the local server.

The /etc/ntp.conf file in Figure 47 shows a more complex configuration, including additional authentication.

```
peer kira.itsc.ibm.com key 22
peer ro.itsc.ibm.com key 4
peer vaatrik.itsc.ibm.com key 6


authenticate yes         # enable authentication
keys /usr/local/etc/ntp.keys  # path for key file
trustedkey 1 2 14 15     # define trusted keys
requestkey 15            # key (6) for accessing server variables
controlkey 15            # key (7) for accessing server variables

authdelay 0.000094       # authentication delay
```

Figure 47. /etc/ntp.conf Additional Authentication Entries

The peer keyword specifies that the local server operate in symmetric active mode with the remote server specified by HostAddress. In this mode, the local server can be synchronized to the remote server, or the remote server can be synchronized to the local server. Use this method in a network of servers where, depending on various failure scenarios, either the local or remote server host may be the better source of time.

The authenticate yes line enables authentication processing, while the keys /usr/local/etc/ntp.keys specifies the path to the keys file. The trustedkey declaration identifies those keys that are known to be uncompromised; the remainder presumably represent the expired or possibly compromised keys. Both sets of keys must be declared by key identifier as shown in the ntp.keys file shown in Figure 48. This provides a way to retire old keys while minimizing the frequency of delicate key-distribution procedures. The requestkey 15 line establishes the key to be used for mode-6 control messages as specified in RFC-1305 and used by the ntpq utility program, while the controlkey 15 establishes the key to be used for mode-7 private control messages used by the xntpdc utility program. These keys are used to prevent unauthorized modification of daemon variables.

```
1    N    29233E0461ECD6AE    # des key in NTP format
2    M    RIrop8KPPvQvYotM    # md5 key as an ASCII random string
14   M    sundial             # md5 key as an ASCII string
15   A    sundial             # des key as an ASCII string

# the following 3 keys are identical

10   A    SeCReT
10   N    d3e54352e5548080
10   S    a7cb86a4cba80101
```

Figure 48. /etc/ntp.keys File

The above examples do not cover all the possible configurations for the ntp daemon. For more information on the keywords in the examples, and for other configurable options, you can search for ntp.conf in the InfoExplorer tool.

8.4.3 ntpdate Command

ntpdate sets the date and time using the Network Time Protocol (NTP).

Syntax:

```
ntpdate [ -b ] [ -d ] [ -s ] [ -u ]
        [ -a Keyid ] [ -e AuthenticationDelay ]
        [ -k KeyFile ] [ -o Version ] [ -p Samples ]
        [ -t TimeOut ] Server
```

The ntpdate command sets the local date and time by polling the NTP servers specified to determine the correct time. It obtains a number of samples from each server specified and applies the standard NTP clock filter and selection algorithms to select the best of the samples.

The ntpdate command makes time adjustments in one of the following ways:

1. If it determines the clock is off by more than 0.5 seconds, it steps the clock's time by calling the settimeofday subroutine. This is the preferred method at boot time.
2. If it determines the clock is off by less than 0.5 seconds, it slews the clock's time by calling the adjtime subroutine with the offset. This method tends to keep a badly drifting clock more accurate, though at some expense to stability. When running the ntpdate command on a regular basis from the cron command instead of running a daemon, doing so once every hour or two results in precise enough timekeeping to avoid stepping the clock.

The ntpdate command's reliability and precision improves dramatically with a greater number of servers. Although you can use a single server, you obtain better performance by providing at least three or four servers.

If an NTP server daemon like the xntpd daemon is running on the same host, the ntpdate command will decline to set the date.

You must have root authority on the local host to run the ntpdate command

Flags

-a Keyid Authenticate all packets using Keyid.

-b Step the clock's time by calling the settimeofday subroutine.

-d Specifies debug mode. Determines what results the ntpdate command produces without actually doing them. The results appear on the screen. This flag uses unprivileged ports.

-e AuthenticationDelay
Specifies the amount of time in seconds to delay the authentication processing.

-k Keyfile Specifies a different name for the file containing the keys when not using the default /etc/ntp.keys file.

-o Version Specifies the NTP version implementation to use when polling its outgoing packets. The values for Version can be 1, 2 or 3. The default is 3.

-p Samples Specifies the number of samples to acquire from each server. The values for Samples can be between 1 and 8 inclusive. The default is 4.

-s Specifies the use of the syslog facility to log actions instead of using standard output. Useful when running the ntpdate command with the cron command.

-t TimeOut Specifies the amount of time to wait for a response. The value given for TimeOut is rounded to a multiple of 0.2 seconds. The default is 1 second.

-u Specifies the use of an unprivileged port to send the packets from. Useful when you are behind a firewall that blocks incoming traffic to privileged ports, and you want to synchronize with hosts beyond the firewall. A firewall is a system or machine that controls the access from outside networks to a private network.

Examples

To set the local date and time by polling the NTP servers at address 9.3.149.107, enter:

```
/usr/sbin/ntpdate 9.3.149.107
```

Output similar to the following appears:

```
28 Feb 12:09:13 ntpdate &lbrk18450] : step time server 9.3.149.107 offset
38.417792 sec
```

8.4.4 ntpq Command

ntpq starts the standard Network Time Protocol (NTP) query program.

Syntax:

```
ntpq [ -i ] [ -n ] [ -p ] [ -c SubCommand ]
     [ Host ]
```

The ntpq command queries the NTP servers running on the hosts specified which implement the recommended NTP mode 6 control message format about current state and can request changes in that state. It runs either in interactive mode or by using command-line arguments. You can make requests to read and write arbitrary variables, and raw and formatted output options are available. The ntpq command can also obtain and print a list of peers in a common format by sending multiple queries to the server.

If you enter the ntpq command with one or more flags, the NTP servers running on each of the hosts specified (or defaults to local host) receive each request. If you do not enter any flags, the ntpq command tries to read commands from standard input and run them on the NTP server running on the first host specified or on the local host by default. It prompts for subcommands if standard input is the terminal.

The ntpq command uses NTP mode 6 packets to communicate with the NTP server and can query any compatible server on the network which permits it.

The ntpq command makes one attempt to retransmit requests, and will time-out requests if the remote host does not respond within a suitable time.

Specifying a flag other than -i or -n sends the queries to the specified hosts immediately. Otherwise, the ntpq command attempts to read interactive format subcommands from standard input.

Flags

-c SubCommand
Specifies an interactive format command. This flag adds SubCommand to the list of commands to run on the specified hosts. You can enter multiple -c flags.

-i Specifies interactive mode. Standard output displays prompts and standard input reads commands.

-n Displays all host addresses in dotted decimal format (x.x.x.x) rather than the canonical host names.

-p Displays a list of the peers known to the server and a summary of their state. Same as using the peers subcommand.

Examples

To start the Network Time Protocol query program in interactive mode, enter:

```
ntpq -i
```

To add a time interval of 1000 milliseconds to timestamps, enter:

```
ntpq -c "delay 1000"
```

8.4.5 ntpq Internal Subcommands

The following subcommands can only be used while running the ntpq query program.

Interactive Format Subcommands

Interactive format subcommands consist of a keyword followed by zero to four arguments. You only need to type enough characters of the full keyword to uniquely identify the subcommand. The output of a subcommand goes to standard output, but you can redirect the output of individual subcommands to a file by appending a greater-than sign (>), followed by a file name, to the command line.

Some interactive format subcommands run entirely within the ntpq query program and do not result in sending NTP mode 6 requests to a server. The data carried by NTP mode 6 messages consists of a list of items of the form:

```
Variable=Value
```

where Value is ignored, and can be omitted, in requests to the server to read variables. The ntpq query program maintains an internal list where data to be included in control messages can be assembled and sent using the readlist and writelist control message subcommands listed below:

? [**SubCommand**]
Displays command usage information. When used without SubCommand, displays a list of all the ntpq command keywords. When used with SubCommand, displays function and usage information about the subcommand.

addvars Variable [**=Value**] [ ]
Specifies the variables and their optional values to be added to the internal data list. If adding more than one variable, the list must be separated by commas and not contain spaces.

authenticate yes | no
Specifies whether to send authentication with all requests or not. Normally the ntpq query program does not authenticate requests unless they are write requests.

clearvars Removes all variables from the internal data list.

cooked Displays all results received from the remote server reformatted. A trailing question mark (?) marks variables that do not have decodeable values.

debug more | less | off
Turns the ntpq query program debugging on or off. The more and less options control the verbosity of the output. If you enter this subcommand without an argument, it prints the current setting for this subcommand.

delay MilliSeconds
Specifies the time interval to add to timestamps included in requests which require authentication. This subcommand enables unreliable server reconfiguration over long delay network paths or between machines whose clocks are unsynchronized. If you enter this subcommand without an argument, it prints the current setting for this subcommand.

host HostName Specifies the host to send queries to. HostName may be either a host name or a numeric address. If you enter this subcommand without an argument, it prints the current setting for this subcommand.

hostnames yes | no
Specifies whether to output the hostname (yes) or the numeric address (no). Defaults to yes unless the -n flag is used. If you enter this subcommand without an argument, it prints the current setting for this subcommand.

keyid Number Specifies the server key number to use to authenticate configuration requests. If you enter this subcommand without an argument, it prints the current setting for this subcommand.

ntpversion 1 | 2 | 3
Specifies the NTP version implementation to use when polling its packets. The default is 3. If you enter this subcommand without an argument, it prints the current setting for this subcommand.

Note: Mode 6 control messages and modes did not exist in NTP Version 1.

passwd Prompts you to type in the NTP server authentication password to use to authenticate configuration requests.

quit Exits the ntpq query program.

raw Displays all results received from the remote server without formatting. Only transforms non-ASCII characters into printable form.

rmvars Variable [**=Value**] [ ]
Specifies the variables and their optional values to be removed from the internal data list. If removing more than one variable, the list must be separated by commas and not contain spaces.

timeout MilliSeconds
Specifies the time-out period for responses to server queries. The default is 5000 milliseconds. If you enter this subcommand without an argument, it prints the current setting for this subcommand.

Note: Since `ntpq` query program retries each query once after a time-out, the total waiting time for a time-out is twice the time-out value set.

Control Message Subcommands

Each peer known to an NTP server has a 16-bit integer association identifier assigned to it. NTP control messages which carry peer variables must identify the peer that the values correspond to by including its association ID. An association ID of 0 is special and indicates the variables are system variables whose names are drawn from a separate name space.

The `ntpq` control message subcommands result in one or more NTP mode 6 messages sent to the server, and outputs the data returned in some format. Most subcommands currently implemented send a single message and expect a single response. The current exceptions are the `peers` subcommand, which sends a preprogrammed series of messages to obtain the data it needs, and the `mreadlist` and `mreadvar` subcommands, which iterate over a range of associations.

associations Obtains and prints a list of association identifiers and peer statuses for in-spec peers of the server being queried. The list is printed in columns. The first column contains the index numbering the associations from 1 for internal use. The second column contains the actual association identifier returned by the server. The third column contains the status word for the peer. The rest of the columns contain data decoded from the status word.

Note: The data returned by the associations subcommand is cached internally in the `ntpq` query program. When dealing with servers that use difficult association identifiers, use the index as an argument, in the form `&index`, as an alternative to the association identifier.

clockvar or cv [**AssocID**] [**Variable** [**=Value**] **...**]
Displays a list of the server's clock variables. Servers which have a radio clock or other external synchronization respond positively to this. To request the system clock variables, leave AssocID blank or enter 0. If the server treats clocks as pseudo-peers and can possibly have more than one clock connected at once, referencing the appropriate peer association ID shows the variables of a particular clock. Omitting the variable list causes the server to return a default variable display.

lassocations Displays a list of association identifiers and peer statuses for all associations for which the server is maintaining state. This subcommand differs from the associations subcommand only for servers which retain state for out-of-spec client associations.

lpassociations Displays data for all associations, including out-of-spec client associations, from the internally cached list of associations.

lpeers Displays a summary of all associations the server maintains state for, similar to the peers subcommand. It may produce a longer list of peers from out-of-spec client servers.

mreadvar or mrv AssocID AssocID [**Variable** [**=Value**] **...**]
Displays the values of the specified peer variables for each server in the range of given nonzero association IDs. The association list cached by the most recent associations command determines the range.

mreadlist or mrl AssocID AssocID
Displays the values of the specified peer variables in the internal variable list for each server in the range of given nonzero association IDs. The association list cached by the most recent associations command determines the range.

opeers An old form of the peers subcommand. Replaces the reference ID with the local interface address.

passociations Displays association data concerning in-spec peers from the internally cached list of associations. This subcommand works like the associations subcommand except that it displays the internally stored data rather than making a new query.

peers Displays a list of in-spec peers of the server and a summary of each peer's state. Summary information includes:

- Address of the remote peer
- Reference ID (0.0.0.0 for an unknown reference ID)
- Stratum of the remote peer (a stratum of 16 indicates the remote peer is unsynchronized)
- Type of the peer (local, unicast, multicast or broadcast)
- Time the last packet was received, the polling interval (seconds)
- Polling interval (seconds)
- Reachability register (octal)
- Current estimated delay, offset and dispersion of the peer (seconds)

The character in the left margin indicates the fate of this peer in the clock selection process:

- ' ' (space), discarded due to high stratum and/or failed sanity checks
- 'x', designated falseticker by the intersection algorithm
- '.' (dot), culled from the end of the candidate list
- '-' (minus), discarded by the clustering algorithm
- '+' (plus), included in the final selection set
- '#' (hash), selected for synchronization but distance exceeds maximum
- '*' (asterisk), selected for synchronization
- 'o', selected for synchronization, pps signal in use

The contents of the host field may be a hostname, an IP address, a reference clock implementation name with its parameter or REFCLK (ImplementationNumber ,Parameter). Only IP addresses display when using hostnames no.

Note:

1. The peers subcommand depends on the ability to parse the values in the responses it gets. It may fail to work from time to time with servers that poorly control the data formats.
2. The peers subcommand is non-atomic and may occasionally result in spurious error messages about invalid associations occurring and terminating the command.

pstatus AssocID

Displays the names and values of the peer variables of the server with the given association by sending a read status request. The output displays the header preceding the variables, both in hexidecimal and in English.

readlist or rl [**AssocID**]

Displays the values of the peer variables in the internal variable list of the server with the given association. To request the system variables, leave AssocID blank or enter 0. If the internal variable list is empty, the server returns a default variable display.

readvar or rv [**AssocID**] [**Variable** [**=Value**] **...**]

Displays the values of the specified peer variables of the server with the given association by sending a read variables request. To request the system variables, leave AssocID blank or enter 0. Omitting the variable list causes the server to return a default variable display.

writevar [**AssocID**] [**Variable** [**=Value**] **...**]

Writes the values of the specified peer variables to the server with the given association by the specified peer variables to the server with the given association by sending a write variables request.

writelist [**AssocID**]

Writes the values of the peer variables in the internal variable list of the server with the given association.

8.4.6 ntptrace Command

`ntptrace` traces a chain of Network Time Protocol (NTP) hosts back to their master time source.

Syntax:

```
ntptrace [ -d ] [ -n ] [ -v ] [ -r Retries ]
         [ -t TimeOut ] [ Server ]
```

The `ntptrace` command determines where a given NTP server gets its time from, and follows the chain of NTP servers back to their master time source (that is, stratum 0 server).

Flags

-d	Turns on debugging output
-n	Outputs host IP addresses instead of host names
-r Retries	Specifies the number of retransmission attempts for each host. The default is 5.
-t TimeOut	Specifies the retransmission timeout in seconds. The default is 2 seconds.
-v	Specifies verbose mode.

Examples

To trace where the local host NTP server gets its time from, enter:

```
ntptrace
```

Output similar to the following appears:

```
localhost: stratum 4, offset 0.0019529, synch distance 0.144135
server2.bozo.com: stratum 2, offset 0.0124263, synch distance 0.115784
usndh.edu: stratum 1, offset 0.0019298, synch distance 0.011993, refid
'WWVB'
```

On each line, the fields are:

- Host's stratum
- Time offset between that host and the local host, as measured by the ntptrace command, (this is why it is not always zero for localhost)
- Host's synchronization distance, which is a measure of the quality of the clock's time
- Reference clock ID, this only applies to stratum-1 servers

All times are given in seconds.

8.4.7 xntpdc Command

Starts the query/control program for the Network Time Protocol daemon, xntpd.

Syntax:

```
xntpdc [ -i ] [ -l ] [ -n ] [ -p ]
       [ -s ] [ -c SubCommand ] [ Host ]
```

The xntpdc command queries the xntpd daemon about its current state and requests changes to that state. It runs either in interactive mode or by using command-line arguments. The xntpdc command interface displays extensive state and statistics information. In addition, nearly all the configuration options which you can specify at start up using the xntpd daemon's configuration file, you can also specify at run time using the xntpdc command.

If you enter the xntpdc command with one or more request flags, the NTP servers running on each of the hosts specified (or defaults to local host) receive each request. If you do not enter any request flags, the xntpdc command tries to read commands from standard input and run them on the NTP server running on the first host specified or on the local host by default. It prompts for subcommands if standard input is the terminal.

The xntpdc command uses NTP mode 7 packets to communicate with the NTP server and can query any compatible server on the network which permits it.

The xntpdc command makes no attempt to retransmit requests, and will time-out requests if the remote host does not respond within a suitable time.

Specifying a flag other than -i or -n sends the queries to the specified hosts immediately. Otherwise, the xntpdc command attempts to read interactive format commands from standard input.

Flags

-c SubCommand
: Specifies an interactive format command. This flag adds SubCommand to the list of commands to run on the specified hosts. You can enter multiple -c flags.

-i
: Specifies interactive mode. Standard output displays prompt and standard input reads commands.

-l
: Displays a list of the peers known to the servers. This is the same as the listpeer subcommand.

-n
: Displays all host addresses in dotted decimal format (0.0.0.0) rather than the canonical host names.

-p
: Displays a list of the peers known to the server and a summary of their state. This is the same as the peers subcommand.

-s
: Displays a list of the peers known to the server and a summary of their state but in a format different from the -p flag. This is the same as the dmpeers subcommand.

Examples

To start the query/control program for the Network Time Protocol daemon, enter:

```
xntpdc
```

To display the statistic counters of the peer at address 127.127.1.0 on host 9.3.149.107, enter:

```
xntpdc -c "pstats 127.127.1.0" 9.3.149.107
```

Output similar to the following appears:

```
remote host:           LOCAL(0)
local interface:       127.0.0.1
time last received:    49s
time until next send:  15s
reachability change:   818s
packets sent:          13
packets received:      13
bad authentication:    0
bogus origin:          0
duplicate:             0
bad dispersion:        4
bad reference time:    0
bad reference time:    0
candidate order:       1
```

xntpdc Internal Subcommands

You can run a number of interactive format subcommands entirely within the xntpdc command that do not send NTP mode 7 requests to a server. Interactive format subcommands consist of a keyword followed by zero to four arguments. You only need to type enough characters of the full keyword to uniquely identify the subcommand. The output of a subcommand goes to standard output, but you can redirect the output of individual subcommands to a file by appending a greater-than sign (>), followed by a file name, to the command line.

The following subcommands can only be used while running the xntpdc query program.

-? [**SubCommand**]
: Displays command usage information. When used without SubCommand, displays a list of all the xntpdc command keywords. When used with SubCommand, displays function and usage information about the command.

help [**SubCommand**]
: Same as the -? [SubCommand] subcommand.

delay Milliseconds
: Specifies the time interval to add to timestamps included in requests which require authentication. This subcommand enables unreliable server reconfiguration over long delay network paths or between machines whose clocks are unsynchronized. If you enter this subcommand without an argument, it prints the current setting for this subcommand.

host HostNames
: Specifies the host to send queries to. HostName may be either a host name or a numeric address. If you enter this subcommand without an argument, it prints the current setting for this subcommand.

hostnames yes | no
: Specifies whether to display the host name (yes) or the numeric address (no). The default is yes unless the -n flag is used. If you enter this subcommand without an argument, it prints the current setting for this subcommand.

keyid Number
: Specifies the server key number to use to authenticate configuration requests. If you enter this subcommand without an argument, it prints the current setting for this subcommand.

passwd
: Prompts you to type in the NTP server authentication password to use to authenticate configuration requests.

quit
: Exits the xntpdc query program.

timeout Milliseconds
: Specifies the time-out period for responses to server queries. The default is 8000 milliseconds. If you enter this subcommand without an argument, it prints the current setting for this subcommand.

Query Subcommands

The xntpdc query subcommands result in sending NTP mode 7 packets containing requests to the server. These subcommands are read-only (they do not modify the server configuration state).

clkbug ClockPeerAddress [**Addr2**] [**Addr3**] [**Addr4**]
Displays debugging information for a reference clock driver. Some clock drivers provide this information which is mostly undecodable without a copy of the driver source in hand.

clockbug ClockPeerAddress [**Addr2**] [**Addr3**] [**Addr4**]
Displays information concerning a peer clock. The values obtained provide information on the setting of fudge factors and other clock performance information.

dmpeers Displays a list of peers for which the server is maintaining state, along with a summary of that state. Identical to the output of the peers subcommand except for the character in the leftmost column. Characters only appear beside peers which were included in the final stage of the clock selection algorithm.

The possible character in the leftmost column are:

1. ′.′ (dot), indicates this peer was cast off in the falseticker detection.
2. ′+′ (plus), indicates the peer made it through.
3. ′*′ (asterisk), denotes the peer the server is currently synchronizing with.

iostats Displays statistics counters maintained in the input-output module.

kerninfo Displays kernel phase-lock loop operating parameters. This information is available only if the kernel of the hosts being generated has been specially modified for a precision timekeeping function.

listpeers Displays a brief list of the peers for which the server is maintaining state. These include all configured peer associations as well as those peers whose stratum is such that the server considers them to be possible future synchonization candidates.

loopinfo [**oneline | multiline**]
Displays the values of selected loop filter variables. The loop filter is the part of NTP that adjusts the local system clock. The offset is the last offset given to the loop filter by the packet processing code. The frequency is the frequency error of the local clock in parts-per-million (ppm). The poll adjust controls the stiffness (resistance to change) of the phase-lock loop and the speed at which it can adapt to oscillator drift. The watchdog timer is the number of elasped seconds since the last sample offset given to the loop filter. The oneline and multiline options specify the format to display this information. The multiline option is the default.

memstats Displays statistics counters related to memory allocation code.

monlist Displays traffic counts collected and maintained by the monitor facility.

peers Displays a list of peers for which the server is maintaining state, along with a summary of that state. Summary information includes:

- Address of the remote peer
- Reference ID (0.0.0.0 for an unknown reference ID)

- Stratum of the remote peer (a stratum of 16 indicates the remote peer is unsynchronized)
- Polling interval (seconds)
- Reachability register (octal)
- Current estimated delay, offset and dispersion of the peer (seconds)

The character in the left margin indicates the mode this peer entry is in:

- '+' (plus), symmetric active
- '-' (minus), symmetric passive
- '=' (equals), remote server polled in client mode
- '^' (caret) server is broadcasting to this address
- '~' (tilde), remote peer is sending broadcasts
- '*' (asterisk), marks the peer the server is currently synchronized to

The contents of the host field may be a host name, an IP address, a reference clock implementation name with its parameter or REFCLK (ImplementationNumber, Parameter). Only IP addresses display when using hostanames no.

pstats PeerAddress [**Addr2**] [**Addr3**] [**Addr4**]
Displays per-peer statistic counters associated with the specified peers.

reslist Displays the server's restriction list which may help to understand how the restrictions are applied.

sysinfo Displays a variety of system state variables related to the local server. All except the last four lines are described in the NTP Version 3 specification, RFC 1305. The system flags show various system flags, some of which can be set and cleared by the enable and disable configuration statements.

