IBM

IBM AIX Version 6.1 Differences Guide

AIX - The industrial strength UNIX operating system

AIX Version 6.1 enhancements explained

An expert's guide to the new release

Roman Aleksic
Ismael "Numi" Castillo
Rosa Fernandez
Armin Röll
Nobuhiko Watanabe

Redbooks

ibm.com/redbooks

International Technical Support Organization

IBM AIX Version 6.1 Differences Guide

March 2008

SG24-7559-00

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

First Edition (March 2008)

This edition applies to AIX Version 6.1, program number 5765-G62.

Contents

Figures

Tables

Notices

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

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

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

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

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

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

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

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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

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

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

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

AIX 5L™
AIX®
alphaWorks®
Blue Gene®
DPI®
DS4000™
Enterprise Storage Server®
Everyplace®
General Parallel File System™
Geographically Dispersed Parallel Sysplex™
GDPS®
GPFS™
HACMP™
IBM®
Language Environment®
OS/2®
Parallel Sysplex®
PowerPC®
POWER™
POWER3™
POWER4™
POWER5™
POWER5+™
POWER6™
PTX®
Redbooks®
Redbooks (logo) ®
S/390®
System p™
System x™
System Storage™
Tivoli®
WebSphere®
Workload Partitions Manager™
z/OS®

The following terms are trademarks of other companies:

Oracle, JD Edwards, PeopleSoft, Siebel, and TopLink are registered trademarks of Oracle Corporation and/or its affiliates.

InfiniBand, and the InfiniBand design marks are trademarks and/or service marks of the InfiniBand Trade Association.

CacheFS, Java, ONC+, Solaris, Ultra, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

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

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

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

Preface

This IBM® Redbooks® publication focuses on the differences introduced in IBM AIX® Version 6.1 when compared to AIX 5L™ Version 5.3. It is intended to help system administrators, developers, and users understand these enhancements and evaluate potential benefits in their own environments.

AIX Version 6.1 introduces many new features, including workload partitions, advanced security, continuous availability, and managing and monitoring enhancements. There are many other new features available with AIX Version 6.1, and you can explore them all in this publication.

For clients who are not familiar with the enhancements of AIX through Version 5.3, a companion publication, *AIX 5L Differences Guide Version 5.3 Edition*, SG24-7463 is available, along with an addendum, *AIX 5L Differences Guide Version 5.3 Addendum*, SG24-7414, which includes between release enhancements that are available through applying service updates.

The team that wrote this book

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

Roman Aleksic is a System Engineer working for Zürcher Kantonalbank, a major bank in Switzerland. He has seven years of experience with IBM System p and AIX in the fields of application integration, performance management, TCP/IP networking, logical partitioning, and advanced shell scripting. He also implements and supports large HACMP™ and NIM environments.

Ismael "Numi" Castillo is an IBM Senior IT Specialist and Technical Consultant for the IBM ISV Business Strategy and Enablement organization. He has three years of experience in AIX performance tuning and benchmarks. He has 19 years of professional experience in IT with a background in software development, consulting, system performance measurement and tuning, benchmarking, problem determination, and sizing. He is also the team leader for the IBM ISV BSE technical collateral team. Numi completed studies for a Bachelor Degree in Computer Science at the Catholic University of Santo Domingo in Dominican Republic. He also holds several advanced levels industry certifications.

Rosa Fernandez is a Certified Advanced AIX Expert, and an IBM IT Certified professional who joined IBM France in 1990. She holds a Masters degree in Computer Science from Tours University (1985) and is an AIX pre-sales leader since 1996. She is recognized for the management of customer satisfaction, AIX performance delivery, UNIX® software migration, and is incremental in creating and supporting the AIX French User Group. She co-authored the *AIX 64-bit Performance in Focus*, SG24-5103 publication.

Armin Röll works as a System p™ IT specialist in Germany. He has twelve years of experience in System p and AIX pre-sales technical support and, as a team leader, he fosters the AIX skills community. He holds a degree in experimental physics from the University of Hamburg, Germany. He co-authored the AIX Version 4.3.3, the AIX 5L Version 5.0, and the AIX 5L Version 5.3 Differences Guide IBM Redbooks.