Stability is the residual frequency error remaining after applying the system frequency correction. You use it for maintenance and debugging. In most architectures, this value will initially decrease from as high as 500 ppm to a nominal value in the range .01 to 0.1 ppm. If it remains high for some time after starting the daemon, something may be wrong with the local clock, or the value of the kernel variable Tick may be incorrect. The broadcastdelay shows the default broadcast delay, as set by the broadcastdelay configuration statement, while the authdelay shows the default authentication delay, as set by the authdelay configuration statement.

sysstats Displays statistics counters maintained in the protocol module.

timerstats Displays statistics counters maintained in the timer/event queue support code.

Run-time Configuration Requests Subcommands

The server authenticates all requests which cause state changes in the server by using a configured NTP key. The server can also disable this facility by not configuring a key. You must make the key number and the corresponding key known to the `xtnpdc` command. You can do this by using the `keyid` and `passwd` subcommands, which prompts at the terminal for a password to use as the encryption key. The `xtnpdc` command will also prompt you automatically for both

the key number and password the first time you give a subcommand which would result in an authenticated request to the server. Authentication not only verifies that the requester has permission to make such changes, but also protects against transmission errors.

Authenticated requests always include a timestamp in the packet data, as does the computation of the authentication code. The server compares this timestamp to the time at which it receives the packet.

The server rejects the request if they differ by more than 10 seconds. This makes simple replay attacks on the server, by someone able to overhear traffic on your LAN, much more difficult. It also makes it more difficult to request configuration changes to your server from topologically remote hosts.

While the reconfiguration facility works well with a server on the local host, and may work adequately between time-synchronized hosts on the same LAN, it works very poorly for more remote hosts. So, if you choose reasonable passwords, are careful in the distribution and protection of keys and apply appropriate source address restrictions, the run-time reconfiguration facility should provide an adequate level of security.

The following subcommands all make authenticated requests.

addpeer PeerAddress [**Keyid**] [**Version**] [**prefer**]
: Adds a configured peer association operating in symmetric active mode at the specified address. You may delete an existing association with the same peer or simply convert an existing association to conform to the new configuration when using this subcommand. If the Keyid is a nonzero integer, all outgoing packets to the remote server will have an authentication field attached encrypted with this key. To specify no authentication, enter Keyid as 0 or leave blank. The values for Version can be 1, 2 or 3, with 3 as the default. The prefer option indicates a preferred peer used primarily for clock synchronisation if possible. The preferred peer also determines the validity of the PPS signal. If the preferred peer is suitable for synchronization, so is the PPS signal.

addserver PeerAddress [**Keyid**] [**Version**] [**prefer**]
: Same as the addpeer subcommand, except that the operating mode is client.

addtrap Address [**Port**] [**Interface**]
: Sets a trap for asynchronous messages at the specified address and port number for sending messages with the specified local interface address. If you do not specify the port number, the value defaults to 18447. If you do not specify the interface address, the value defaults to the source address of the local interface.

authinfo
: Displays information concerning the authentication module, including known keys and counts of encryptions and decryptions performed.

broadcast PeerAddress [**Keyid**] [**Version**]
: Same as the addpeer subcommand, except that the operating mode is broadcast. The PeerAddress can be the broadcast

address of the local network or a multicast group address assigned to NTP (224.0.1.1).

clrtrap Address [**Port**] [**Interface**]
Clears a trap for asynchronous messages at the specified address and port number for sending messages with the specified local interface address. If you do not specify the port number, the value defaults to 18447. If you do not specify the interface address, the value defaults to the source address of the local interface.

delrestrict Address Mask [**ntpport**]
Deletes the matching entry from the restrict list.

disable Option Disables various server options. Does not affect options not mentioned. The enable subcommand describes the options.

enable Option Enables various server options. Does not affect options not mentioned. You can specify one or more of the following values for Option:

- **auth** Causes the server to synchronize with unconfigured peers only if the peer has been correctly authenticated using a trusted key and key identifier. The default for this option is disable (off).
- **bclient** Causes the server to listen for a message from a broadcast or multicast server, following which an association is automatically initiated for that server. The default for this argument is disable (off).
- **monitor** Enables the monitoring facility, with default enable (on).
- **pll** Enables the server to adjust its local clock, with default enable (on). If not set, the local clock free-runs at its intrinsic time and frequency offset. This option is useful when the local clock is controlled by some other device or protocol and NTP is used only to provide synchronization to other clients.
- **stats** Enables statistics facility filegen, with default enable (on).

fudge PeerAddress [**Time1**] [**Time2**] [**Stratum**] [**Refid**]
Provides a way to set certain data for a reference clock.

Time1 and Time2 are in fixed point seconds and used in some clock drivers as calibration constants.

Stratum is a number in the range zero to 15 and used to assign a nonstandard operating stratum to the clock.

Refid is an ASCII string in the range one to four characters and used to assign a nonstandard reference identifier to the clock.

monitor yes | no
Enables or disables the monitoring facility. A monitor no subcommand followed by a monitor yes subcommand is a good way of resetting the packet counts.

readkeys Purges the current set of authentication keys and obtains a new set by rereading the keys file specified in the xntpd configuration file. This allows you to change encryption keys without restarting the server.

reset Module Clears the statistics counters in various modules of the server. You can specify one or more of the following values for Module:

- io
- sys
- mem
- timer
- auth
- allpeers

restrict Address Mask Option

Adds the values of Option to an existing restrict list entry, or adds a new entry to the list with the specified Option. The Mask option defaults to 255.255.255.255, meaning that Address is treated as the address of an individual host. You can specify one or more of the following values for Option:

- **ignore** Ignore all packets from hosts which match this entry. Does not respond to queries nor time server polls.
- **limited** Specifies that these hosts are subject to client limitation from the same net. Net in this context refers to the IP notion of net (class A, class B, class C, etc.). Only accepts the first client_limit hosts that have shown up at the server and that have been active during the last client_limit_period seconds. Rejects requests from other clients from the same net. Only takes into account time request packets. Private, control, and broadcast packets are not subject to client limitation and therefore do not contribute to client count. The monitoring capability of the xntpd daemon keeps a history of clients. When you use this option, monitoring remain s active. The default value for client_limit is three. The default value fo r client_limit_period is 3600 seconds.
- **lowpriotrap** Declare traps set by matching hosts to low priority status. The server can maintain a limited number of traps (the current limit is 3), assigned on a first come, first served basis, and denies service to later trap requestors. This parameter modifies the assignment algorithm by allowing later requests for normal priority traps to override low priority traps.
- **nomodify** Ignore all NTP mode 6 and 7 packets which attempt to modify the state of the server (run time reconfiguration). Permits queries which return information.
- **nopeer** Provide stateless time service to polling hosts, but not to allocate peer memory resources to these hosts.
- **noquery** Ignore all NTP mode 6 and 7 packets (information queries and configuration requests) from the source. Does not affect time service.
- **noserve** Ignore NTP packets whose mode is not 6 or 7. This denies time service, but permits queries.
- **notrap** Decline to provide mode 6 control message trap service to matching hosts. The trap service is a subsystem of the mode 6 control message protocol intended for use by remote event logging programs.
- **notrust** Treat these hosts normally in other respects, but never use them as synchronization sources.

- **ntpport** Match the restriction entry only if the source port in the packet is the standard NTP UDP port (123).

setprecision Precision
Sets the precision which the server advertises. Precision should be a negative integer in the range -4 through -20.

traps Displays the traps set in the server.

trustkey Keyid Adds one or more keys to the trusted key list. When you enable authentication, authenticates peers with trusted time using a trusted key.

unconfig PeerAddress [**Addr2**] [**Addr3**] [**Addr4**]
Removes the configured bit from the specified peers. In many cases deletes the peer association. When appropriate, however, the association may persist in an unconfigured mode if the remote peer is willing to continue on in this fashion.

unrestrict Address Mask Option
Removes the specified options from the restrict list entry indicated by Address and Mask. The restrict subcommand describes the values for Option.

untrustkey Keyid
Removes one or more keys from the trusted key list.

8.4.8 xntpd Daemon

xntpd starts the Network Time Protocol (NTP) daemon.

Syntax:

```
xntpd [ -a ] [ -b ] [ -d ] [ -m ]
      [ -c ConfigFile ] [ -e AuthenticationDelay ]
      [ -f DriftFile ] [ -k KeyFile ] [ -l LogFile ]
      [ -p pidFile ] [ -r BroadcastDelay ]
      [ -s StatsDirectory ] [ -t TrustedKey ]
      [ -v SystemVariable ] [ -V SystemVariable ]
```

The xntpd daemon sets and maintains a UNIX system time-of-day in compliance with Internet standard time servers. The xntpd daemon is a complete implementation of the Network Time Protocol (NTP) Version 3 standard, as defined by RFC 1305, and also retains compatibility with Version 1 and 2 servers as defined by RFC 1059 and RFC 1119, respectively. The xntpd daemon does all computations in fixed point arithmetic and does not require floating point code.

The xntpd daemon reads from a configuration file (/etc/ntp.conf is the default) at startup time. You can override the configuration file name from the command line. You can also specify a working, although limited, configuration entirely on the command line, eliminating the need for a configuration file. Use this method when configuring the xntpd daemon as a broadcast or multicast client, that determines all peers by listening to broadcasts at runtime. You can display the xntpd daemon internal variables with the ntpq command (Network Time Protocol (NTP) query program). You can alter configuration options with the xntpdc command.

The xntpd daemon operates in several modes, including symmetric active/passive, client/server and broadcast/multicast. A broadcast/multicast client can automatically discover remote servers, compute one-way delay

correction factors and configure itself automatically. This mode makes it possible to deploy a group of workstations without specifying a configuration file or configuration details specific to its environment.

Flags

-a Runs in authenticate mode.

-b Listens for broadcast NTP and synchronizes to them if available.

-c ConfigFile Specifies the name of an alternate configuration file.

-d Specifies debugging mode. This flag may occur multiple times (maximum of 10), with each occurrence indicating greater detail of display.

-e AuthenticationDelay
Specifies the time, in seconds, it takes to compute the NTP encryption field on this computer.

-f DriftFile Specifies the location of the drift file.

-k KeyFile Specifies the location of the file which contains the NTP authentication keys.

-l LogFile Specifies the use of a log file instead of logging to syslog.

-m Listens for multicast messages and synchronizes to them if available. Assumes multicast address 224.0.1.1.

-p pidFile Specifies the name of the file to record the daemon′s process id. There is no default.

-r BroadcastDelay
Specifies the default delay (in seconds) if the calibration procedure fails. Normally, the xntpd daemon automatically compensates for the network delay between the broadcast/multicast server and the client.

-s StatsDirectory
Specifies the directory to use for creating statistics files.

-t TrustedKey Adds the specified key number to the trusted key list.

-v SystemVariable
Adds the specified system variable

-V SystemVariable
Adds the specified system variable listed by default.

Reference Clock Support

For the purposes of configuration, the xntpd daemon treats reference clocks in a manner analogous to normal NTP peers as much as possible. It refers to reference clocks by address, same as a normal peer is, though it uses an invalid IP address to distinguish them from normal peers. AIX Version 4.2 supports one type of reference clock, based on the system clock (type 1).

Reference clock addresses are of the form 127.127.Type.Unit where Type is an integer denoting the clock type and Unit indicates the type-specific unit number. You configure reference clocks by using a server statement in the configuration file where the HostAddress is the clock address. The key,version and ttl options are not used for reference clock support.

Reference clock support provides the fudge command, which configures reference clocks in special ways. This command has the following format:

```
fudge 127.127.Type.Unit [ time1 Seconds ] [ time2 Seconds ]
  [ stratum Integer ] [ refid Integer ]
  [ flag1 0|1 ] [ flag2 0|1 ] [ flag3 0|1 ]
  [ flag4 0|1 ]
```

The time1 and time2 options are in fixed point seconds and used in some clock drivers as calibration constants.

The stratum option is a number in the range zero to 15 and used to assign a nonstandard operating stratum to the clock. Since the xntpd daemon adds one to the stratum of each peer, a primary server ordinarily displays stratum one. In order to provide engineered backups, use the stratum option to specify the reference clock stratum as greater than zero. Except where noted, this option applies to all clock drivers.

The refid option is an ASCII string in the range one to four characters and used to assign a nonstandard reference identifier to the clock.

The binary flags: flag1, flag2, flag3 and flag4 are for customizing the clock driver. The interpretation of these values, and whether they are used at all, is a function of the needs of the particular clock driver.

Examples

To start the xntpd daemon, enter:

```
startsrc -s xntpd
```

To stop the xntpd daemon, enter:

```
stopsrc -s xntpd
```

To use the authentication key file /etc/ntp.new.keys when running the xntpd daemon, enter:

```
/usr/sbin/xntpd -k /etc/ntp.new.keys
```

8.5 Streams

AIX Streams is a kernel extension that is loaded during PHASE I of the boot process. AIX V4.2 has enhanced the implementation of Streams to provide for tunable parameters that can be modified after Streams has been loaded by the boot process. The `strload` command is still used to load the Streams framework in the AIX kernel. The `no` command has been modified to allow the user to display or set the values of Streams parameters.

8.5.1 Streams Tunable Parameters

The first `strload` command without any arguments loads the Streams framework into the AIX kernel whereby all parameters will be initialized to their default values. The `strload` command then reads from the /etc/pse_tune.conf file and passes the parameter names and values to the str_init() function. If it is a load-time parameter, it will be set only during this strload. If it is a run-time parameter, it can be set at any time using the `no` command.

8.5.1.1 Load-Time Parameters

The following are the load-time parameters in AIX V4.2:

nstrpush The number of modules that may be pushed onto a single stream. The default value is set to 8.

pseintrstack Maximum number for the interrupt stack size allowed by Streams while running in the offlevel. The default value is set to 0x3000.

8.5.1.2 Run-Time Parameters

The following are the run-time parameters introduced in AIX V4.2.

strmsgsz Maximum number of bytes of information that a single system call can pass to a stream to be placed into the data part of a message. Any write() function exceeding this size will be broken into multiple messages. If strmsgsz is set to zero, then the number of bytes passed to a stream will be set to MAXALLOCSAVE (defined in /usr/include/net/net_globals.h and currently set to 4 times the size of a memory page). The default value is set to 0.

strctlsz Maximum number of bytes of information that a single system call can pass to a stream to be placed into the control part of a message. The default value is set to 1024.

strthresh Maximum number of bytes Streams are normally allowed to allocate. When the threshold is exceeded, users without the appropriate privilege will not be allowed to open Streams, push modules, or write to Streams devices. The threshold only applies to output. Data coming into the system is not affected. The strthresh represents a percentage of thewall and the value is between zero and one hundred inclusively. The default value is set to 85 (85 percent of thewall).

psetimers Maximum number of timers to allocate by Streams. In AIX, the Streams subsystem allocates a certain number of timer structures at initialization time so that the Streams driver or module can register their timeout() function requests. *Reducing the psetimers value below the default value of 20 is not advisable*.

psebufcalls Maximum number of bufcalls to allocate by Streams. In AIX, the Streams subsystem allocates a certain number of bufcall structures at initialization time so that when the allocb() function fails, the user can register their requests for the bufcall() function. *Reducing the psebufcalls value below the default value of 20 is not advisable*.

strturncnt Maximum number of requests handled by current running thread for a module or Elsewhere level Streams synchronization. In AIX V4.1, the Module level synchronization works in such a way that only one thread can run in the module at any time and all other threads which try to acquire the same module will queue their request and leave. After the current running thread completes its work, it will take all the previously queued requests one-by-one and execute them. If there are a large number of requests in the queue, then the current running thread has to serve everyone, and therefore it will always be busy serving others and starve itself. To eliminate this, the current running thread will serve only strturncnt number of threads; after that, a separate kernel thread wakes up and executes all the pending requests. The default value is set to 15.

lowthresh Maximum number of bytes as a percentage of “thewall” that can be allocated using the allocb() function for the BPRI_LO priority. When the total amount of memory allocated by the net_malloc() function reaches this threshold, then the allocb() function request for the BPRI_LO priority will return zero. The lowthresh can be set to any value between zero and one hundred, inclusive. The default value is set to 90 (90% of thewall).

medthresh Maximum number of bytes as a percentage of “thewall” that can be allocated using the allocb() function for the BPRI_MED priority. When the total amount of memory allocated by the net_malloc() function reaches this threshold, then the allocb() function request for the BPRI_MED priority will return zero. The medthresh can be set to any value between zero and one hundred, inclusive. The default value is set to 95 (95 percent of thewall).

Streams in AIX V4.2 maintains full compatibility with AIX V4.1 Streams. Developers however, need to be aware when moving a function from an AIX V4.1 environment to an AIX V4.2 environment that the following parameters are now tunable parameters and can be affected by changes in the /etc/pse_tune.conf file.

These are STRMSGSZ, STRCTLSZ, STRTHRESH, and NSTRPUSH.

Figure 49 is an example of the /etc/pse_tune.conf file.

```
# COMPONENT_NAME: CMDPSE - Streams commands.
#
# ORIGINS: 27 83
#
# (C) COPYRIGHT International Business Machines Corp. 1991, 1994
# All Rights Reserved
# Licensed Materials - Property of IBM
#
# US Government Users Restricted Rights - Use, duplication or
# disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
#

# Streams Tunable Parameters
#
# This file contains Streams tunable parameter values.
# The initial values are same as the default values.
# To change any parameter, modify the parameter value and
# the next system reboot will make it effective.
# To change the run-time parameter, use the "no" command at any time.

strmsgsz        0       # run-time parameter
strctlsz        1024    # run-time parameter
nstrpush        8       # load-time parameter
psetimers       20      # run-time parameter
psebufcalls     20      # run-time parameter
strturncnt      15      # run-time parameter
strthresh       85      # run-time parameter, 85% of "thewall"
lowthresh       90      # run-time parameter, 90% of "thewall"
medthresh       95      # run-time parameter, 95% of "thewall"
pseintrstack    12288   # load-time parameter, (3 * 4096)
```

Figure 49. /etc/pse_tune.conf File

The no command now supports the Streams framework for display, modification, and default setting of Streams run-time parameters. Since the necessary interface was already included in the no command, there was no need to add further commands for Streams Tunable Parameters, and it has not been necessary to add or change flags in the no command.

8.5.2 Streams Performance and Reliability

The performance and reliability of the Streams framework has been enhanced in AIX V4.2 by addressing tioc module performance overheads and memory/timeout handling.

The tioc module was provided in the tty subsystem to facilitate processing of the transparent IOCTLs in the lower modules. Since a module or driver has no user context, it has to request the stream head to perform data transfers between user and kernel environments. The tioc module is responsible for the transfer of this data to or from user space. Merging the tioc module with the stream head has significantly reduced the overhead of calls made to the ioctl() function. This is particularly true in instances where a Streams message is sent, but does not need to be addressed by the ioctl() function.

Memory handling improvements have been made in the following areas:

- Message block allocation failures are now registered by the network statistics command (`netstat`).
- Lock contention when using two buffer caches in Multi-Processor (MP) mode has been removed. This is accomplished by checking if the system is an MP system. If yes, then the message block is always allocated in the net_malloc() function pool. The two buffer caches are only used in Uni-Processor (UP) systems.
- buffcall() function requests are now handled by priority. Previously, calls were listed as first in, first out.

Timeout calls were previously registered in dynamically created structures. If the system was low on memory, allocation of the structure may have failed and the timeout request not registered. This has been resolved in AIX V4.2 by preallocating a number of structures when the Streams framework is loaded.

8.6 TCP/UDP Checksum

The process of computing and verifying TCP/UDP checksums can take up considerable CPU resource, and as a result reduce the overall throughput of a network. However, there are cases where computing and verifying the checksum is not necessarily required, for example, when source and destination are nodes of an SP system, or when the various elements of the network, such as switch fabric, memory subsystems, and I/O bus, are trusted and it is possible to run without TCP or UDP checksum. Another area is in multimedia applications where it would be impractical to request retransmission of corrupt packets.

8.6.1 Option to Disable TCP/UDP Checksum

AIX V4.2 has modified the `ifconfig` command to provide two new options to allow a system administrator to decide if he wishes to eliminate TCP/UDP checksum calculations:

- tcp_nocksum
- -tcp_nocksum

The default option is to leave TCP/UDP checksum enabled.

By disabling the computation and verification of TCP/UDP checksum over reliable networks and interfaces, the throughput on High Performance Switches and other reliable media should be increased, and it should also decrease the amount of CPU needed to move segments within the local network.

The option to disable TCP/UDP checksum does not disable TCP, UDP, and IP header checksum calculations.

8.6.2 Checksum Computation

To compute the checksum, the sending system prepends a pseudo-header to the segment, appends enough bytes containing zero to pad the segment to a multiple of 16 bits, and computes the 16-bit checksum over the entire result. The padded zeros in the segment are not counted, nor are they transmitted. It also assumes the checksum field itself is zero for purposes of computation.

The pseudo-header is used again at the receiving system to verify that the packet has arrived at the correct destination and intact. The structure of the pseudo-header is the same for both UDP and TCP.

Note: The size or makeup of a packet is not altered by disabling checksum computation. It only affects the send and receive events.

Figure 50 shows the format of a standard UDP Datagram/TCP Segment encapsulated in an IP datagram, further encapsulated in a frame for travel across a single network.

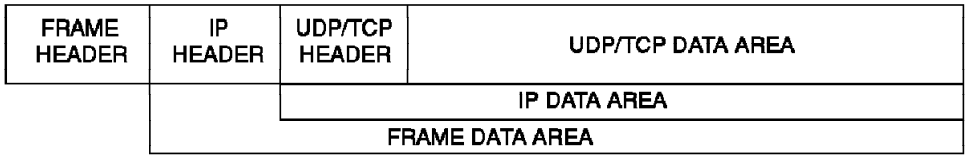


Figure 50. Components of a Network Frame

Figure 51 shows the pseudo-header fields used to calculate the checksum value each time a UDP Datagram/TCP Segment is sent or received on the network.

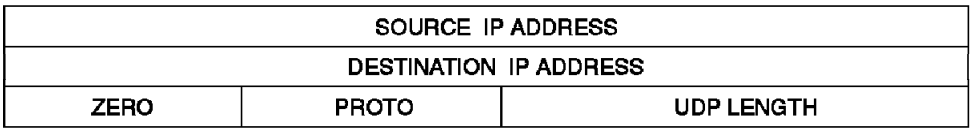


Figure 51. Psuedo-header for Checksum Calculation

8.6.3 Gateways and TCP/UDP Checksum

If a packet is sent across different networks, and the interface has tcp_nocksum set, the software checks against the source IP address and destination IP address. If the addresses are not from within the same network, checksum computation will still take place automatically.

8.7 Soft5080 HostConnect

IBM Soft5080 HostConnect is a new Licensed Program Product (LPP) released along with AIX V4.2 that provides additional connectivity features for the IBM Soft5080 for AIX LPP.

8.7.1 Product Overview

IBM Soft5080 HostConnect emulates functions of the IBM 5088 or 6098 Graphics Channel Controller and runs with AIX Version 4.1.3 or later on a RISC System/6000 connecting to the host through a Block Multiplexer Channel Adapter (BMCA).

IBM Soft5080 HostConnect provides the emulation features provided in IBM Soft5080 for AIX and now also supports non SNA-3270 emulation sessions using TCP/IP over token-ring, Ethernet or fiber networks.

The workstation running the HostConnect software must be connected directly to the host via the BMCA. Other Soft5080 workstations can then be connected to the RS/6000 server via the network. Performance on this type of connection is higher than connections utilizing IBM 5088 or 6098 Channel Controllers.

IBM Soft5080 HostConnect supports up to 63 concurrent TCP/IP connected 5080 or 3270 emulation sessions per Block Multiplexer Channel Adapter. As an example, you could have 20 x 3270 sessions and 43 x Soft5080 sessions or any combination that does not exceed 63 sessions per BMCA.

8.7.2 Product Restrictions

IBM Soft5080 HostConnect has the following restrictions:

- Cannot be used to connect to IBM 5085 or 5086
- Cannot run with Block Multiplexer Channel Adapter Version 1 (5697-037)
- Cannot run with Block Multiplexer Channel Connection for AIX Version 1.1 (5765-604)
- Cannot run with the EMGW program

8.7.3 Software Configuration Overview

The IBM Soft5080 HostConnect LPP consists of the following modules:

Gateway Daemon

Transmits data between the 5080 or 3270 emulation and the host application.

scabd Subsystem (SubChannel Address Broker)

Manages the gateway daemon and the subchannel addresses. It generates the gateway daemon upon request from the 5080 or 3270 emulation, and it connects the 5080 or 3270 emulation with the gateway daemon. It also responds to inquiries from the HAM for status.

HAM (Host Application Mapper)

Monitors the addresses managed by the HostConnect software. It is used when a user obtains an available address, or a system administrator monitors the system usage status.

Device Drivers and Microcode

Provided by IBM Soft5080 HostConnect.

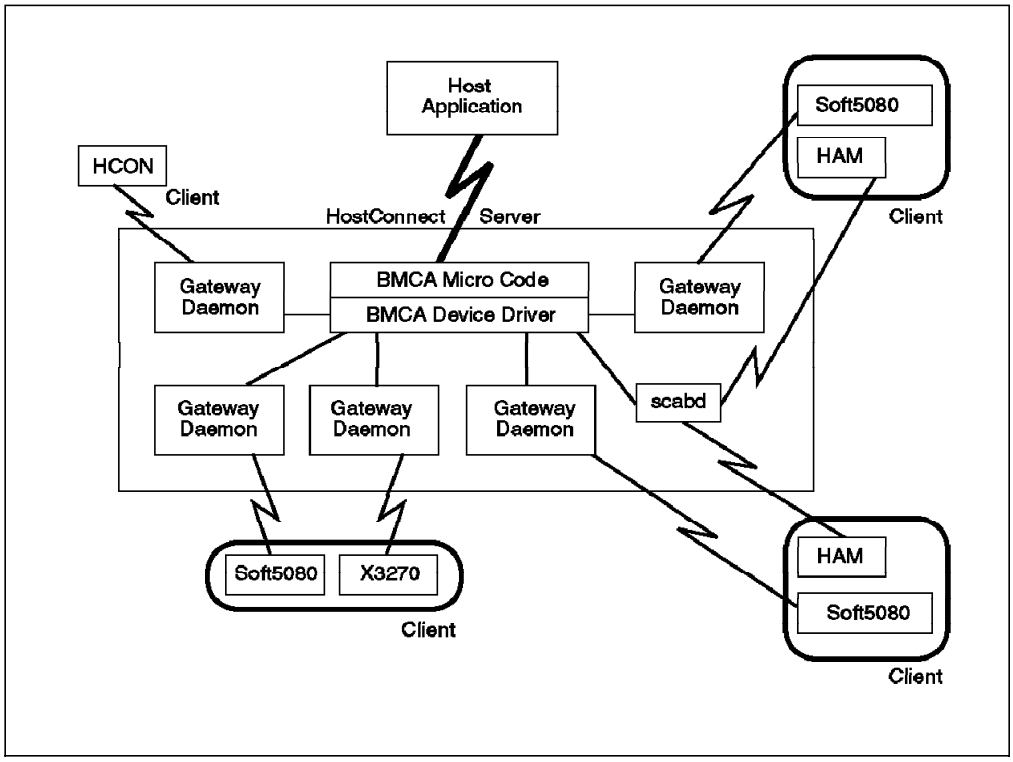


Figure 52. Relationship between HostConnect and Other Programs

8.7.4 Hardware Configuration Overview

The diagram in Figure 53 shows a typical IBM Soft5080 HostConnect topology.

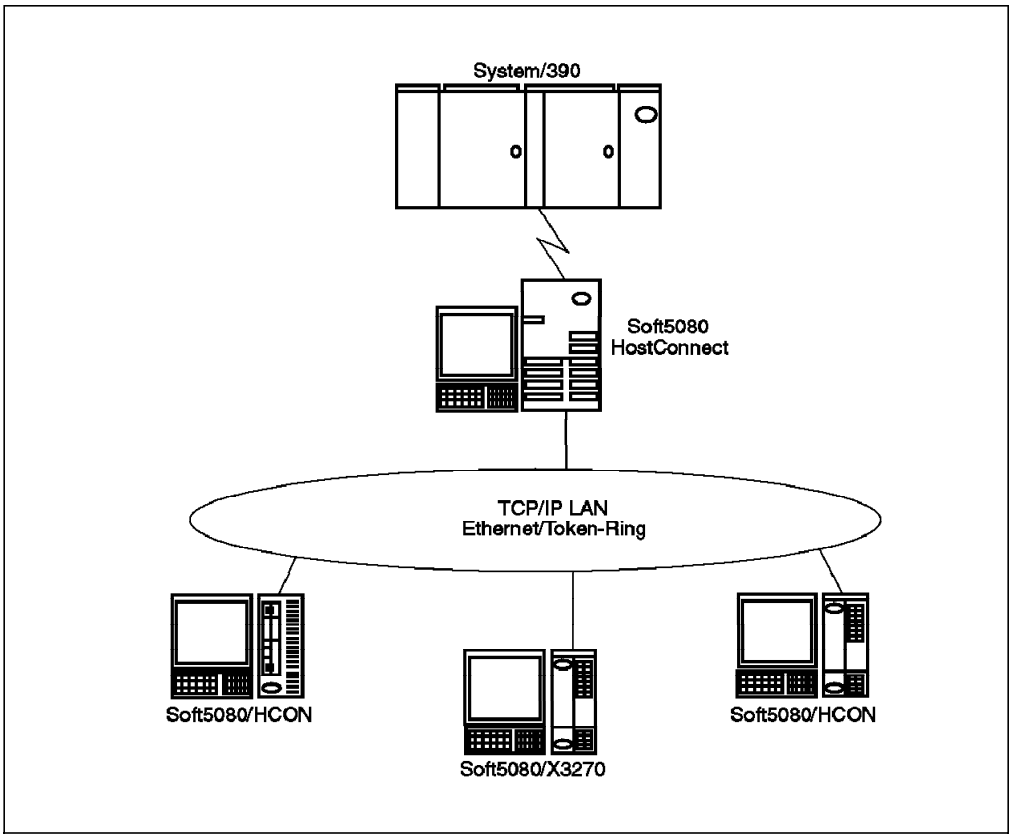


Figure 53. Hardware Topology Using HostConnect

Chapter 9. Performance Enhancements

A number of changes have been made to performance related commands in AIX Version 4.2. Firstly, the stem command is now executable by non-root users. The lockstat command no longer uses the /usr/lpp/perfagent/lockstat/lockname.db file and the lvedit command has been removed. In addition, some minor changes were made to kernel subroutines and locking mechanisms to improve performance under certain conditions.

9.1 Stem Command

The stem (Scanning Tunneling Encapsulating Microscope) command is a tool for inserting debug, monitoring or instrumentation code into subroutines. The instrumentation code may be either user supplied or defaults supplied with stem. It operates on existing libraries and programs without requiring source code or recompilation of the libraries or programs.

stem places the instrumentation code at the entry and exit points of selected subroutines, known as target routines. These target routines can be in user programs or shared libraries. User-defined instrumentation is in the form of subroutines, known as instrumentation routines. Instrumentation routines are simple C subroutines readily created and tailored by you.

In addition to monitoring target routines, the stem command can also be used to replace target routines with instrumentation routines.

9.1.1.1 What's New

For security reasons, previous versions of AIX only allowed the root user to use the stem command. In some environments this causes problems since not all users may be allowed root privileges. With AIX Version 4.2 any user belonging to the newly created *perf* group is allowed to use the stem command.

If *perf* is only in the *group set* of the user and not his primary group, the user must issue the setgroups command to change his real group for this session.

For example:

© Copyright IBM Corp. 1996

```
aix42gra> id
uid=200(vanel) gid=1(staff) groups=20(perf)
aix42gra> stem -p ls
stem: Permission denied.
        To run stem, you have to be root or user
        belonging to the perf group.
aix42gra> setgroups -r perf
aix42gra> id
uid=200(vanel) gid=20(perf) groups=1(staff)
aix42gra> stem -p /bin/ls
*******************************************************
Make.Stem does not exist, issuing make for stem_samples.o
make CFLAGS=-I/usr/samples/perfagent/stem stem_samples.o
        cc -I/usr/samples/perfagent/stem -c stem_samples.c
*******************************************************
The instrumented ls is at /tmp/EXE/ls
aix42gra>
```

9.2 lockstat Command

The lockstat command reports statistics about contention in the operating system among simple and complex kernel locks. Reports generated by the lockstat command can be used to ensure that system performance is not being affected by excessive lock contention.

The lockstat command generates a report for each kernel lock which meets all the specified conditions. If no conditional values are specified, default conditions are used. The reports give information about the number of lock requests for each lock. A lock request is a lock operation (such as taking or upgrading a lock) which in some cases cannot be satisfied immediately. A lock request which cannot be satisfied at once is said to block. A blocked request will either spin (repeatedly execute instructions which do nothing) or sleep (allowing another thread to execute).

9.2.1.1 What's New

In AIX Version 4.1, the symbolic names for the locks are described in two different files:

- /usr/include/sys/lockname.h

  This file is the usual include file used by developers.

- /usr/lpp/perfagent/lockstat/lockname.db

  This file is used by lockstat to correlate the lock identifier with symbolic names.

This raised the possibility of these files being out of synchronization.

In AIX Version 4.2 the /usr/lpp/perfagent/lockstat/lockname.db has been removed and the /usr/include/sys/lockname.h file is used on its own.

In addition, developers who have a requirement for their own locks now have the ability to create their own lockname files in /usr/include/sys. The files must be named lockname_*.h (where '*' can be any string) and lockstat will automatically search for these files.

9.3 lvedit Command

The `lvedit` command has been removed. The role of the `lvedit` command was to customize logical volumes and display the disposition of the physical partitions on the disks.

This functionality exists in a much simpler to use form in `xlvm`.

This an example of the graphical output of `xlvm`.

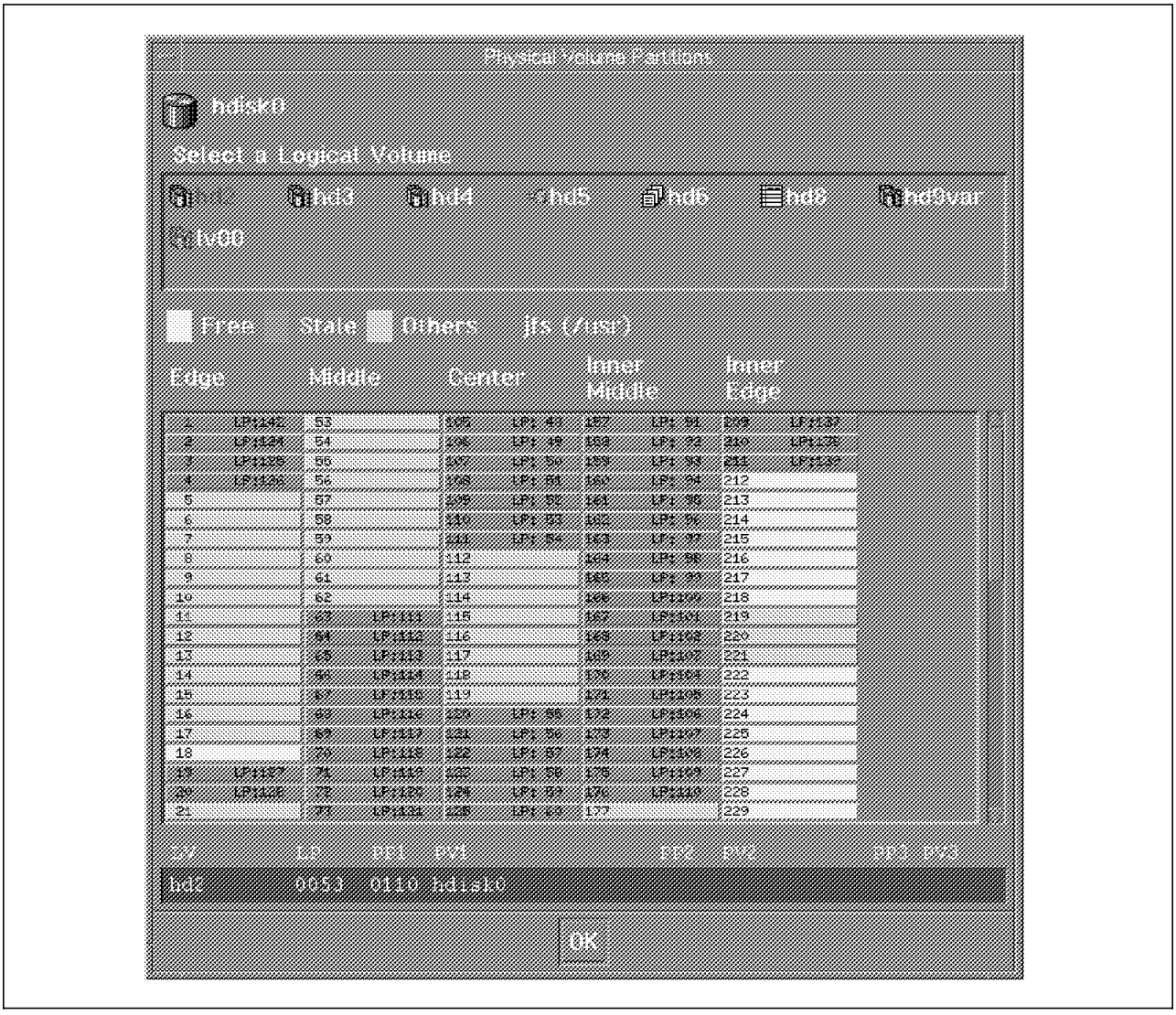


Figure 54. xlvm Window with Physical Volume Partitions

The `lslv` command provides the same information for users who only have access to ASCII terminals.

The following command displays the partitions used by the logical volume /dev/hd2 on disk0 of the system.

```
$lslv -p hdisk0 hd2
hdisk0: hd2 :/usr
USED   USED   USED   USED   USED   USED   USED   USED   USED   USED       1-10
USED   USED   USED   USED   USED   USED   USED   USED   USED   USED      11-20
0076   0077   0078   0079   0080   0081   0082   0083   0084   0085      21-30
0086   0087   0088   USED   USED   USED   USED   USED   USED   USED      31-40
USED                                                                     41-41

USED   USED   USED   USED   USED   USED   USED   USED   USED   USED      42-51
USED   USED   USED   USED   USED   USED   USED   USED   USED   0020      52-61
0021   0022   0023   0024   0025   0026   0027   0028   0029   0030      62-71
0031   0032   0033   0034   0035   0036   0037   0038   USED   USED      72-81
USED                                                                     82-82

0039   USED   0040   0041   0042   USED   USED   USED   USED   USED      83-92
USED   USED   USED   USED   USED   USED   USED   USED   USED   USED      93-102
USED   0043   0044   0045   0046   0047   0048   0049   0050   0051     103-112
0052   0053   0054   0055   USED   USED   USED   USED   USED   USED     113-122

0056   0057   0058   0059   0060   0061   0062   0063   0064   0065     123-132
0066   0067   0068   0069   0070   0071   0072   0073   0074   0075     133-142
USED   USED   USED   USED   USED   USED   USED   USED   USED   USED     143-152
USED   USED   USED   USED   USED   USED   0089   0090   USED   USED     153-162

USED   USED   0091   0092   0093   0094   USED   USED   USED   USED     163-172
USED   USED   USED   0095   0096   0097   0098   0099   0100   0101     173-182
0001   0002   USED   0003   0004   0005   0006   USED   0007   0008     183-192
0009   0010   0011   0012   0013   0014   0015   0016   0017   0018     193-202
0019                                                                     203-203
```

If you wish to specify which physical partitions to use for a particular logical volume you must use the -m Mapfile option to `mklv`.

9.4 Changes to the select/poll Subroutines

The select() and poll() subroutines are based on a system call that allows a single process to wait for the first of any file descriptors in a specified set to become ready. It gives a program the ability to tell the kernel which file descriptors it is interested in and when an event occurs the kernel will inform the program that their status has changed. It is frequently used by server processes that must be capable of maintaining multiple input and output connections concurrently. When calling select(), one of the parameters that must be passed is a bitmask which specifies the file descriptors or message queues that the application is interested in. select() then builds a control block for each descriptor or queue. To allocate memory for the control blocks previous implementations of select() called the xmalloc() service routine for every single control block. The result was a large amount of time spent in this service since some developers would pass the maximum number of descriptors allowed, even though they were only interested in a few.

The implementation of select() in AIX V4.2 now allocates a chunk of memory large enough to hold multiple control blocks, thereby reducing the number of calls to xmalloc() for each select() call. If the amount of memory allocated is still insufficient because of the number of control blocks required then another large chunk is allocated.

The reduced number of calls to xmalloc() greatly improves the time spent in the select() subroutine. (For example, in the case of an application which passes the maximum number of file descriptors, the savings amount to approximately 3000 instructions for each call to select(). Since select() is usually called as part of a loop the total number of instructions saved can be quite appreciable).

9.5 JFS Lock Changes

In the JFS filesystem code the JFS_LOCK was the only lock which existed to protect operations to filesystem data structures (inode freelist, cache list, hash list, dqot free list, locks on inodes, jfsmount structures). This meant that once one operation had started all others had to wait until the first completed even if they were updating totally separate areas.

To eradicate this bottleneck the JFS_LOCK has been replaced by more granular ones. ICACHE_LOCK for operations on inodes, NHASH_LOCK for operations on directory name hash list, IRDWR_LOCK for locks on inodes and QUOTA_LOCK for operations related to quotas.

9.5.1 Redesign of Process Table Locking Mechanism

The process table is the kernel data structure where all information about processes is stored. Similar to the JFS data structures mentioned above, this table was protected by a global lock which prevented it from corruption caused by multiple simultaneous updates. The new implementation locks the table on a per-process basis allowing several process entries to be updated simultaneously. Of course, if one thread of a process is being modified the whole process is locked but other processes can, for example, receive signals.

9.5.2 Streams Improvements

The performance and reliability of the Streams framework has been enhanced in AIX Version 4.2 by addressing tioc module performance overheads, and memory/timeout handling.

Merging the tioc module with the Stream head has significantly reduced the overhead of calls made to `ioctl`. This is particularly true in instances where a streams message is sent, but does not need to be addressed by the `ioctl`.

Memory handling improvements have been made in the following areas:

- Message block allocation failures are now registered by the network statistics command (`netstat`).
- Lock contention when using two buffer caches in MP mode has been removed. This is accomplished by checking if the system is MP. If yes, then the message block is always allocated in the `net_malloc` pool. The two buffer caches are only used in UP systems. All `buffcall` requests are now handled by priority, previously, calls were listed first in, first out. Timeout calls were previously registered in dynamically created structures. If the system was low on memory, it was possible that the structure may not have been allocated and the timeout request not registered. This has been resolved in AIX V4.2 by preallocating a number of structures when the Streams framework is loaded.

```
Sample /etc/pse_tune.conf
# Streams Tunable Parameters
#
# This file contains Streams tunable parameter values.
# The initial values are same as the default values.
# To change any parameter, modify the parameter value and
# the next system reboot will make it effective.
# To change the run-time parameter, use the "no" command at any time.

strmsgsz              0          # run-time parameter
strctlsz           1024          # run-time parameter
nstrpush              8          # load-time parameter
psetimers            20          # run-time parameter
psebufcalls          20          # run-time parameter
strturncnt           15          # run-time parameter
strthresh            85          # run-time parameter, 85% of "thewall"
lowthresh            90          # run-time parameter, 90% of "thewall"
medthresh            95          # run-time parameter, 95% of "thewall"
pseintrstack      12288          # load-time parameter, (3 * 4096)
```

9.6 604 Specific Memory and String Subroutines

The new PowerPC processors (603 and 604) behave somewhat differently than the earlier POWER processors (and the 601) in terms of copying data from one memory location to another. This means that the code that was written for the POWER processors does not work as efficiently on the 603 and 604 processors.

Therefore, some routines in libc.a have been rewritten to detect the processor type and select the most efficient code for that processor. bcopy and memmove are examples of such subroutines.

The end result is that relinked applications should run faster than previously on 603 or 604 processor based machines.

9.7 Use of Hardware Capability to Retrieve Exception Environment

The PowerPC processor architecture includes trace exceptions and, beginning with the 604 processor, an instruction address breakpoint exception and Performance Monitor actions and exceptions. There is therefore, now a more efficient way for the processor to catch exceptions and provide the address of the routine it was about to execute.

Use of this hardware capability of the 604 allows the profiling tools, tpof for example, to provide the same service as before but consume far less resources.

Chapter 10. License Use Management Runtime for AIX

SystemView License Use Management (LUM) is a new License Management Tool developed by IBM Rome Networking Labs. It is intended to be the new IBM standard on OS/2 and AIX platforms to help customers keep track of licenses in use in a distributed environment, and Software vendors to control the distribution of their products.

License Use Management is an implementation of Gradient's iFOR/LS for AIX/6000. It retains all the previous features of iFOR/LS and improves ease of use and configuration for the end user.

It uses two types of licensing models: nodelocked licensing and server-based licensing. A product can be license-enabled using one or both licensing models.

The old iFOR/LS product developed by Gradient Technology under HP license is still the AIX default license manager, but License Use Management will become the default tool as soon as new versions of current iFOR/LS based products become available.

Gradient iFOR/LS and Net/LS enabled applications are totally compatible with LUM.

LUM is provided with new GUI's that make the installation and configuration of both servers and clients easier and more user friendly. New shell script based command line interfaces are also supplied to satisfy the requirements of environments that do not support X/Window terminals and cover the entire set of operation performed by GUI's.

LUM still relies on NCS to provide binding features. It can be installed in existing NCS cells and coexist with other NCS served applications.

A new feature of LUM is the ability to perform direct binding between servers and clients without using the Global Location Broker daemon to advertise services. In this case license clients can be set up without installing bos.net.ncs as a prerequisite. Such a configuration is recommended for small network configurations.

10.1 Packaging

There are three packages that make up the complete LUM offering, as explained below.

10.1.1.1 License Use Toolkit (LUT)

LUT is used by programmers to develop products capable of managing their license distribution through License Use Runtime. It is not part of the runtime package and is sold separately as Systemview License Use Management Application Developer's Toolkit for AIX. It consists of an implementation of Gradient's Application Developer's Kit (ADK).

© Copyright IBM Corp. 1996

10.1.1.2 Server Runtime Kit (SRK)

The Server Runtime Kit must be installed on one or more workstations in the network. These workstations contain the license information and control the execution of the license-enabled products. This component is part of the License Use Runtime product. It is also referred to as License Use Runtime Server.

SRK consists of three filesets:

SRK Base ifor_ls.server.base

SRK GUI ifor_ls.server.gui

SRK Messages ifor_ls.msg.en_US.server

10.1.1.3 Client Runtime Kit (CRK)

The Client Runtime Kit is installed on every workstation where a license-enabled product is to be installed. This component is part of the License Use Runtime product. It is also referred to as License Use Runtime Client.

CRK also consists of three filesets

CRK Base ifor_ls.client.base

CRK GUI ifor_ls.client.gui

CRK Messages ifor_ls.msg.en_US.client

Translated versions of LUM will be provided by the end of 1996.

10.1.1.4 Prerequisites

SRK required Network Computing System (NCS) V.1.5.1 as a prerequisite on severs using both direct and namespace binding. In this last case, even if NCS is not used for location broking, the NCS local location broker daemon must be activated.

CRK does not require NCS as a prerequisite.

10.2 Product Documentation

License Use Management uses Interactive Presentation Facility eXtended (IPF/X) to provide on-line documentation and help texts. There is no documentation provided in InfoExplorer for LUM.

To retrieve the command syntax for LUM commands, run one of the following utilities from the /usr/opt/ifor/ls/doc directory:

```
xview lumcmd <command_name>

man <command_name>
```

To obtain help for LUM utilities messages run:

```
xview lummsg <message_number>
```

from the directory /usr/opt/ifor/ls/doc.

An installation manual is available in IPF/X format in the /usr/opt/ifor/ls/doc directory. To view it, run the following command:

```
xview lumusg
```

xview is the IPF/X browser. IPF/X is included in AIX V4.2 packages as the fileset ipfx.rte and is the official documentation browser for LUM and SystemView on AIX, OS/2 and Windows platforms.

10.3 LUM Licenses

Licensed products developed using IBM′s License Use Toolkit or Gradient′s Application Developer′s Kit can be managed by LUM, and total backward compatibility is guaranteed, allowing customers to migrate to a LUM environment porting NET/LS and iFOR/LS licenses without modification.

A license is the key that gives you the right to run an instance of a LUM or iFOR/LS enabled product.

10.3.1.1 Nodelocked License

A nodelocked license allows the use of a product at the particular node for which the license was created and for as long as the license remains valid. When you purchase a nodelocked license for a product, the vendor associates the license with the unique identifier (target ID) of the system where the license and the product itself will reside.

A license server is not required to install or manage nodelocked licenses. The Nodelock Administration Tool (NAT) on the client permits you to install and manage nodelocked licenses.

10.3.1.2 Dynamic Nodelocked License

Dynamic Nodelocked Licenses are similar to classic Nodelocked Licenses, but they differ in the way that they are installed on the client.

Dynamic nodelocked licenses are installed on the license server using the Basic License Tool (BLT). The first time the end user uses the licensed product on the client, the client workstation requests the dynamic nodelocked license from the server, and, if available, it is granted to the client, where it gets installed. For later uses of the licensed product, the client has its own nodelocked license, and there is no need to request the license from the server.

10.3.1.3 Concurrent Nodelocked License

As a nodelocked license, the concurrent nodelock is bound to a particular node, it can be installed and managed by the Nodelock Administration Tool (NAT) on the client. Furthermore, it allows a limited number of users to run the licensed application simultaneously, and provides a way of controlling the distribution of a product both on a per user and per system basis.

10.3.1.4 Concurrent Access License

A concurrent-access license is a server-based license which, under the direction of the license server on which it is installed, can be temporarily granted to run the product for which the license was created, on a client.

Concurrent Access Licenses allow controlled distribution of products on a per user basis over networks.

When the product is running, that license remains unavailable to other instances of the product. When the application exits the license is returned to the server, where it again becomes available to other users.

Concurrent access licenses allow as many users to run a licensed software application simultaneously as there are valid licenses for the product available from the license servers in your licensing environment.

10.3.1.5 Use-Once License

A use-once license is a server-based license that permits the single use of a particular licensed product within the period for which the license is valid. Use of a use-once license typically begins when the licensed product is started and typically ends when the licensed product stops running.

The precise conditions that constitute the beginning and completion of the use of a use-once license are determined by the product's vendor. Some vendors provide use-once licenses for their products to supplement concurrent-access licenses during times when user demand for those products exceeds the number of available concurrent-access licenses. The vendor designs the product so that when all concurrent-access licenses for the product are in use, a user can request an available use-once license. Many vendors also use use-once licenses to safely distribute promotional or demonstration versions of their software.

10.4 LUM Passwords

Vendors of license-enabled software deliver licenses to use their products in the form of a license password. A license password (or license key) is an encrypted character string that specifies the actual functional characteristics of the license that it contains. This information, determined by the manufacturer, includes the specific number and type of licenses contained in the password, the date when the licenses become active, and the date when the licenses expire.

The vendor includes the password, along with other important information about their product in a file that is sent to you after you purchase the licenses.

For server based licensed products the Basic License Tool (BLT) is used to add passwords and other information to the license database on a particular license server. The Nodelock Administration Tool (NAT) is used to add passwords and other information about the nodelocked licenses on a License Use Runtime client.

Software vendors can create two distinct types of passwords: simple and compound.

10.4.1.1 Simple Password

Vendors of license-enabled software use simple passwords to provide multiple concurrent-use and use-once licenses for vendor-managed use products and to provide nodelocked licenses for nodelock-licensed products.

Simple passwords contain a specific number of licenses with a finite life span determined at or before the product sale.

10.4.1.2 Compound Password

Compound passwords allow the subsequent creation of concurrent access licenses, use-once licenses, and nodelocked (dynamic nodelocked) licenses.

10.5 Security Levels

Software vendors can license their products to the following IBM predefined security levels:

Vendor-Managed Use Control

Customer-Managed Use Control

10.5.1.1 Vendor-Managed Use Control

Vendor-Managed Use products have the highest level of protection provided by the SystemView License Use Management Runtime for AIX. With vendor-managed use products you are technically bound to stay within the limits of the purchase agreement. When you purchase licenses for a vendor-managed use product, the product vendor will ask you to supply the unique identifier (target ID) of each machine where you intend to install the product licenses. For nodelocked licenses this is the identification of the workstation where the enabled product is to be installed; for the other types of licenses this is the license server. You must also supply the license type and the number of licenses you want to make available. The vendor uses this information to create the password that you use to install and activate the licenses that you have purchased.

The passwords are contained in an enrollment certificate, which the vendor gives you with the product. The password, tied to the specified workstation, cannot be used on another workstation.

Every time you want to change the terms and conditions of the contract, (the most frequent change being an increase in the number of licences), you have to provide the vendor with similar information for each of the machines on which you intend to install the licenses to get a new password.

The BLT is used to add concurrent access, use-once, or dynamic nodelock licenses for vendor-managed use products to the license server that you specify.

The Nodelock Administration Tool is used to add nodelocked and concurrent nodelocked licenses for vendor-managed use products on the machine where the product runs.

10.5.1.2 Customer-Managed Use Products

To provide vendors with greater flexibility in the way they deliver licensed software, license use runtime supports customer-managed use product security level. With this level, licenses are not directly associated by the vendor with a particular license server (or group of license servers). The vendor does not set an upper limit on the number of licenses that you are entitled to use. The License Use Runtime provides you with information on the usage of the enabled products, so helping you to stay within the boundaries of the signed contract.

All license usage events are recorded in the server log.

Vendors license customer-managed use products by means of concurrent-access licenses or use-once licenses, or both.

Vendors typically ship a customer-managed use product with a compound license password that is used to install and distribute licenses to use the product. Each compound password is contained within an enrollment certificate.

Again, the BLT is used to install passwords for customer-managed use products and distribute the licenses they contain among your license servers.

The compound password contains the total number of licenses you can extract (which can be virtually unlimited) and has an expiration date. Both the number of licenses and the expiration date are determined by the vendor when creating the compound password. The duration of the licenses that you derive and distribute from a compound password cannot be longer than the length of time remaining before the compound password expires.

Licenses can be extracted from a compound password at any time before the expiration date of the compound password itself.

Vendors of customer-managed use products may reserve the right to examine the License Use Runtime log files on your servers to determine whether or not the terms of your existing purchase agreement need to be amended.

10.6 Hardstop and Softstop on License Request

When a LUM client application is started and asks for a license from the license server, if no more licenses are available three different behaviors are allowed by the LUM Developer′s Kit:

Client application hardstop

Client application softstop

Client application enqueueing

It is up to the application developer to set the appropriate strategy, according to the level of protection they intend to implement.

A hardstop strategy stops the client application starting if there are no available licenses in the server repository. It is the normal solution for high value Vendor-Managed products.

A softstop strategy lets the client application start even if no licenses are available. Interactive messages or event logging can be implemented to record this event. This solution suits low value products, demos, and Customer-Managed products.

Finally, LUM supports Wait Queue mechanisms. In this case, when a user invokes the client product, and there are no concurrent-access licenses currently available, the product asks if the user wants to wait for a license and eventually prompts those licenses in use and the userid′s and hosts owning those keys. If the user responds affirmatively, that user′s name is added to the wait queue on each of the License Use Runtime servers that provide concurrent-access licenses to use the product. User names are added to the wait queues in chronological order. When a license becomes available, it is released to the user. The user is then removed from all of the wait queues and the next user in the queue becomes eligible for the next available license.

10.7 Network Configuration of License Use Management

License Use Management allows two options for network configuration:

Namespace Binding

Direct Binding

Gradient′s iFOR/LS, available with AIX V.4.1 does not allow Direct Binding configurations and requires the setting of an NCS cell even for small and simple networks.

10.7.1 Network Computing System

The Network Computing System (NCS) is a set of tools for heterogeneous distributed computing. License Use Runtime runs on top of the Network Computing Kernel (NCK) of NCS, which is included in the LUM Runtime.

The NCK includes:

Remote Procedure Call (RPC) Runtime Library: The backbone of the network computing system. It provides the calls that enable local programs to execute procedures on remote hosts. These calls transfer requests and responses between clients (the programs calling the procedures), and servers (the programs executing the procedures). The RPC embedded in all license servers and enabled products provides a common mechanism for supporting the request and acquisition of licenses.

Location Broker: A mechanism that allows the enabled products to locate license servers in the network. Location broker software includes the global location broker (GLB), the local location broker (LLB), and administrative tools. The location broker software is included in the License Use Runtime.

LUM user interfaces hide the complexity of NCS configuration. NCK′s components are transparent to the user.

A prerequisite knowledge of NCS is assumed here, therefore, we will not discuss details of NCS components and configuration.

10.7.2 Namespace Binding

LUM still uses NCS V.1.5.1 to manage namespace binding. This means that it retains all the pros and cons of NCS. Remember that NCS will not allow systems to be allocated to more than one cell, it is therefore imperative that the network environment is planned carefully before installing and configuring LUM. Otherwise, it may not be possible for the client to connect to the correct license server. LUM, like any other NCS based application, can share Global Location Broker services with other applications. This allows the LUM environment to be easily added into a preexisting cell; generally, cells represent corporate departments where it is expected that services and products will be shared.

LUM implements the local location broker using the well-known local location broker daemon subsystem (llbd) to manage communications between servers and the global location brokers.

Every node that provides services to the NCS (that is, every license server) must run the llbd subsystem, including systems that run the global location broker service. Several configurations can be established depending upon the number

of servers available. It is also possible to have mixed systems, for interoperability purposes. For example, a license server for OS/2 (i4lmd process) or iFOR/LS AIX server (netlsd subsystem) may already be running in your environment.

LUM implements the global location broker using the global location broker daemon (glbd) subsystem to maintain a database of where all services reside on the network. There can be a global location broker on one or more license servers on the network. At least one node in an NCS cell must run the glbd subsystem.

The global location broker dynamically updates network location information for each license server. If you configure new license servers, or move existing license servers to new locations on the network, licensed applications will always be able to find them. NCS V.1.5.1 supports multiple replicas of the global location broker.

Namespace binding provides a powerful method of administering large client/server networks.

10.7.3 Direct Binding

If you configure your license server with namespace binding disabled, it will not register itself with the global location broker. Therefore, it cannot be found by requesting client applications using standard NCS location broker services. To locate servers configured for direct binding, client applications read a local configuration file with the network addresses of these license servers.

All license servers listen for incoming communications on a well-known port (1515). The client code uses this port number, together with the network addresses of the server systems specified in the local configuration file, to locate and connect to the servers.

Direct binding is a new feature introduced by LUM. Net/LS and iFOR/LS require a full NCS configuration to be set up anytime you intend to use concurrent licenses on the network.

For small environments, with one or two servers, setting up and managing namespace binding can be expensive. Direct binding responds directly to the market need for simple licensing enviroment installation, configuration, and management.

As the licensing enviroment increases, the complexity of managing direct binding increases, and namespace binding becomes the best way to manage the license use management environment.

It is not recommended that a mixed environment of both namespace and direct binding is used. Also, switching from a namespace to a direct binding environment and vice versa is not recommended.

10.7.4 Tips on Network Configuration

Most licensing failures are due to network problems. Ensure that your networking environment is reliably maintained. This is particularly important when programs that follow hardstop policies are used.

Do not configure license servers on unreliable systems, that is, systems that are likely to be be shutdown or rebooted often.

If you have an environment that consists of several subnetworks, ideally, license servers should be on the same subnetwork as the majority of clients that will run the licensed products. Accessing license servers in another subnetwork, across a bridge or router, may not be quite as fast. The subnetworks may also be connected by routers that do not support NCS broadcast. In this case additional configuration will be required on the clients.

10.8 License Use Management Runtime Subsystems

LUM has two new subsystems to manage the licensing system. They are:

Administration Database Server

License Server

10.8.1 Administration Database Server (i4gdb)

The administration database server subsystem provides a mechanism for storing licensing information in a database common to all the servers. It is used for the administration of customer-managed use products. It is called i4gdb. There must be one and only one administration database subsystem running in a cell when using namespace binding, and one and only one running in all the license use management environment when direct or mixed binding is used. That is, only one server node must have the i4gdb started. This ensures the data is accurate and complete.

This subsystem manages the synchronization of various replica Global Location Brokers. It helps to keep consistent images of these vital databases. Basically the task of this daemon is to automatically perform sync operations on the various Global Location databases (/etc/ncs/glb.e) of the cell at configurable time-outs.

It is important to remember that after the administration database is assigned to a particular network node it must always reside on it. This means the administration database server must always be started on the same node.

Also, in order to perform administration tasks with the Basic License Tool (BLT), the administration database must be up and running.

In namespace binding the configuration tool automatically places and starts the administration database subsystem on the server that starts the new global location broker. If more than one administration database is found, the newly started i4gdb subsystem automatically shuts down.

In direct binding you need to decide where to start the i4gdb subsystem using the configuration tool. Make sure that there is no more than one i4gdb in your network licensing environment.

This subsystem is totally new and is not available with Gradient's iFOR/LS.

10.8.2 License Server (i4lmd)

The license server subsystem provides the actual license services, and is called `i4lmd`. A node that works as a license server must run this subsystem.

This important subsystem replaces Gradient's `netlsd` daemon and they cannot coexist on the same server. This does not cause a problem since `i4lmd` transparently serves licenses previously managed by `netlsd` and allows easy and simple migration to the new license manager.

LUM, like Gradient's iFOR/LS, does not allow duplication of servers. That is, you are able to set up multiple servers on the same network, but they must manage different sets of licenses. If a server crashes the license environment loses all the licenses served by that system. Having multiple license servers on the network will however help to avoid the situation where a whole department is idle because the sole license server failed.

10.9 License Use Runtime Installation and Configuration

LUM consists of three main components:

Server Runtime Kit (SRK)

Client Runtime Kit (CRK)

Network Computing System (NCS)

10.9.1 Installation Prerequisites

To install License Use Runtime you must provide the following prerequisites:

6.2 MB of free disk space to install the SRK package

4.1 MB of free disk space to install the CRK package

2 MB of free disk space to install NCS

A system running AIX V.4.2

In addition, both the SRK and CRK base packages require the installation of TCP/IP Base for AIX V4.2 (bos.net.tcp). SRK and CRK GUI packages (ifor_ls.server.gui and ifor_ls.client.gui) require installation of IPF/X Runtime Support (ipfx.base.rte) and Xwindows Runtime (X11.base.rte).

10.9.2 Migration from NetLS or iFOR/LS

At installation time the license database, the nodelock file, and the user file are moved to License Use Runtime directories. The global location broker database and the NCS configuration files are preserved and the `netlsd` subsystem is removed from the system.

The License Use Runtime installation deletes the automatic startup of old subsystems.

All the machines where License Use Runtime is installed (CRK or SRK) are by default configured as clients, so no further action is required on client machines. To maintain the same configuration on the server machines, enter the following command:

```
i4cfg -script
```

This command starts the script configuration command, which asks some basic questions. Specify that you want to:

Configure your system as server and client

Use NCS namespace and direct binding

Keep your current NCS configuration

Leave the license server configuration unchanged

Skip the client configuration

Save the configuration

Have the subsystems start up automatically

Remember to update your PATH variable to include the LUM executables directory. Shell scripts used by LUM runtime all refer to files without their fully qualified pathnames.

A migration from NetLS or iFOR/LS is transparent and does not affect the usability of applications enabled for the previous license managers. However, be aware of possible impacts of the migration on customer-implemented shell scripts that could still be referencing the old daemons or file system locations for executables or database files. These problems can be easily fixed by using symbolic links pointing to the new directories or updating the names of the executables.

Note that you cannot support concurrent NetLS or iFOR/LS licenses using LUM direct binding mode, only NCS namespace binding is supported for those licenses.

10.9.3 License Use Management Configuration

The configuration of a LUM environment can be split into two steps:

1. License Server Configuration
2. License Client Configuration

A license server configuration is performed by customizing the Server Runtime Kit. SRK can be configured either as a server and client or as a client only.

The configuration of SRK is mainly related to the type of binding being used in the cell.

NCS namespace binding mode

Direct binding mode

Both binding modes

If a machine is configured as a license server, server based licensed products will run immediately. It will also work as a client without any further configuration.

If you have installed the server on a machine that was already working as a client only, be sure to configure the new server as a replica of the server from where the client receives the licenses. This will allow the license enabled products on the client to continue to work.

Configuring the License Use Runtime SRK with NCS namespace binding disabled, you do not need to configure the NCS subsystems on the server. The server will be available only for clients configured to work in direct binding mode.

10.9.3.1 Configuration of License Use Management Server by i4cfg Tool

SRK can be configured using an interactive graphic tool called i4cfg. Before starting this tool, stop the server subsystems with the command:

```
i4cfg -stop
```

Start the tool with the command `i4cfg` from an X Window session. You must run this command with root user authority.

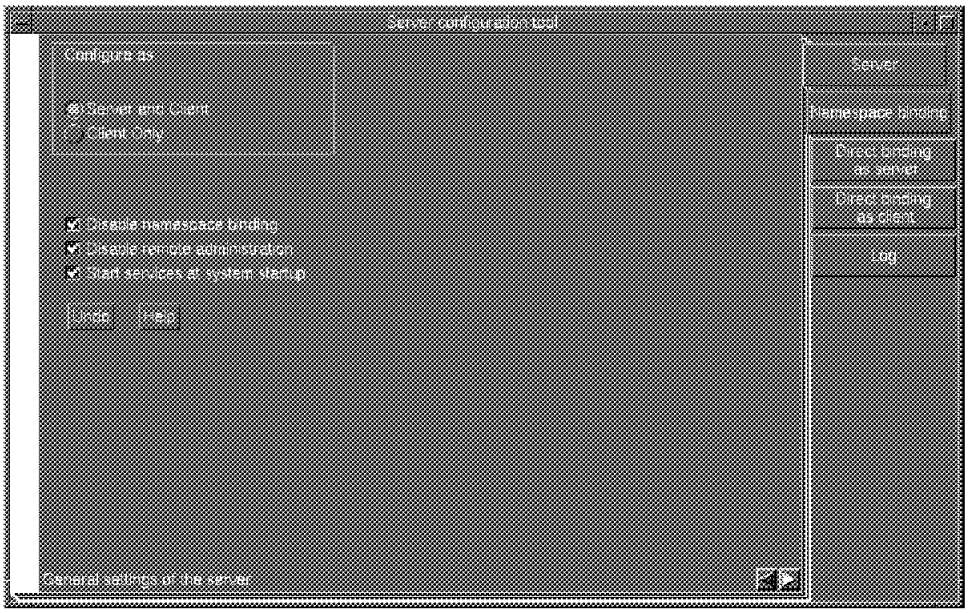


Figure 55. i4cfg Server General Configuration Screen

On the first page of the Server Configuration Tool select the `Server and Client` option to configure your machine as both server and client.

Select the `Disable namespace binding` check box to configure your server to support only clients configured to work in direct binding mode. If you do not select it, support for both namespace binding and direct binding mode will be configured.

Select the `Disable remote administration` check box to disable remote administration on this server. Checking this box means that remote users cannot manage the licenses on this server by running the Basic License Tool on another server.

Select the `Start services at system startup` check box to make the License Use Runtime and NCS subsystems automatically start when the machine is booted.

For a detailed explanation of each field, use the online help.

To configure namespace binding mode for both server and client on this machine, select the Namespace binding tab at the right side of the notebook. The Namespace binding section is displayed.

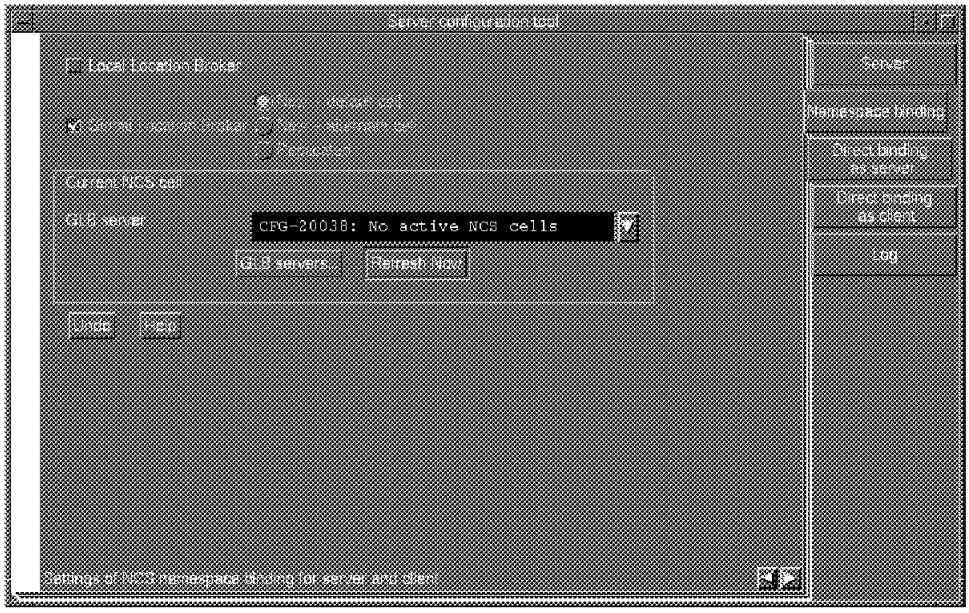


Figure 56. i4cfg Server Namespace Binding Configuration Panel

If you did not select Disable namespace binding in the previous section, the Local Location Broker check box will be automatically checked and you will not be able to change it. If you selected Disable namespace binding, the Local Location Broker check box is enabled. Checking this box allows the direct binding clients to query the license server port number dynamically from the local location broker rather than having to specify it directly.

To start the global location broker on the server, select the Global Location Broker check box. Then select one of the following radio buttons:

New - default cell To start a new global location broker process on the current server and place the server in the default cell

New - alternate cell To start a new global location broker process and place the server in an alternate cell

Replicated To replicate the database of an existing global location broker and place the server in the same NCS cell as the server where the existing global location broker is

If you do not want to start the global location broker on your server, do not check the Global Location Broker check box.

If you have not selected the Global Location Broker check box, or if you have selected it with the Replicated check box, or if you are configuring this machine as client only, the current NCS cell controls are enabled. Use the GLB server field to see a list of servers where the global location broker runs in each existing NCS cell and select one of them.

If more than one server runs the global location broker in a cell, only the server running the most recently started global location broker is displayed. Your workstation is added to the cell of the selected server.

Select the Refresh now push button to see the updated list of GLB servers.

Select the GLBs Servers push button if you want to display detailed information on all the servers running the global location broker in the selected NCS cell.

To configure the direct binding settings of the server select the Direct binding as server tab at the right side of the notebook. The Direct binding as server page is displayed

Figure 57. i4cfg Server Direct Binding Configuration Screen

Use this section to specify the port number information used by the server for communication with the clients.

The Start administration server check box is enabled only if the server is configured to work with namespace binding disabled. If you check it, the administration database server starts on this machine. If, in your licensing environment you also have servers configured for namespace binding, do not select this check box on any server since an administration database server starts automatically with the first GLB of the NCS cell.

Once the administration database server has been started on a particular machine it should not be moved to another machine.

To connect the client component of this machine to the license servers in direct binding mode, select the Direct binding as client tab at the right side of the notebook.

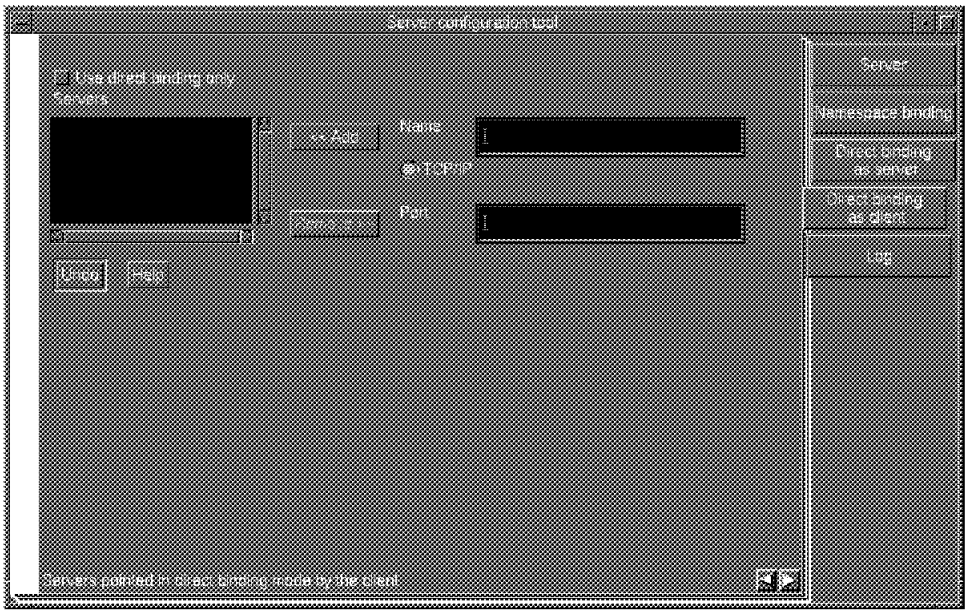


Figure 58. i4cfg Client Direct Binding Screen

Check the Use direct binding only check box if you want the client component of this machine to use direct binding only when connecting to the license servers.

The Servers field will contain a list of license servers this client can connect to. To add a server to this list, fill in the host Name and Port number of the server. Use the TCP/IP fully-qualified domain name for the hostname if the server is in a different domain. Select the Add push button to place the server in the list of available servers. To remove a server from the list, click on the server and then on the Remove push button.

If you plan to run the Basic License Tool and you have checked Use direct binding only, the Servers field must contain at least two entries:

One with the name and port number of the license server you are configuring

One with the name and port number of the administration database server.

This last entry must also be added if the administration database server is running on the machine you are configuring. Then, add an entry for each additional license server in your network. The default port number for the administration database server is 10999, and for the license server it is 1515.

If the local location broker is active on the server you specified, you do not need to specify the corresponding Port number.

Select the Log tab at the right side of the notebook to move to the next section.

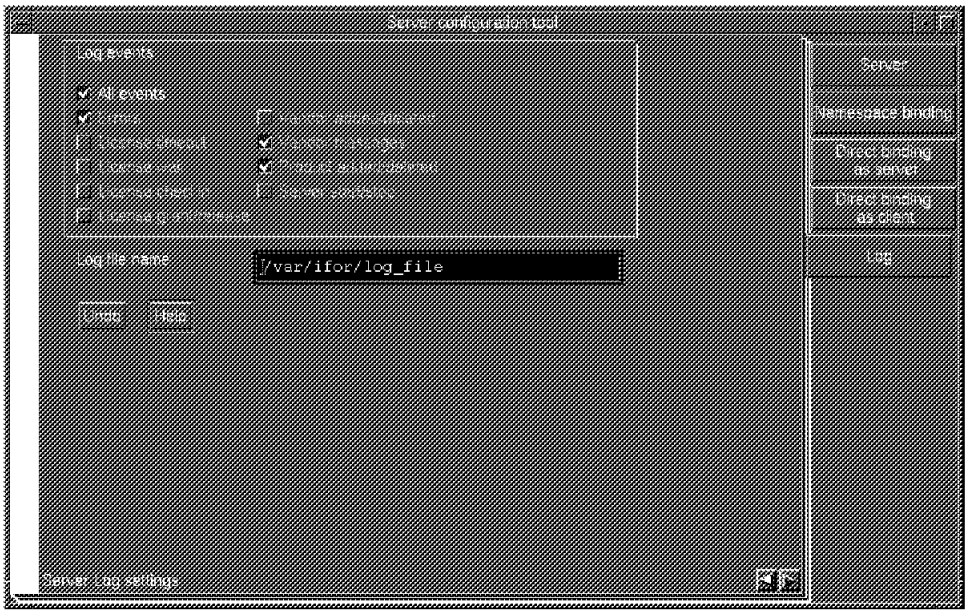


Figure 59. i4cfg Event Log Configuration

Select the check box corresponding to the events you want to log. Use the Log file field to enter the complete path and filename where you want to record them.

To confirm the values you entered, select Close from the system menu at the upper left corner of the window.

Now you can use your machine to install licenses, and grant and monitor their usage.

10.9.3.2 Configuration of License Use Management Server by Script Command

The same configuration can be performed on character-oriented screens using the command:

```
i4cfg -script
```

10.9.3.3 Configuration of License Use Management Client

If only nodelock licensed products are to be used on the system, client configuration does not need to be performed. You can simply install the products.

Before installing use-once, or concurrent access licensed products on the client, the CRK must be configured to specify which mode the client will use to connect to the license servers.

If NCS namespace binding mode is chosen and a default cell exists you do not need to configure the client. A default configuration exists for which the client joins the default cell. In NCS namespace binding mode, at least one server must be running to properly configure the client workstation.

In direct binding mode, only the name and the port number of the servers you want to connect to are needed.

To start the configuration of the client, log in with root authority and run the i4cfg command. The Client Configuration Tool notebook appears.

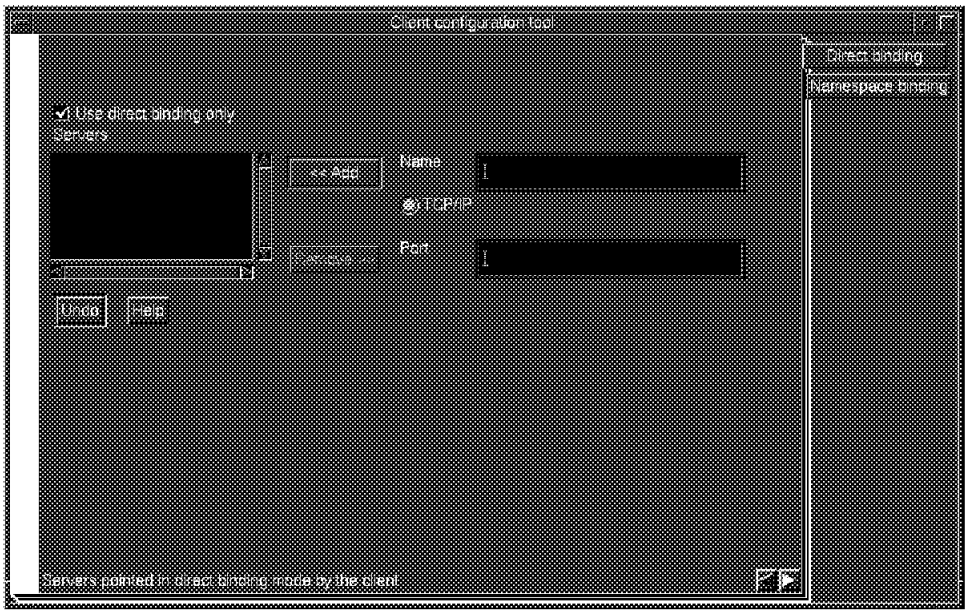


Figure 60. i4cfg Client Direct Binding Configuration Screen

You can connect your client to the license servers in the following ways:

Direct binding: To connect to the license servers using only direct binding mode, check the Use direct binding only checkbox in the Direct binding section of the Client Configuration Tool.

The Servers field displays a list of servers you can connect to. To add a server to this list, fill in the Name and Port number of the server. Then select the Add push button to place the server in the list of available servers. Use the TCP/IP fully-qualified domain name for the hostname if the server is in a different domain. To remove a server from the list, select it, and then select the Remove push button.

If the local location broker is active on the server you specified, you do not need to specify the corresponding Port number.

If you do not check the Use direct binding only check box, your client can also connect to all the license servers registered in the NCS cell specified in the Namespace binding section.

Namespace binding: To configure your client for NCS namespace binding mode, select the Namespace binding tab at the right side of the notebook. The Namespace binding section is displayed.

Figure 61. i4cfg Client Namespace Binding Configuration Screen

Use the GLB server field to obtain a list of servers where the global location broker is running in each existing NCS cell and select one. This will put your client in the cell of the server you selected. If more than one server in a cell is running the global location broker, only one of them is displayed. Select the GLBs Servers push button to display detailed information on all the servers running the global location broker in the selected NCS cell.

Select the Refresh now push button to see the updated list of GLB servers.

To save the values you entered, select Close from the system menu at the upper left corner of the window.

You are now able to use server-based licensed products.

10.9.3.4 LUM Subsystem start and stop

It is important to start and stop the LUM subsystems in the correct sequence. To avoid problems use the following command instead of a sequence of SRC commands.

```
i4cfg -start
```

to start the configured subsystems.

```
i4cfg -stop
```

to stop the configured subsystems.

LUM also provides a command to list the running subsystems.

```
i4cfg -list
```

10.9.4 user File

The user file that was available in Gradient's iFOR/LS is also provided by LUM. It is used to prohibit some users from accessing certain products or to assign different access priorities. It is a flat ASCII file located in the /usr/opt/ifor/ls/conf directory of the license server.

The keywords that are recognized in a user file are:

crawl Causes the priority of users to be changed to the next higher level after the application has polled the queue n times. You can specify crawling only once in a user file. This keyword applies only to products that use queueing.

vendor Specifies a vendor of licensed products. It can be followed by either All, or by the name of a vendor.

allow Specifies that the user names that follow this keyword are allowed to use the product. If no user names follow this keyword, it means users cannot use the product. *allow* and *disallow* are mutually exclusive.

disallow Specifies that the user names that follow this keyword are not allowed to use the product. If no user names follow this keyword, it means all users can use the product. *allow* and *disallow* are mutually exclusive.

-P Specifies the priority of a user. This keyword only applies to products that use queueing. It can assume the following values:

- 1 - Specifies a priority-one user (highest priority)
- 2 - Specifies a priority-two user

Priority-one users may displace lower-priority users already waiting in a queue. The default priority is 3 (lowest priority).

Priority-two users may displace priority-three users already waiting in a queue.

The default priority is 3 (lowest priority).

An example user file is shown in Figure 62 on page 224

```
% This line is a comment
crawl 10
% ******
vendor "Modern Solutions,Inc." "The New Math"
allow fritz -p 2 harry -p 1 monique -p 1 penny
% ******
vendor "Grafix,Inc." all
disallow heather jason
% ******
vendor "Unique Applications,Inc." dynamath
disallow bob
vendor "Unique Applications,Inc." versitex
disallow bob
vendor "Unique Applications,Inc." minigraph
disallow bob
```

Figure 62. LUM Example User File

The crawl 10 statement causes all users listed in the file to move up one level in priority after the product polls the queues of users waiting for licenses 10 times. Using crawl makes sense only when different users have different priorities. The crawl keyword and the user priorities apply only to products that use wait queues.

```
vendor "Modern Solutions,Inc." "The New Math"
allow fritz -p 2 harry -p 1 monique -p 1 penny
```

With the above two lines the user file declares the vendor *Modern Solutions* and its only installed product *The New Math*. The vendor name and the product name contain spaces and are enclosed in quotation marks. Four users can use *The New Math*: Fritz is a priority two user; Harry and Monique are priority one users (the highest); Penny's priority is not specified, so it defaults to three (the lowest).

When a high-priority user requests a license for which low-priority users are already waiting, the high-priority user moves ahead of the other users in the queues. In this case, if Penny and Fritz are unsuccessful in obtaining a license and are willing to wait, crawl eventually changes their priority to one. At that time, the declared priority one users (Harry and Monique) can no longer displace Penny and Fritz in the user queues.

```
vendor "Grafix,Inc." all
disallow heather jason
```

The keyword all means that all licensed products of this vendor (whatever the products may be) have the same user authorizations. In this case everyone except Heather and Jason can use all Graphix software products.

```
vendor "Unique Applications,Inc." dynamath
disallow bob
vendor "Unique Applications,Inc." versitex
disallow bob
vendor "Unique Applications,Inc." minigraph
disallow bob
```

These lines specify three products of Unique Applications Inc.; Dynamath, Versitex, and Minigraph (the product names are not delimited because strings

contain no spaces). Note that the keyword vendor takes exactly one vendor and one product declaration. The keyword disallow excludes only Bob from using the three products of Unique Applications, Inc.. All other users have priority three by default.

This user file specifies the product of Unique Applications individually. This implies that Unique Applications has other installed products that anyone, including Bob, can use. If Dynamath, Versitex and Minigraph were the only installed products of Unique Applications (and Bob was prohibited from using them) the user file would have included the keyword all, instead of specifying each product individually:

```
vendor "Unique Applications, Inc." all
disallow Bob
```

Although you can write different user files for different server nodes, or put user files on some nodes and not on others, such a policy will cause access to a product to vary, depending on which server is granting the license. The same user file should be placed on all server nodes in the interest of a consistent user authorization policy.

When adding a new product or changing user priorities for existing products, remember to update user files at all license server nodes accordingly.

A limit of both iFOR/LS and LUM is that you can't specify a user at a particular node. This means that synonym userids across the network share the same LUM permissions.

10.10 License Installation

Before using a licensed product, it has to be entered into the License Use Runtime database, so that licenses can be granted.

The Nodelock Administration Tool (NAT) is used to manage software products installed locally (on the same system as NAT), and licensed under the License Use Runtime nodelock licensing model.

You can use the tool to add nodelocked licenses to the local nodelock file and to display nodelocked license data.

The Basic License Tool (BLT), is used to manage products whose licenses are installed on the license server. This tool provides an easy-to-use graphical interface that enables you to perform a variety of management and license-monitoring operations on the chosen license server. Tasks include adding products to a license server, distributing customer-managed use products, and generating displays and reports on license usage and server events. By default, the tool displays information for all product licenses from all vendors installed on the license server that you specify. You can easily restrict the view of the tool to gather management information about a subset of products. The Basic License Tool is packaged with the Server Runtime Kit (SRK).

10.10.1 Target Identifier

Before installing the license of a nodelock licensed product, or of a vendor-managed use control product, you must obtain the target ID, the unique identifier of your machine.

To get the target ID from your machine, enter the following command:

```
/usr/opt/ifor/ls/bin/i4target -O
```

Supply your software vendor with the target ID and the operating system on which you are running. Based on the target ID, your software vendor will establish a unique key (password) for your system and will provide you with this key and any other information needed to install the licenses.

10.10.2 Installation of Nodelock Licenses

Log in with root authority and enter the `i4nat` command. The Nodelock Administration Tool window is displayed.

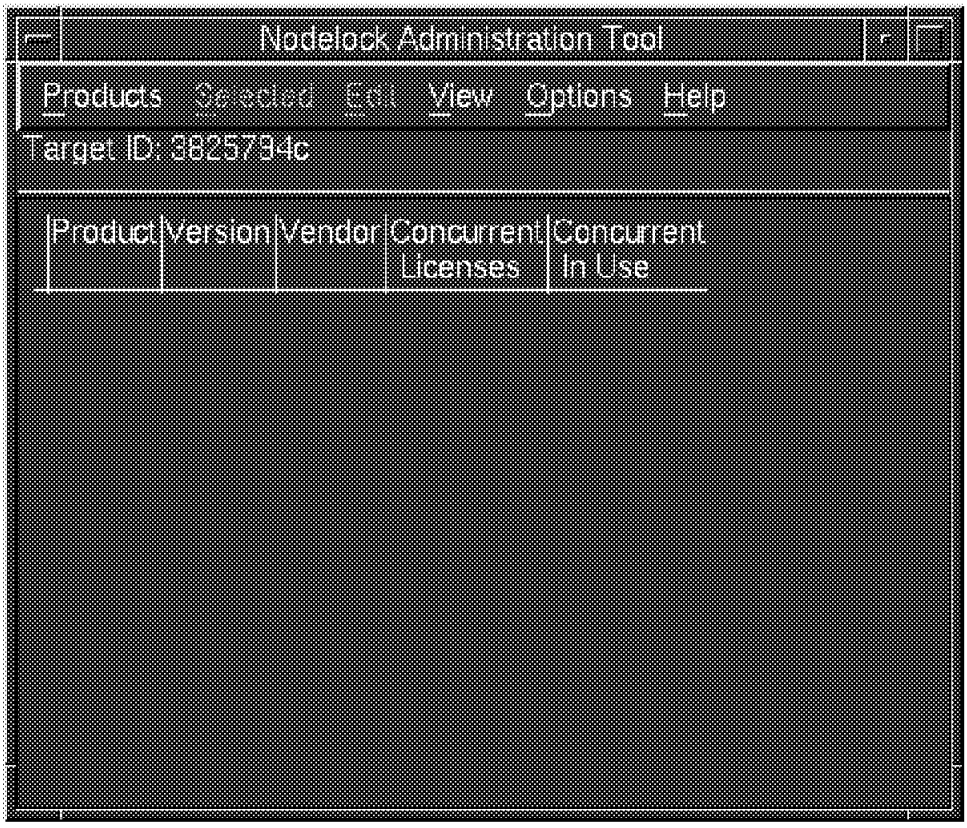


Figure 63. LUM Nodelock Administration Tool (NAT)

Check that the `Target ID` in the upper left corner matches the one you passed to the software vendor.

From the Nodelock Administration Tool window select `New` from the Products pull-down menu.

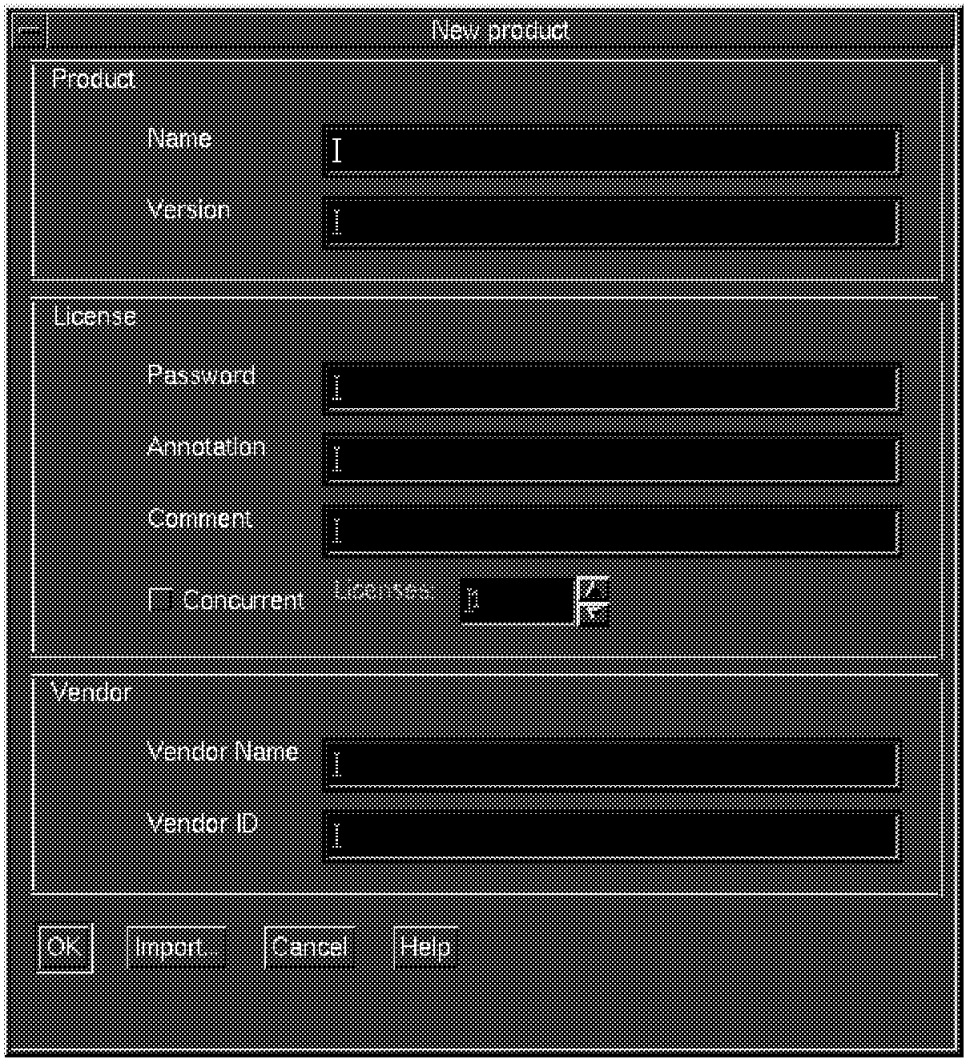


Figure 64. LUM Nodelock Administration Tool Products Window

A license is installed by entering the information required in this window or by importing the same information from an enrollment certificate file.

If the product vendor provided you with an enrollment certificate in the form of an electronic file, select the `Import` push button.

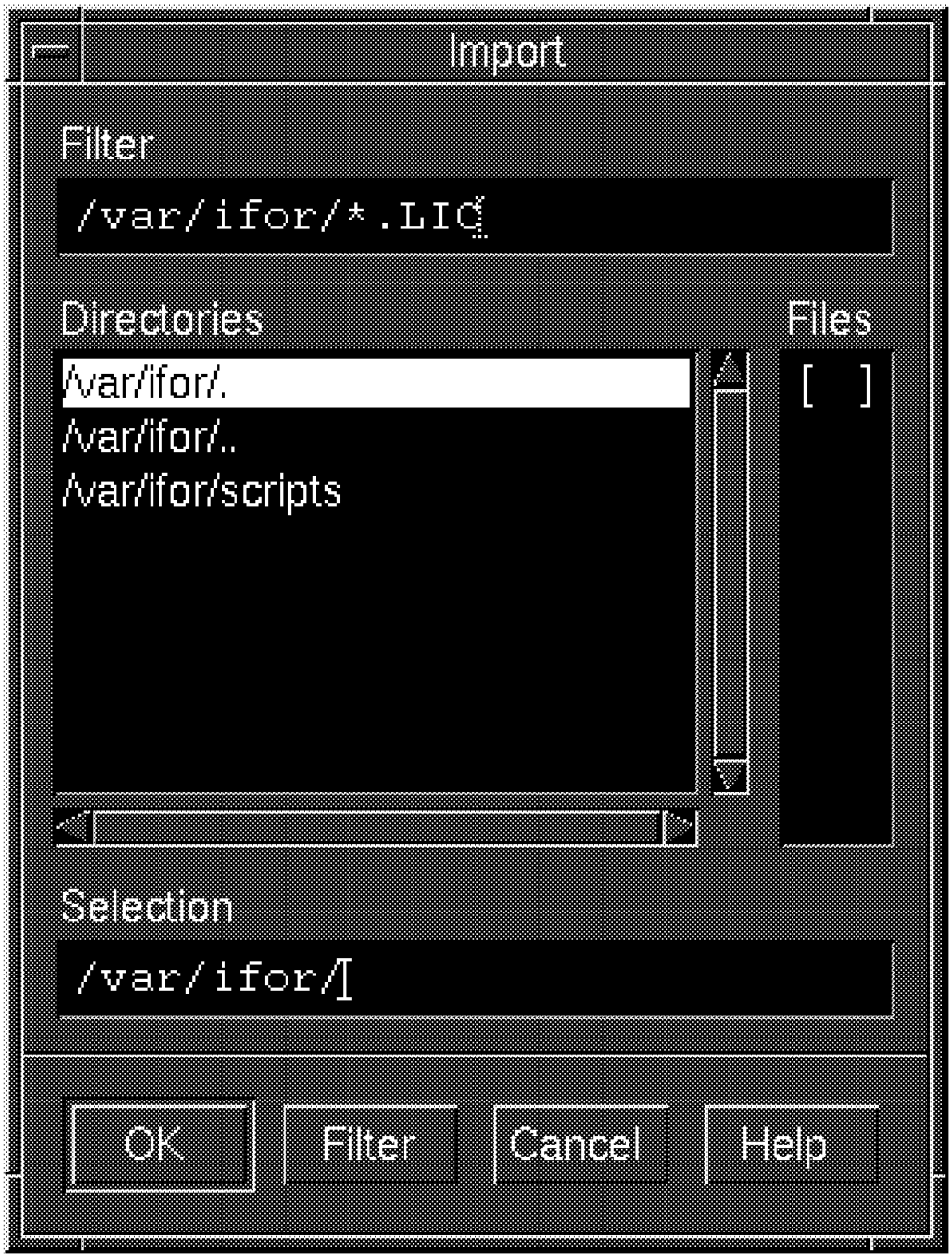


Figure 65. LUM Nodelock Administration Tool Import Window

If the enrollment certificate is on paper, at the New product window fill in the Product, License, and the Vendor input fields as explained in the online help.

Note that the Annotation field is an optional string field that modifies the use of a license in a way defined by the vendor of the software product. If the vendor does not send annotation information, simply leave it blank.

Select the Concurrent check box only if you intend to install a concurrent nodelocked license.

10.10.2.1 Update of Concurrent Nodelocked Licenses

When you purchase the right to add more users for a concurrent nodelock licensed product, you have to update the total number of available licenses.

On the Nodelock Administration Tool window select the concurrent nodelocked license you wish to update.

Choose `Update concurrent license` from the Selected pull-down menu and when the Update concurrent licenses window appears enter the total number of users that can now use the product in the `New total licenses` field.

10.10.2.2 Deletion of Nodelocked Licenses

On the Nodelock Administration Tool select the product whose license you intend to delete.

Select `Remove` from the Selected pull-down menu. The license is now removed.

10.10.2.3 Command Line Interface - i4nat

LUM provides a command line interface to manage all the operations described above. Refer to LUM documentation for a description of `i4nat` command line version parameters.

10.10.3 Server Based Licenses Management

The Basic License Tool's area of responsibility for administration of namespace binding is restricted to the products on servers residing in the same NCS cell as the server from which BLT was started. In the case of direct binding it is limited to the list of servers configured in the Direct Binding as Client section of the server from which you start the tool.

When starting the Basic License Tool, remember that the disable remote administration flag on the server configuration may restrict the list of servers on which you can perform remote administration tasks.

10.10.3.1 Basic License Tool Usage

Log in with root authority and issue the `i4blt` command. The Basic license Tool window is displayed.

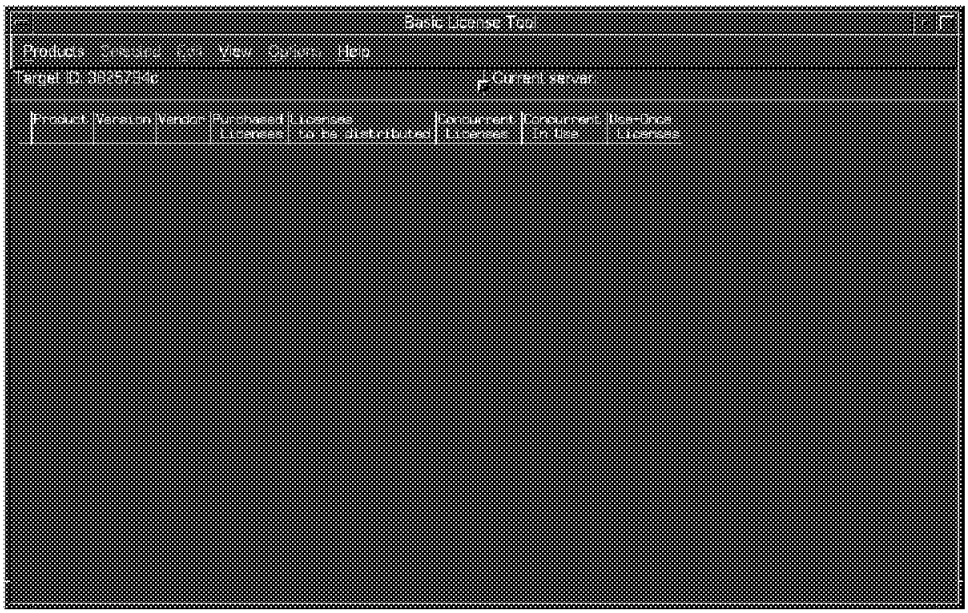


Figure 66. i4blt Basic License Tool Screen

Check that the Target ID in the upper left corner is correct.

The window also contains the list of products whose licenses have been installed. The first time you start the Basic License Tool it only contains the iFOR/LS test product, a sample licensed product.

With the Basic License Tool you can work with only one server at a time. The list of products you see on the Basic License Tool window belongs only to the current server. You can see the current server name or address on the title bar of all the Basic License Tool windows.

To work with a different server from the one displayed, select Switch server from the View pull-down menu in the Basic License Tool window. The Switch server window is displayed.

Select the down arrow to the right of the Current server field. This will list all the active servers in your environment. Select the server you want to work with, and then the OK push button. You can now work with the new current server.

10.10.3.2 Enrollment of a Vendor-Managed Use Product

Perform the following steps to install a license for a vendor-managed use product on the current server, whose name is displayed on the title bar of the Basic License Tool window.

On the Basic License Tool window select Enroll product from the Products pull-down menu. When the enroll product window is displayed enter the information required on this window, or import the same information from the enrollment certificate file.

If the product vendor provided you with an enrollment certificate in the form of an electronic file, select the Import push button.

If you have the enrollment certificate on paper, in the Enroll product window fill in the Product, License, and Vendor input fields as explained in the online help.

Note that the Annotation field is an optional string field that modifies the use of a license in a way defined by the vendor of the software product. If the vendor does not send annotation information, simply leave it blank.

10.10.3.3 Installation of Additional Licenses of a Vendor-Managed Use Product

On the Basic License Tool window select the product for which you intend to install new licenses.

Select Add license password from the Selected pull-down menu. When the Add license password window appears fill in the License input fields or import the same information from the enrollment certificate file.

10.10.3.4 Enrollment of a Customer-Managed Use Product

It is possible to install a compound password containing concurrent or use-once licenses for a new customer-managed use product on the license servers in your network.

You cannot enroll the same customer-managed use product on more than one server in your licensing environment network.

Perform the following steps to install a license for a customer-managed use product on the current server.

Follow the steps described in 10.10.3.2, “Enrollment of a Vendor-Managed Use Product” on page 230 and then continue with the following steps.

When the **OK** push button on the Enroll product window is selected, the Enroll licenses window will be displayed. Fill in the Customer information as described in the online help. In the Purchased licenses field enter the number of licenses you purchased. It must be smaller than the license count included in the enrollment certificate. Then select the **OK** push button.

Before being able to use the product the compound password licenses must be distributed.

Distributing licenses to one or more servers means extracting licenses from the single compound license that has been installed, and making them available on the servers.

On the Basic License Tool window select the installed product whose licenses you want to distribute then select Distribute licenses from the Selected pull-down menu.

On the Distribute licenses window select the servers to which you want to distribute the licenses and click with the right mouse button on the selected server. A pop-up menu appears.

Select the Set licenses option to display the Set licenses window.

Enter the number of licenses you want to distribute in the Number of licenses field, and select the **OK** push button.

To confirm data and distribute the licenses, select **Close** from the system menu of the Distribute licenses window. If everything is OK a confirmation message will be displayed.

10.10.3.5 Update of Licenses of Customer-Managed Use Products

When you purchase the right to use more licenses for customer-managed use control products, you have to change the total number of purchased licenses in the Basic License Tool. To change the number, select the installed product whose purchased licenses you want to change on the Basic License Tool window. Then select Update licenses from the Selected pull-down menu. The Update licenses window appears. In the New purchased licenses field enter the total number of licenses you now own, that is, the existing number plus the newly purchased and select the OK push button.

10.10.3.6 Setting the License Usage Threshold

For all the products you see on the Basic License Tool window, you can specify a usage threshold. When use of the product reaches this threshold a message specifying the percentage of licenses in use for that product is logged. You can see the message by requesting a Vendor messages report through the Basic License Tool. You can also see the message in the AIX window from which the BLT was started.

To set the license threshold, select Thresholds from the Options pull-down menu. The Thresholds window appears.

On the Thresholds window enter the appropriate information as explained in the online help and then select the OK push button.

10.10.3.7 Displaying Product Licenses

To display product licenses, select the product whose licenses you intend to display from the Basic License Tool window. Select Open as Details from the Selected pull-down menu. When the Details-Product page is displayed select the tab corresponding to the licenses you want to display (Compound, Concurrent, or Use-once). Note, you may not necessarily have all three tabs. This depends on the actual licenses installed.

10.10.3.8 Deleting Product Licenses

To delete expired or unwanted product licenses, scroll through the list of product licenses on the Details-Concurrent window to locate the license that you want to delete.

Select the license, and click on it with mouse button two. A pop-up menu is displayed. Select Remove license. A pop-up window is displayed asking you to confirm the deletion.

Note: You cannot delete compound and use-once license passwords until they have expired.

10.10.3.9 Command Line Interface - i4blt

LUM provides a command line interface to manage all the operations described above. Refer to LUM documentation for a description of i4blt parameters.

10.10.4 Reporting Events

Using the Basic License Tool you can display event reports based on a set of event filters. Each filter allows you to display a different category of events. Moreover, you can display different types of events, some of which allow you to get statistical information. You can also redirect the display to a file and print it. For every report you can specify a starting and ending date for the data that is to be reported.

Refer to LUM Documentation for a full description of this logging feature.

Chapter 11. National Language Support (NLS)

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 the messages. The NLS environment is defined by a combination of language and geographic or cultural requirements. These conventions consist of four basic components:

1. Translated language of the screens, panels and messages
2. Language convention of the geographical area and culture
3. Language of the keyboard
4. 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.

In order 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

This feature of AIX V.4 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 the 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

To assist in classifying the extent to which a separately purchasable AIX licensed program provides National Language Support the following levels of National Language Support Enablement may be found in the program product announcement material. They are listed in decending order of national language capability.

Universal Language Support (ULS)
Support for multiple languages based on universal character set ISO 10646

Full International Language Support (ILS)
Support for all locales in the underlying operating system

Multi-Byte Character Set Support (MBCS)
Support for locales based on multi-byte and single-byte code sets in the underlying operating system. Bidirectional code set support is limited.

© Copyright IBM Corp. 1996

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/commands do not support full ILS:

- Adobe Acrobat reader - Supports PASSTHRU only graphics, tplot, graph, spline commands
- Sun's Java Programming Environment - Supports PASSTHRU only
- Netscape Navigator(TM) - Supports PASSTHRU only
- NFS - Supports PASSTHRU only
- NCS - Supports PASSTHRU only
- TCP/IP telnet command - Does not support NLS

11.3 National Languages Enhancements

The following enhancements have been made to National Language support in AIX Version 4.2.

11.3.1.1 Baltic Rim Languages

AIX V.4.2 has introduced support for three new Baltic Rim languages:

- Estonian ET_EE.UTF-8 and Et_EE.IBM-922
- Latvian LV_LV.UTF-8 and Lv_LV-IBM-921
- Lithuanian LT_LT.UTF-8 and Lt_LT.IBM-921

11.3.1.2 Albanian Locale

An additional locale for Albanian language has been added:

- Albanian sq_AL, ISO8859-1

11.3.1.3 Traditional Chinese Code Page 950

AIX V4.2 also supports the new Traditional Chinese Code Page 950 (Big5) which consists of 13056 characters, 1004 symbols, 684 symbols in CNS 11643.192 and 325 IBM Unique symbols.

11.3.1.4 iconv Tables

Two more iconv tables have been added. These tables are used for mapping between PC ASCII CP 866 and ISO8859-5. CP 866 is the most widely used Russian PC code page and the ability to map between CP 866 and ISO8859-5 is necessary to maintain data integrity in mixed PC and RS/6000 LAN Environments.

11.3.1.5 Program Integrated Information (PII)

PII allow screens, prompts, panels and messages to be translated into different national languages. The translatable PII is kept totally separate from the program executable code which allows new languages or variants to be added very easily. X/Open base message facilities are also incorporated in AIX Version 4 to support:

- Isolation of most PII text from program source code
- Translation of PII into supported languages
- Manipulation of PII catalogs via library routines
- Utility and application access to PII catalogs

Translated PII for AIX Version 4.2 and all AIX Version 4.2 SPO LPs are available.

11.3.1.6 New Keyboards Support

The keyboard mappings for Latin American Spanish, Korean and Taiwan were released with AIX 4.1.3; however, manufacturing preload required special handling and users were unable to customize their keyboards. This issue has been resolved with the release of Version 4.2.

11.4 Translations of GUI′s

AIX V.4.2 provides additional translations for Graphic User Interfaces.

11.4.1.1 Translation of Installation Assistant Helps

The Installation Assistant helps are help volumes that are available to dthelpview, the CDE help viewer.

In AIX V4.2 the Installation Assistant help volumes have been translated into all the languages (except for Polish) that ship with AIX V4.2. Since this is one of the first items a user sees during installation it is not unreasonable to expect that this information be delivered in their native language.

11.4.1.2 Translation of Common Desktop Environment Helps

The translation of CDE helps has been extended to include coverage for the previously missing languages:

- Russian
- Czech
- Catalan
- Polish
- Hungarian

AIX CDE now provides more national language support than most other COSE desktop deliveries.

Chapter 12. AIX Version 4.2 License Program Products

This chapter contains information on Licensed Program Products (LPPs) that are available for AIX Version 4.1 (5765-393) and AIX Version 4.2 (5765-655). Also included are target dates for some LPPs that have not as yet announced general availability dates for AIX Version 4. Exact availability dates will be announced in the future, and these dates are subject to change without notice. LPPs are safe on Symmetric Multiprocessor (SMP) and will run in Serial mode on Scalable POWERparallel (SP) System hardware unless otherwise noted. Please refer to section 12.5, “Footnotes” on page 245 for the definition of footnotes that are used in all sections of this Road Map.

12.1 LPPs based on AIX Version 4.1

The following LPPs are based on AIX Version 4 and are currently available:

LPP Name		Version		Number
AIXwindows Display PostScript		1.1		5696-904
ADSTAR Distributed Stg Manager (ADSM) With HSM (Space Management) Client		2.1.5		5765-564
AFS	(2,16)	3.4,3.4a		From Transarc
ARTIC - Realtime Interface Coprocessor	(14)	4.1		See Footnote
Block Multiplexer Channel Adapter	(1,5,12)	1.1		5697-037
Block Multiplexer Channel Connectivity		1.1		5765-604
BookManager BookServer for AIX	(5)	2.0		83H9467
C for AIX	(20)	3.1		5765-423
C Set ++ for AIX	(20)	3.1		5765-421
CICS for AIX	(4,19)	2.1		5765-553
CICS Internet Gateway or 5692-AIX FC 0595	(4,19)			5765-553
COBOL Set for AIX		1.0	CD	28H2176
			8mm	33H4408
CommonPoint Application System for AIX	(3)	1.1		5765-561
CommonPoint Application Development Toolkit for AIX	(3)	1.1		5765-562
Data Encryption Standard Library Routines		1.1,1.1.2		5765-418
DCE Cell Directory Services for AIX		2.1		5765-534
DCE Enhanced Distributed File System for AIX		2.1		5765-537
Getting Started with DCE for Application Developers		2.1		5765-532
DCE NFS to DFS Authenticating Gateway for AIX		2.1		5765-540

© Copyright IBM Corp. 1996

DCE Security Services for AIX		2.1		5765-533
DCE User Data Masking Encryption Facility for AIX		2.1		5765-538
DirectTalk/6000 V1.6	(18)	1.6		5765-001
Distributed SMIT for AIX	(9)	2.1,2.2		5696-902
Encina Client for AIX	(4,19)	2.1		5765-554
Encina Monitor for AIX	(4,19)	2.1		5765-559
Encina Monitor Suite for AIX	(4,19)	2.1		5697-195
Encina PPC Executive for AIX	(4,19)	2.1		5765-555
Encina PPC Gateway for AIX	(4,19)	2.1		5765-557
Encina Server for AIX	(4,19)	2.1		5765-558
Encina Structured File Server for AIX	(4,19)	2.1		5765-556
ESCON Channel Connectivity		1.1		5765-603
ESSL/6000	(1)	2.2.2		5765-042
XL Fortran for AIX	(1,20)	3.2		5765-176
XL Fortran Runtime Environment for AIX (Different Feature Codes for AIX V3 or V4)	(1,20)	3.2		5765-526
geoGPG for AIX V2	(2)	2.1		5765-040
HACMP (modes 1,2,3 w/serial SSA, & HANFS)	(3)	4.1.1		5696-933
High Performance Parallel Interface (HIPPI) Driver Group	(1,9)	4.1		5765-551
Hypertext Information Base Libraries	(4)	1.1		5696-919
Hypertext Information Base Libraries	(4*)	1.2		5696-919
IconAuthor for UNIX	(15)	6.0		5758-AT8
InfoCrafter		2.1		5696-893
InfoExplorer License Extension		1.1		5696-898
Internet Connection Secure Server Version 1.1 for AIX	(2)	1.1		33H4191
Internet Connection Server for AIX	(2)	1.0		33H4190
LoadLeveler	(2,3a,9,17)	1.2.1		5765-145
Lotus Notes	(16)	R4		From Lotus
MERVA for AIX	(2)	1.1.1		5765-449
Multimedia Server for AIX	(3,18)	1.1		5697-213
NetBIOS and IPX/SPX Support for AIX		2.1		5765-550
NetWare for AIX		3.11B		5697-021
Open Systems Standard Communications (OSSC) (Followon Product to OSI/6000)	(2,21) (12)			MH-88AIX
OpenGL and GL 3.2 Version 4.1.4	(4)	4.1		5696-939
Parallel Environment for AIX	(1,3a,17)	2.1		5765-543
Parallel ESSL for AIX	(1,3a,17)	1.1		5765-422
Parallel System Support Programs with HACWS feature	(1,3a,17)	2.1		5765-529
PVMe for AIX	(1,3a,17)	2.1		5765-544
Performance Aide for AIX	(3)	2.1		5696-899
Performance Toolbox for AIX	(4)	2.1		5696-900
PEX & PHIGS Version 4.1.4 for AIX	(4)	4.1		5696-907
PL/I Set for AIX		1.0	CD	33H1858
		1.0	8mm	33H5425
ProductManager V3.1 for AIX	(2)	3.1		5765-605
Smalltalk Professional for AIX, V3.0	(1)	3.0		62H8140
Smalltalk Professional Server for OS/2, for AIX, and for Windows, V3.0	(1)	3.0		62H8150
SNA Client Access for AIX		1.2		5696-944
AIX SNA Server/6000	(1,12)	2.2		5765-247
AIX SNA Gateway/6000	(1,12)	2.2		5765-261
SNA Server for AIX Version 3.1 including: - SNA Gateway/6000 function - AnyNet (TM): APPC over TCP/IP function - AnyNet: Sockets over SNA	(4,12,19)	3.1		5765-582

SNA Manager/6000	(2)	1.1.3	5765-233
Soft5080 for AIX	(2,4,9)	1.1.1	5765-528
StarWorks for AIX	(2)	2.0	5765-552
also available: 50 Mbps Feature			
UIM/X (AIXwindows Interface Composer)	(20)	2.8,2.8.1	5765-400
Ultimedia Services for AIX	(20,22)	2.1	5696-906
Visualage for Smalltalk,	(1)	3.0	62H8200
Professional for AIX, V3.0			
Visualage for Smalltalk,	(1)	3.0	62H8210
Professional Server for OS/2,			
for AIX, and for Windows, V3.0			
Visualage for Smalltalk,	(1)	3.0	62H8220
Communications/Transactions			
for OS/2, for AIX, and for			
Windows, V3.0			
Visualage for Smalltalk,	(1)	3.0	62H8240
Database for Oracle			
for OS/2, for AIX, and for			
Windows, V3.0			
Visualage for Smalltalk,	(1)	3.0	62H8260
Distributed for OS/2, for AIX,			
and for Windows, V3.0			
Visualage for Smalltalk,	(1)	3.0	62H8270
IMS Connection for OS/2, for AIX,			
and for Windows, V3.0			
X.25/AIXlink		1.1	5696-926
3270 Host Connection Program for AIX		2.1	5765-398
5086 Connectivity Enabler for AIX	(2)	2.0	5765-560

12.2 LPPs based on AIX Version 3.2 that run on AIX Version 4

The following LPPs released for AIX Version 3.2 are also compatible with AIX Version 4 on or before today. Some of these programs may require a program update to achieve compatibility.

LPP Name		Version	Number
AIX Access for DOS Users (AADU)	(2,5,6,10,12)	3.1, 3.2	5696-383
ADSTAR Distributed Storage Manager	(12)	1.2.1	5697-078
(ADSM)			
ADSTAR Distributed Stg Manager (ADSM)		2.1.0	5765-564
With HSM (Space Management) Client			
APL2/6000	(7)	1.2	5765-012
Architecture & Engineering Series	(2,18)	2.3	5696-054
Graphics Application			
ATM Campus Manager 1.1.1 for AIX	(2,5,9,12)	1.1,1.1.1	5871-AAA
		or	8OG3078
			8OG3079
Automatically Programmed Tool for AIX	(2)	2.1	5765-496
BookManager Read/6000	(5)	1.2	5765-086
CallPath Server/6000	(5,7)	1.1.1	5765-266
Support for AIX 4.1 for remote			
connection to switch via SS/2 and			
Support for local NTI SL1 and			
DMS100 switched (UP only)	(1)		
Support for AT&T G3 via ethernet connection			
CICS Systems Manager for AIX	(4,19)	1.1	5765-427
Client Input Output/Sockets	(3,7)	2.1	5648-129
(Support for AIX V4 - MVS Product)			

CMVC/6000	(1)	2.2,2.3		5765-207
Connection Program/400 for Unix	(2,5,7)	3.1		5798-RZB
Connection Program for OS/400 for UNIX Environments	(2,5)	3.6		5798-TBE
Continuous Speech Runtime	(3,5,7)	1.0		79G9710
Continuous Speech Toolkit	(3,5,7)	1.0		79G9709
DataHub for UNIX Operating Systems	(9)	1.1		33H1660
DataJoiner for AIX Version 1.1		1.1		33H5840
DataPropagator Relational Apply	(5,12)	1.2		5765-363
Data Replication Starter Kit		1.2	CD	33H1960
			8mm	33H1962
DB2/6000	(5,7,12)	1.2		20H4519
DB2 Client Support/6000	(5,7,12)	1.2		20H4520
DB2 for AIX Version 2.1 Single-User		2.1		41H2127
DB2 for AIX Version 2.1 Server		2.1		41H2128
DB2 Parallel Edition for AIX		1.1		5765-328
DB2 Software Developer's Kit/6000	(5,7,12)	1.2		20H4522
DB2 SDK for AIX Version 2.1		2.1		41H2138
DDCS/6000	(5,12)	1.2		20H4523
DDCS for AIX Version 2.3 Multi-User Gateway		2.3		41H2133
geoGPG/6000	(2,9,11)	1.1.5		5765-026
HCL-eXceed/W	(5)	3.3		5696-466
AIX InfoCrafter/6000	(5)	1.1, 1.2		5696-108
Initial Graphics Exchange Specification (IGES) Processor for AIX and UNIX		2.2		5765-083
IGES Doctor for AIX and UNIX	(2)	1.1		5765-588
Intelligent Hub Manager 2.1.2 for AIX	(2,5,9,12)	2.1.2		80G3050,1
Intelligent Hub Manager Entry 2.1.2 for AIX	(2,5,9,12)	2.1.2		80G3052,3
Interactive System Productivity Facility (ISPF) for MVS V4.2 (Support for AIX)	(5)	4.2		5655-042
InterMix for AIX	(3)	1.2		5765-223
Internet Connection Secured Network Gateway for AIX Version 2.1 (NetSP)	(3,18)	2.1		33H4290
LAN Management Utilities for AIX	(2,5,9)	1.1.3		5765-264
LAN Network Manager for AIX	(2,3)	1.1		5765-251
Legato Networker	(5,12)	4.0		5765-316
			or	89G1685+
MQSeries for AIX	(1,5,12)	2.2	CD	27H8436
			8mm	27H8430
MQSeries for AIX		2.2.1	CD	33H2267
			8mm	33H2242
MultiView Mascot	(2,12)	3.3.2		5696-925
NetView for AIX	(2,3,7,9)	3.1,3.1.1		5696-731
NetView Entry for AIX	(2,7)	3.1		5696-905
NetView Distribution Manager/6000	(5,7,9,12)	1.1.2		5765-196
NetView Distribution Management Agent/6000	(5,7,9,12)	1.1.2		5765-214
NetView FTP for AIX (Server)	(1,7)	1.1		5765-435
NetView FTP for AIX (Client)	(1,7)	1.1		5765-242
NetView Service Point/6000	(2)	1.2.2		5621-107
Network License System (NetLS) Toolkit	(2)			5696-465
Neural Network Utility for AIX	(2,5)	3.1		5765-353
NovaManage	(2,18)	6.2		5697-050
NovaWorkbench	(2,18)	6.2		5697-052
Numerical Control PostProcessor Generator/6000	(2)	1.1.2		5765-020

Numerical Control PostProcessor Execution Library/6000	(2)	1.1.2	5765-003
Nways Broadband Switch Manager for AIX	(2,3,9)	1.2.3	5765-320
Nways Campus Manager LAN for AIX	(2,3,9)	1.0	31H7060,1
Nways Campus Manager ATM for AIX	(2,3,9)	1.0	31H7063,4
Nways Campus Manager Suite for AIX	(2,3)	1.0	33H8430,1
Nways Campus Manager Remote Monitor for AIX	(2,3,9)	1.0	33H8381,2
Nways Campus Manager Remote Monitor Advance-AIX	(2,3,9)	1.0	33H8383,4
ObjectStore	(2,5)	4.0	5758-DN0+
OpenMail for AIX	(2,5,12)	B.02	5765-352
AIX OSL/6000	(2)	1.2.1	5621-013
Parallel I/O File System	(1,3a,7,17)	1.1	5765-297
Parallel OSL (OSLp)	(2,3,17)	1.1.1	5765-392
Parallel Visual Explorer for AIX (PVE)	(1,5)	1.1,1.1.1	5765-469
XL Pascal for AIX		2.1	5765-245
Print Services Facility for AIX (PSF)	(9)	2.1	5765-505
Printing Systems Manager for AIX	(5)	1.2	5765-273
Printing Systems Manager Graphical User Interface for AIX	(5)	1.2	5765-541
Recoverable Virtual Shared Disk	(1,2,3a)	1.1	5765-444
REXX/6000	(2,5,8)	1.1	5764-057
RMONitor for AIX	(2,5,9,12)	1.1.2	5765-292
SMARTsort for Workstations V1.1		1.1	27H8163
SOMobjects Developer Toolkit	(5)	2.1	10H9767
SOMobjects Workgroup Enabler	(5)	2.1	10H9769
Time and Place/6000	(1,3)	1.1,1.1.1	5765-319
Trouble Ticket for AIX Version 3.2	(2,5,9,13)	3.2	5765-265
UNIX Presentation System	(2)		5758-AT9
VisualGen Workgroup Services for AIX	(2)	2.0	5622-587 or 31H3711
Visualization Data Explorer	(5,12)	2.1	5765-210
Visualization Data Explorer V3.1	(1,17)	3.1	5765-586
Visualization Data Explorer for SMP V3.1	(17)	3.1	5765-587
Visualizer Query for AIX/6000	(2)	1.1	5765-326
Visualizer Query for AIX	(2)	1.2	33H2635
Wabi for AIX	(1)	2.0	5765-315
Workstation Interactive Test Tool for X-Windows (X/WITT)	(2)	1.1	5765-222
3270 Emulator For X Window System (x3270)	(3,5)	1.2.2	5765-011

12.3 Packaged LPP Solution Offerings for AIX Version 4.2

This section does *not* include information about similar offerings for AIX Version 3.2.5.

SystemView for AIX:

- Also known as IBM SystemView Server for AIX
- Requires AIX 4.1.3 or later
- Please refer to the announcement letter to see which features will work with both NetView (3.1) for AIX and/or NetView (4.1) for AIX.
- Features are listed below in feature code order

SYSTEMVIEW FOR AIX		5765-527
Launch Window	(3d)	FC 5600
NetView (3.1) for AIX	(3)	FC 5601
LAN Management Utilities	(3d)	FC 5605
LAN Network Manager (1.1) for AIX	(2,3,18)	FC 5607
NetView (4.1) for AIX	(4)	FC 5608
Distributed SMIT (2.2)	(3d)	FC 5615
Job Scheduler (1.2) for AIX	(2,3c,18)	FC 5617
Print Services Facility (2.1)	(3d)	FC 5618
Page Printer Formatting Aid (1.2)	(3d)	FC 5619
Performance Reporter (1.2) for AIX	(2,3d,18)	FC 5622
Trouble Ticket (3.2) for AIX	(3)	FC 5623
NetView Distribution Manager See FC 6605 too.	(3d)	FC 5624
NetView Distribution Manager Agents		FC 5625
Extended Systems Administration (1.2)	(2,3d,18)	FC 5628
Systems Monitor (2.2, 2.3)	(3d)	FC 5632
SNA Manager for AIX	(3d)	FC 6401
Nways Campus Manager for Remote Monitor for AIX	(2,3c)	FC 6402
Nways Campus Manager for Remote Monitor Advance for AIX	(2,3c)	FC 6403
Printing Systems Manager (1.2) (and PSM GUI)	(3d)	FC 6404
CICS System Manager	(3d)	FC 6405
Application Manager for SAP R/3	(2,3d)	FC 6406
DataHub for UNIX Operating Systems	(3d)	FC 6600
ADSTAR Distributed Storage Manager (without HSM Client)	(3d)	FC 6601
SystemView Agent for AIX	(3d,18)	FC 6602
Performance Monitoring of SNA Networks (1.1)	(3d)	FC 6603
Performance Monitoring of Applications (1.1)	(3d)	FC 6604
Software Distribution (3.1) for AIX manager and AIX client (replaces NetView DM)	(3d)	FC 6605
Performance Monitoring of Application Agents (1.1)	(3d)	FC 6606
Nways Campus Manager LAN for AIX	(2,3c)	FC 6608
LoadLeveler (1.2.1)	(3b,17)	FC 6609
Nways Campus Manager ATM for AIX	(3c,2)	FC 6611
Nways Broadband Switch Manager	(2,3c,18)	FC 6614

The following features of SystemView have been withdrawn. Please refer to SystemView announcements for details on the modification level of AIX that was supported for these features.

Intelligent Hub Manager	(3,12)	FC 5609
Intelligent Hub Manager Entry	(3,12)	FC 5610
ATM Campus Manager	(3,12)	FC 5613
RMONitor	(3,12)	FC 5629
Systems Information Agent of Systems Monitor	(3,12)	FC 5633

Internet Power Solutions for AIX-Internet Connection Server

- See US Hone Announcement Letter 695-015 October 10, 1995
- Internet POWERsolutions for AIX - Internet Connection Server
- Internet POWERsolutions for AIX - Internet Connection Secure Server

Internet Power Solutions for AIX - Netscape Servers

- See US Hone Announcement Letter 695-016 October 10, 1995

IBM Software Servers for AIX Version 4 (2,5,18)

- An integrated suite of software servers listed below that have been tested together, have simplified terms and conditions, have simplified installation

procedures, and provide a single service and support strategy. The servers have been tested on AIX Version 4.1.4.

- Please see the overview announcement letter 296-071 for details.

  IBM SystemView Server for AIX, Version 4 5765-527

- See US Hone Announcement Letter 296-080 March 12, 1996

  IBM Directory and Security Server for AIX Version 4 5765-639

- See US Hone Announcement Letter 296-078 March 12, 1996

  IBM Internet Connection Server for AIX Version 4 5765-638

- See US Hone Announcement Letter 296-077 March 12, 1996

  IBM Transaction Server for AIX Version 4 (4*) 5697-251

- See US Hone Announcement Letter 296-076 March 12, 1996

  IBM Database Server for AIX Version 4 5765-642

- See US Hone Announcement Letter 296-074 March 12, 1996

  IBM Communication Server for AIX Version 4 (4*) 5765-652

- See US Hone Announcement Letter 296-073 March 12, 1996

  Lotus Notes Server

- See US Hone Announcement Letter 296-071 March 12, 1996

12.4 Early Support and/or Beta Programs

Early support programs have been announced for the following:

LPP Name	For More Information
World Wide Web Connection	See World Wide Web for info: http://www.software.ibm.com (--> Hot Stuff) See HONE Announcement 295-279
Distributed Security Manager for AIX	Per HONE Announcement 295-454 Contact IBM Marketing Rep.-->Contact Country Marketing -->Contact Armin Hummel at location in Anno. Letter
HIPPI (SMP Support)	Contact Jerry Chapman AUSVM6(JCHAPMAN)

12.5 Footnotes

The following table explains the footnotes that have been used throughout this chapter.

(1) Uni-processor (UP) support only
(2) SMP support information pending (generally means it hasn't been tested.)
(3) This LPP is not supported on AIX 4.2.
(3a) Support for AIX 4.2 planned for the 1st Half of 1997.
(3b) Support for AIX 4.2 planned for the 1st Quarter of 1997.
(3c) Support for AIX 4.2 planned for the 2nd Half of 1996.
(3d) Support for AIX 4.2 planned for the 2nd Quarter of 1996.
(4) Not supported on AIX 4.2, but is replaced by product with (4*) footnote which IS supported on AIX 4.2. In the case of the NetView for AIX V4 feature of SystemView, this will be replaced by a standalone product in the near future that will support AIX V4.2.
(4*) These LPPs do support AIX 4.2. They are follow-on products for LPPs that are not themselves supported on AIX 4.2.
(5) Product is not supported in CFRS6000 for AIX Version 4.
Specific Notes:
MQSeries V2.2.0 is supported in CFPC - see HONE Announcements (295-145 950328) for details about US GEMS and/or AAS ordering options. For InfoCrafter, please use the AIX 3.2 path of CFRS6000.
CFRS6000 Support on 4/23/96 for PSM/PSMGUI, DataHub for UNIX Rel. 2, BookManager BookServer V2.0 and the IBM Software Servers in Section 3.
PCRS6000 is already updated for the IBM Software Servers.
Other products with this footnote may be in the AIX 3.2 path of the configurator, in other paths of the configurator, or not in the configurator at all.
(6) Only token ring and ethernet connections between AADU and DOS Server are supported.
(7) Program Update (formerly called PTF) required.
(8) RPQ or PRPQ required. (See HONE Application RPQ.)
(9) See other RoadMap Sections for information about other versions/releases/levels of this product or a replacement of this product.
(10) Product has been withdrawn from IBM Marketing, however, IBM will continue to ship DOS Server in AIX Version 4.1. Please see Announcement Letter 295-152 for a vendor alternative to AADU.
(11) There are several Geographic Information Systems (GIS) products that are available for AIX V4 via PRPQ only. Please see HONE Industry Option "GIS", Section "PRPQ".
(12) An announcement has been made to withdraw this product from Marketing. Please see HONE for the effective dates and details.
(13) PTF/Upgrade U437812 is required. This PTF + LPP work ONLY on systems that have had AIX V3.2.5 upgraded to AIX V4.
(14) These device drivers are available (device driver cost is included in the cost of the adapter card) either for upgrades or new installations from the WWW (by end of January 1996) at http://wwprodsoln.bocaraton.ibm.com/artic/index.html or (as of 12/15/95) through a Bulletin Board System accessed at 407-443-0134 (worldwide).
(15) This program runs on a client system and therefore SMP status in not likely to be relevant.

(16 These are not "really" LPPs (AFS is sold by Transarc and Lotus Notes by Lotus; not IBMers), IBM has purchased Transarc and Lotus and many people inquire about their status for AIX V4. Due to testing resource constraints, Transarc announced support for AFS on AIX 4.1.1, 4.1.3, and AFS V3.4a supports AIX 4.1.4. Customers wanting to purchase AFS can email to sales@transarc.com or to afssales@transarc.com or call 412-338-4400.
(17 Parallel mode is supported on SP Systems.
(18) SP support information pending (generally means it hasn't been tested.)
(19) The IBM Software Servers were announced March 12, 1996. Refer to Section (12.3, "Packaged LPP Solution Offerings for AIX Version 4.2" on page 243). These servers are for AIX Version 4 (not AIX 3.2.5). CICS, CICS System Manager and the Encina LPPs functions are now included in the IBM Transaction Server for AIX and the SNA Server for AIX function is in the IBM Communications Server for AIX.
(20) Customers using AIX applications that are tightly integrated with the AIX Common Desktop Environment (CDE) including UIM/X, IBM Ultimedia Services, or Software Development Environment (SDE, the graphical toolkit packaged with AIX compilers) may need to upgrade to the latest level of that product to run with CDE 1.0 which is the default desktop supplied with AIX 4.1.3 or later. Updates for SDE are automatically installed when CDE 1.0 is installed.
(21) Open Systems Standard Communications (OSSC) detailed information is included in Announcement Letter 995-063.
(22) UMS V2.1.4 will be shipping on the bonus pack with AIX V4.2 (no charge), but is still available as an LPP for AIX V4.1.

12.6 Status of Available ISV Applications - AIX Version 4

The following update represents Independent Software Vendor (ISV) Applications that are available on AIX Version 4.

Product/Vendor	Status	Description/Release
Oracle	Available	v 7.1.4
Informix	Available	v 7.1.0
Sybase	Available	v 10.0.2/SMP Cat#: 12606
Progress	Available	v6.3F, v7.3
Software AG	Available	ADABAS-C v2.2/Uni
	Available	Entire Network V2.1/Uni
	Available	NATURAL/Uni&SMP
	Available	NATURAL Security/Uni&SMP
Redbrick	Available	v2.1,v3.0
CA/Ingres	Available	OpenIngres/Uni
Object Design	Available	Look!
Information Builders	Available	FOCUS v6.5.1
	Available	EDA/SQL
Information Mgmt Co.	Available	Tuxedo ETP
PICK Systems	Available	Advanced PICK
UniData	Available	UniData RDBMS
VMark	Available	UniVerse
SAP	Available	R/3 - Uni & SMP
Baan	Available	Triton
qad.inc	Available	MFG/PRO
Oracle	Available	v 10.5

Marcam	Available	MXP
SSA	Available	BPCS
Camstar	Available	
Cimlinc	Available	Linkage
FASTech	Available	CELLworks
GE Fanuc	Available	CIMPLICITY
Gensym	Available	G2. Dynamic Scheduling Package
Hilco	Available	MONITROL/US
US Data	Available	FactoryLink
Vertex	Available	BridgeNet
Dassault France	Available	CATIA/4OP only
Dassault US	Available	CADAM 3.6
Autodesk	Available	Release 12
	Available	HOOPS
SDRC	Available	I-DEAS v 2.0
MacNeal Schwendler	Available	NASTRAN v 68.1
StereoCAD	Available	
SAS	Available	SAS/Uni
Cadence	Available	Contact Vendor for more information
Mentor Graphics	Available	Contact Vendor for more information
ViewLogic	Available	Powerview 5.3.1
ESRI	Available	ARC/Info
Halliburton ES	Available	Contact Vendor for more information
AVS	Available	AVS 5.0
Reuters (Effix)	Available	ATW
Teknekron	Available	Rendezvous
Micrognosis	Available	MIPS (3.25 Binaries on AIX 4.1)
	Available	Infotrade
HBO & Co.	Available	CLINSTAR
EPIC Healthcare	Available	EPICare
	Available	Managed Care
	Available	Cadence
	Available	Resolute
Intersystems	Available	Open M Enterprise C/S
Cerner	Available	PathNet
	Available	RadNet
	Available	MS-Meds
	Available	Open Engine
	Available	Open Clinical Foundation
PDX, Inc	Available	Pharmacy Solution
Biosym Tech	Available	Converter
	Available	DMol
	Available	DelPhi
	Available	Discover
	Available	Homology
	Available	Insight II
	Available	NMRchitect
	Available	Polymer
	Available	Small Molecule
	Available	Solid State SW Modeling
	Available	TurboMole
Molecular Simulation	Available	Quanta
	Available	Cerius2
	Available	CHARMm
	Available	XPLOR
Wavefunction	Available	Spartan
Aspen	Available	Aspen Plus
Computer Associates	Available	Unicenter
Candle	Available	Contact Vendor for more information
Frame	Available	FrameMaker 4
Interleaf	Available	Interleaf 5.4, 6.0
Lotus	Available	Notes/Uni
		Office

Novell (WordPerfect)	Available	WordPerfect 6.0
		Office
	Available	SoftWindows v1.2
VSystems	Available	VSI-Fax
Geo. Davidson & Son	Available	Contact Vendor for more information
Softlinx	Available	Replix
MicroFocus	Available	COBOL
Taligent	Available	CommonPoint Toolkit Beta
Synon		Contact Vendor for more information
Template	Available	SNAP
Franz, Inc	Available	Allegro Common LISP
Neuron Data	Available	Open Interface Elements
AimTech	Available	IconAuthor (MM Authoring)
Insoft	Available	Communique! (Video Conferencing)
KI Research	Available	OpenDNA, OpenDNM

Appendix A. Problem Determination Update

This appendix is intended as a quick reference guide to the areas of AIX Version 4.2 which have changed significantly and may be the cause of unexpected behavior and/or problems. It contains a list of the commands which are new or have been updated in AIX V4.2. In addition it refers the reader to sections of this redbook containing more in-depth information for particular subjects.

The following commands are new to AIX Version 4.2:

hps_dump The hps_dump command dumps the contents of a Network Terminal Accelerator adapter memory to a disk file for later analysis and debugging. See A.1, “hps_dump Command” on page 253.

ntpdate The ntpdate command sets the local systems date and time by polling the specified Network Time Protocol servers. It obtains a number of samples from each server and uses standard filters and algorithms to choose the best sample. See 8.4.3, “ntpdate Command” on page 171.

ntpq The ntpq command starts the Network Time Protocol query program. It allows the user to query the NTP servers on the specified hosts for their current state and if required, request changes to that state. See 8.4.4, “ntpq Command” on page 172.

ntptrace The ntptrace command traces a chain of NTP hosts back to their master time source. It effectively determines where a given NTP server obtains its time from. See 8.4.6, “ntptrace Command” on page 177.

rtl_enable The rtl_enable command relinks a module or an archive containing modules to enable run-time linking. See 3.13.5, “rtl_enable Command” on page 81.

xntpdc The xntpdc command starts the query program for the xntpd daemon. It allows the users to query the state of the xntpd daemon and make changes to that state if required. Additionally, nearly all the configuration options that can be specified at startup by the xntpd configuration file can also be specified at run time using the xntpdc command. See 8.4.7, “xntpdc Command” on page 178.

A new daemon is included in AIX Version 4.2:

xntpd The xntpd daemon sets and maintains the systems time and date in compliance with Internet standard time servers. In AIX V4.2 it is a complete inplementation of the NTP Version 3 standard as defined by RFC 1305. See 8.4.8, “xntpd Daemon” on page 186.

The following existing commands that have been changed in AIX Version 4.2:

bosboot The bosboot command now has a -v and a -T flag. The -v flag will verify but not build a boot image while the -T flag specifies the hardware platform type. See 3.24, “bosboot Command” on page 99.

bootinfo Support for the bootinfo command has been withdrawn in AIX Version 4.2. See 3.22, “Removal of Support for bootinfo Command” on page 92.

© Copyright IBM Corp. 1996

bootlist The bootlist command has been enhanced by addition of the -o flag. When used in conjunction with the -m flag it will print out the device list specified by the -m flag. See 3.23, “Reading bootlist Information” on page 95.

chvg chvg now has -c and -x flags for use with Concurrent Capable volume groups. See 3.12.5, “chvg Command” on page 71.

crfs. Two new options are now recognised with the -a flag to crfs which allow creation of large file enabled filesystems. See 3.3.3, “crfs Command” on page 34.

ifconfig The ifconfig command now allows tcp/ip checksumming to be turned on or off. See 8.6.1, “Option to Disable TCP/UDP Checksum” on page 193.

importvg importvg now has -c and -x flags for use with Concurrent Capable volume groups. See 3.12.3, “importvg Command” on page 69.

ld Several new options and features have been added to the ld command for use with run-time linking. See 3.13.4, “Linker Changes” on page 77.

lslpp lslpp has a new -w flag that allows users to list which fileset a particular file belongs to. A precondition is that the paticular fileset must be installed on the system. See 1.7, “lslpp Enhancement: lslpp -w” on page 6.

mkfs Similar to crfs the mkfs command now supports new options to the -o flag to allow creation of large file enabled filesystems. See 3.3.4, “mkfs Command” on page 35.

mksysb mksysb in AIX Version 4.2 supports the -v and -p flags for listing and packing of files respectively. See 3.11.5, “New mksysb Options” on page 65.

mkvg For use with Concurrent Capable volume groups the mkvg command now supports the -c and -x flags. See 3.12.4, “mkvg Command” on page 70.

no The no command has been enhanced to allow display and setting of Streams parameters. See 8.5, “Streams” on page 189.

shutdown The shutdown command now recognizes the -p flag which performs a shutdown without powering off the system. See 3.21, “Power-Off Facilities” on page 89.

varyonvg varyonvg has been enhanced to make volume groups accessible to multi-initiator nodes without use of the HACMP product. New options have been added to provide this facility. See 3.12.2, “varyonvg Command” on page 68.

The following command has been removed in AIX Version 4.2:

lvedit The lvedit command has been removed in AIX Version 4.2. Better functionality is provided by the xlvm command. See 9.3, “lvedit Command” on page 201.

A.1 hps_dump Command

hps-dump dumps contents of Network Terminal Accelerator (NTX) adapter memory to a host file.

```
hps_dump [ -f Name ] [ -d Device ]
```

The hps_dump command uses the loader interface to upload all of the memory from the adapter board into a file. This produces a snapshot of a system for later analysis and debugging. The first 1024 bytes of the file contains the following:

80 bytes Identification string, includes version.

80 bytes Time and date of dump from host system.

80 bytes Comments.

268 bytes Log table from the host adapter.

32 bytes System address table.

8 bytes Starting and ending address range of dump.

476 bytes Padding to 1024 bytes total.

A.1.1.1 Flags

-f Name Specifies the name of the dump. Use this option to override the default filename hpscore.

-d Device Specifies the raw device file name of the adapter. Use this option to override the default device name dev/rhp0.

A.1.1.2 Examples

To get a dump of memory of the default adapter to the file hpscore in the current directory, enter:

```
hps_dump
```

To get a dump of memory of the default adapter to the file hpsdebug in the current directory of the default adapter, enter:

```
hps_dump -f hpsdebug
```

To get a dump of memory of the adapter dev/rhp1 to the file hpsdebug in the current directory of the default adapter, enter:

```
hps_dump -f hpsdebug -d dev/rhp1
```

A.1.1.3 Disk Media Service Aid

When displaying the disks to be selected for format and/or certify under the Disk Media Service Aid, the size and description of the drive is displayed. This aids in identifying the correct drive to be formatted andor certified.

Appendix B. Special Notices

This publication is intended to help systems administrators and developers working with AIX Version 4.2 understand the key technical differences between this release and earlier versions of AIX. In particular AIX Version 4.1. The information in this publication is not intended as the specification of any programming interfaces that are provided by AIX Version 4.2. See the PUBLICATIONS section of the IBM Programming Announcement for AIX Version 4.2 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, 500 Columbus Avenue, Thornwood, NY 10594 USA.

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 them into the customer′s operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

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

© Copyright IBM Corp. 1996

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

AIX	AIX/6000
AS/400	BookManager
CICS	IBM
InfoCrafter	InfoExplorer
LoadLeveler	Micro Channel
NetView	Nways
OS/2	POWER Gt3
PowerPC	RISC System/6000
RS/6000	System/390
SystemView	Time and Place
Ultimedia	

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Microsoft, Windows, and the Windows 95 logo
are trademarks or registered trademarks of Microsoft Corporation.

Java and HotJava are trademarks of Sun Microsystems, Inc.

Acrobat	Adobe, Inc.
Laserjet	Hewlett-Packard Company
NCS	Apollo Computer, Inc.
Netscape	Netscape, Inc.
Network File System, NFS	Sun Microsystems, Inc.
Net/LS, iFOR/LS	Gradient Technologie, Inc.
ObjectStore	Object Design, Inc.
OSF, Motif, Open Software Foundation	Open Environment Corporation
POSIX	Institute of Electrical and Electronic Engineers
X/Open	X/Open Company Limited
X-Windows, Athena	Massachusetts Institute of Technology

Other trademarks are trademarks of their respective companies.

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 261.

- *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

C.2 Redbooks on CD-ROMs

Redbooks are also available on CD-ROMs. **Order a subscription** and receive updates 2-4 times a year at significant savings.

CD-ROM Title	Subscription Number	Collection Kit Number
System/390 Redbooks Collection	SBOF-7201	SK2T-2177
Networking and Systems Management Redbooks Collection	SBOF-7370	SK2T-6022
Transaction Processing and Data Management Redbook	SBOF-7240	SK2T-8038
AS/400 Redbooks Collection	SBOF-7270	SK2T-2849
RISC System/6000 Redbooks Collection (HTML, BkMgr)	SBOF-7230	SK2T-8040
RISC System/6000 Redbooks Collection (PostScript)	SBOF-7205	SK2T-8041
Application Development Redbooks Collection	SBOF-7290	SK2T-8037
Personal Systems Redbooks Collection	SBOF-7250	SK2T-8042

C.3 Other Publications

These publications are also relevant as further information sources:

- *AIX Version 4 Getting Started*, SC23-2527
- *AIX Version 4.2 Installation Guide*, SC23-1924
- *AIX Version 4.2 Quick Installation Guide*, SC23-1925
- *AIX Version 4.2 Network Installation Management Guide and Reference*, SC23-1926
- *AIX Version 4 System Management Guide: Operating System and Devices*, SC23-2525
- *AIX Version 4 System Management Guide: Communications and Networks*, SC23-2526

© Copyright IBM Corp. 1996

- *AIX Version 4 System User's Guide: Operating System and Devices*, SC23-2544
- *AIX Version 4 System User's Guide: Communications and Networks*, SC23-2545
- *AIX Version 4 Problem Solving Guide and Reference*, SC23-2606
- *AIX Version 4 Messages Guide and Reference*, SC23-2641
- *AIX Versions 3.2 and 4 Performance Tuning Guide*, SC23-2365
- *AIX Version 4 Files Reference*, SC23-2512
- *AIX Version 4 Commands Reference*, SBOF-1851 (Contains the following publications that may also be ordered separately:)
  - *AIX Version 4 Commands Reference, Volume 1*, SC23-2537
  - *AIX Version 4 Commands Reference, Volume 2*, SC23-2538
  - *AIX Version 4 Commands Reference, Volume 3*, SC23-2539
  - *AIX Version 4 Commands Reference, Volume 4*, SC23-2540
  - *AIX Version 4 Commands Reference, Volume 5*, SC23-2639
  - *AIX Version 4 Commands Reference, Volume 6*, SC23-2640
- *X/Open Single Unix Specification Go Solo: How to Implement and Go Solo with the Single Unix Specification*, SR28-5705
- *AIX Version 4 General Programming Concepts: Writing and Debugging Programs*, SC23-2533
- *AIX Version 4 Communications Programming concepts*, SC23-2610
- *AIX Version 4 Technical Reference*, SBOF-1852 (Contains the following publications that may also be ordered separately:)
  - *AIX Version 4 Technical Reference, Volume 1: Base Operating System and Extensions*, SC23-2614
  - *AIX Version 4 Technical Reference, Volume 2: Base Operating System and Extensions*, SC23-2615
  - *AIX Version 4 Technical Reference, Volume 3: Communications*, SC23-2616
  - *AIX Version 4 Technical Reference, Volume 4: Communications*, SC23-2617
  - *AIX Version 4 Technical Reference, Volume 5: Kernel and Subsystems*, SC23-2618
  - *AIX Version 4 Technical Reference, Volume 6: Kernel and Subsystems*, SC23-2619
  - *AIX Version 4 Technical Reference, Volume 7: AIXwindows*, SC23-2620
  - *AIX Version 4 Technical Reference, Volume 8: Enhanced Xwindows*, SC23-2621
  - *AIX Version 4 Technical Reference, Volume 9: Enhanced Xwindows*, SC23-2622
  - *AIX Version 4 Technical Reference, Volume 10: Enhanced Xwindows*, SC23-2623
  - *AIX Version 4 Technical Reference, Volume 11: Master Index*, SC23-2624

- *AIX Version 4 Quick Reference*, SC23-2529
- *AIX Version 4 iFOR/LS Tips and Techniques*, SC23-2666
- *AIX Version 4 AIXwindows Programming Guide*, SC23-2632
- *AIX Version 4 Enhanced Xwindows Programming Guide*, SC23-2636
- *Common Desktop Environment 1.0 for AIX*, SBOF-1869 (Contains the following Publications that may also be ordered separately:)
  - *Common Desktop Environment 1.0: Application Builder User's Guide*, SC23-2785
  - *Common Desktop Environment 1.0: Desktop KornShell User's Guide*, SC23-2786
  - *Common Desktop Environment 1.0: Help System Author's and Programmer's Guide*, SC23-2787
  - *Common Desktop Environment 1.0: Internationalization Programmer's Guide*, SC23-2788
  - *Common Desktop Environment 1.0: Programmer's Overview*, SC23-2789
  - *Common Desktop Environment 1.0: Programmer's Guide*, SC23-2790
  - *Common Desktop Environment 1.0: Style Guide and Certification Checklist*, SC23-2791
  - *Common Desktop Environment 1.0: ToolTalk Messaging Overview*, SC23-2792
  - *Common Desktop Environment 1.0: User's Guide*, SC23-2793
  - *Common Desktop Environment 1.0: Advanced User's and System Administrator's Guide*, SC23-2795

How To Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, CD-ROMs, workshops, and residencies. A form for ordering books and CD-ROMs is also provided.

This information was current at the time of publication, but is continually subject to change. The latest information may be found at URL http://www.redbooks.ibm.com.

How IBM Employees Can Get ITSO Redbooks

Employees may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **PUBORDER** — to order hardcopies in United States
- **GOPHER link to the Internet** - type GOPHER.WTSCPOK.ITSO.IBM.COM
- **Tools disks**

  To get LIST3820s of redbooks, type one of the following commands:

  ```
  TOOLS SENDTO EHONE4 TOOLS2 REDPRINT GET SG24xxxx PACKAGE
  TOOLS SENDTO CANVM2 TOOLS REDPRINT GET SG24xxxx PACKAGE (Canadian users only)
  ```

  To get lists of redbooks:

  ```
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET ITSOCAT TXT
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET LISTSERV PACKAGE
  ```

  To register for information on workshops, residencies, and redbooks:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ITSOREGI 1996
  ```

  For a list of product area specialists in the ITSO:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ORGCARD PACKAGE
  ```

- **Redbooks Home Page on the World Wide Web**

  http://w3.itso.ibm.com/redbooks

- **IBM Direct Publications Catalog on the World Wide Web**

  http://www.elink.ibmlink.ibm.com/pbl/pbl

  IBM employees may obtain LIST3820s of redbooks from this page.

- **REDBOOKS category on INEWS**
- **Online** — send orders to: USIB6FPL at IBMMAIL or DKIBMBSH at IBMMAIL
- **Internet Listserver**

  With an Internet E-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an E-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank). A category form and detailed instructions will be sent to you.

© Copyright IBM Corp. 1996

How Customers Can Get ITSO Redbooks

Customers may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Online Orders** (Do not send credit card information over the Internet) — send orders to:

	IBMMAIL	**Internet**
In United States:	usib6fpl at ibmmail	usib6fpl@ibmmail.com
In Canada:	caibmbkz at ibmmail	lmannix@vnet.ibm.com
Outside North America:	dkibmbsh at ibmmail	bookshop@dk.ibm.com

- **Telephone orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	(long distance charges apply)
(+45) 4810-1320 - Danish	(+45) 4810-1020 - German
(+45) 4810-1420 - Dutch	(+45) 4810-1620 - Italian
(+45) 4810-1540 - English	(+45) 4810-1270 - Norwegian
(+45) 4810-1670 - Finnish	(+45) 4810-1120 - Spanish
(+45) 4810-1220 - French	(+45) 4810-1170 - Swedish

- **Mail Orders** — send orders to:

IBM Publications	IBM Publications	IBM Direct Services
Publications Customer Support	144-4th Avenue, S.W.	Sortemosevej 21
P.O. Box 29570	Calgary, Alberta T2P 3N5	DK-3450 Allerød
Raleigh, NC 27626-0570	Canada	Denmark
USA		

- **Fax** — send orders to:

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	(+45) 48 14 2207 (long distance charge)

- **1-800-IBM-4FAX (United States)** or **(+1) 415 855 43 29 (Outside USA)** — ask for:

  Index # 4421 Abstracts of new redbooks
  Index # 4422 IBM redbooks
  Index # 4420 Redbooks for last six months

- **Direct Services** - send note to softwareshop@vnet.ibm.com
- **On the World Wide Web**

Redbooks Home Page	http://www.redbooks.ibm.com
IBM Direct Publications Catalog	http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Internet Listserver**

  With an Internet E-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an E-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank).

IBM Redbook 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.

DO NOT SEND CREDIT CARD INFORMATION OVER THE INTERNET.

List of Abbreviations

AIX	Advanced Interactive Executive
AIX V3.2	AIX Version 3.2 Operating System
AIX V4.1	AIX Version 4.1 Operating System
AIX V4.2	AIX Version 4.2 Operating System
ARTIC	A Real Time Interface Coprocessor
ASCII	American Standard Code for Information Interchange
BOS	Base Operating System
CD	Compact Disk
CDE	Common Desktop Environment
CLVM	Concurrent Logical Volume Manager
CD-ROM	Compact Disk Read Only Memory
DOS	Disk Operating System
FDDI	Fiber Distributed Data Interface
GLB	Global Location Broker
GUI	Graphical User Interface
HACMP	High Availability Cluster MultiProcessing
HFT	High-Function Terminal
HP	Hewlett-Packard
IBM	International Business Machines Corporation
iFOR/LS	information For Operation and Retrieval/License System
IPL	Initial Program Load
ISV	Independent Software Vendor
ITSO	International Technical Support Organization
JFS	Journalled File System
LAN	Local Area Network
LCD	Liquid Crystal Display
LED	Light Emitting Diode
LFT	Low-Function Terminal
LPP	Licensed Program Product
LVM	Logical Volume Manager
MCA	Micro Channel Architecture
MP	Multiprocessor
MPA	Multi-Protocol Adapter
MTU	Maximum Transmission Unit
NCS	Network Computing System
NFS	Network File System
NIM	Network Installation Manager
NLS	Network Language Support
NTA	Network Terminal Accelerator
ODM	Object Data Manager
OS	Operating System
PC	Personal Computer
PCI	Peripheral Component Interconnect
PMP	Preventive Maintenance Package
POSIX	Portable Operating System Interface for Computer Environments
POST	Power-On Self Test
POWER	Performance Optimization With Enhanced RISC
PP	Physical Partitions
PTF	Program Temporary Fix
RAM	Random Access Memory
RFC	Request For Comment
RISC	Reduced Instruction Set Computer or Reduced Instruction Set Cycles
RS/6000	IBM RISC System/6000
SCSI	Small Computer Systems Interface
SMIT	System Management Interface Tool
SMP	Symetric Multiprocessor
SP2	IBM Scalable POWERparallel Systems 2
TCP/IP	Transmission Control Protocol / Internet Protocol
TTY	TeleType or Asynchronous Terminal
UDP	User Datagram Protocol
UNIX	UNIX Operating System
UK	United Kingdom
UP	UniProcessor

© Copyright IBM Corp. 1996

US	United States of America	***WDFM***	Withdrawn From Marketing
VG	Volume Group	***WWW***	World Wide Web
VSM	Visual Systems Manager	***XPG4***	X/Open Portability Guide Issue 4

Index

Numerics

A

B

C

D

© Copyright IBM Corp. 1996

E

F

G

H

I

J

K

L

M

N

P

R

S

T

U

V

W

X

Printed in U.S.A.