Nobuhiko Watanabe is an advisory IT specialist and team leader of the System p and AIX division of IBM Japan Systems Engineering that provides the ATS function in Japan. He has 16 years of experience in the AIX and Linux® fields. He holds a Bachelor degree in Library and Information Science from Kieo University. His areas of expertise also include Solaris™ and HP-UX.

The project that produced this publication was managed by:
Scott Vetter, PMP

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

Janet Adkins, Vishal C Aslot, Dwip N Banerjee, Paul Bostrom, David Bradford, Carl Burnett, David Clissold, Julie Craft, Matthew Cronk, Jim Cunningham, Prakash Desai, Saurabh Desai, Robert Feng, Frank Feuerbacher, Matthew Fleming, Arnold Flores, Kevin Fought, Eric P Fried, Mark Grubbs, Jan Harris, John Harvey, Debra Hayley, David A. Hepkin, Duen-wen Hsiao, Praveen Kalamegham, Jay Kruemcke, Ashley D. Lai, Su Liu, Yantian Lu, Michael Lyons, Brian McCorkle, Marshall McMullen, Dan McNichol, Camilla McWilliams, Bruce Mealey, Dirk Michel, James Moody, Grover Neuman, Dac Nguyen, Frank L Nichols, Frank O'Connell, Matthew Ochs, Michael Panico, Jim Partridge, Steve Peckham, Jose G Rivera, Mark Rogers, Lance Russell, Robert Seibold, Jim Shaffer, Nishant B Shah, Ravi A. Shankar, Saurabh Sharma, David Sheffield, Luc Smolders, Donald Stence, Marc Stephenson, Pedro V Torres, Marvin Toungate, Murali Vaddagiri, Venkat Venkatsubra, Xinya Wang, Suresh Warrier, Ken Whitmarsh, Jonathan A Wildstrom
IBM Austin TX

Arun P Anbalagan, Tejas N Bhise, Abhidnya P Chirmule, Madhusudanan Kandasamy, Neeraj Kumar Kashyap, Manoj Kumar,

Mallesh Lepakshaiah, Pruthvi Panyam Nataraj, G Shantala
IBM India

David Larson
IBM Rochester MN

Francoise Boudier, Bernard Cahen, Damien Faure, Matthieu Isoard, Jez Wain
Bull, France

Bruno Blanchard, Thierry Fauck, Philippe Hermes, Emmanuel Tetreau
IBM France

Bernhard Buehler
IBM Germany

Liviu Rosca
IBM Romania

Become a published author

Join us for a two- to six-week residency program! Help write a book dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

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

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

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

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

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

  ibm.com/redbooks

- Send your comments in an e-mail to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

1

Application development and system debug

This chapter contains the major AIX Version 6.1 enhancements that are part of the application development and system debug category, including:

1.1 Transport independent RPC library

The Open Network Computing Plus (ONC+™) distributed computing environment consists of a family of technologies, services, and tools, including the transport-independent remote procedure call (TI-RPC) API library that provides a distributed application development environment by isolating applications from any specific transport feature. The TI-RPC implementation supports threaded applications and utilizes streams as an interface to the network layer.

Previous AIX releases internally use a comprehensive subset of the TI-RPC API to provide base operating system features, namely the Network File System (NFS) services. In that context, but not limited to it, the AIX operating system also facilitates the RPCSEC_GSS security version of the General Security Services (GSS) API to enable advanced security services. For example, the RPCSEC_GSS routines are used by the AIX Network Data Administration Facility (NDAF).

AIX V6.1 now formally supports the AIX base operating system related subset of the TI-RPC routines as ported from the ONC+ 2.2 source distribution. The code is exported by the network services library (libnsl.a), which is installed by default on any AIX V6.1 system through the bos.net.tcp.client fileset. Additionally, the RPCSEC-GSS security services interface routines are now formally supported and documented in the AIX V6.1 product documentation.

TI-RPC APIs are classified into different levels. These levels provide different degrees of control balanced with different amounts of interface code to implement, in order of increasing control and complexity. The top level classification defines two distinct routine classes:

- Simplified interface routines
- Standard interface routines

The simplified interface routines specify the type of transport to use. Applications using this level do not have to explicitly create handles.

The standard interface routines give a programmer much greater control over communication parameters such as the transport being used, how long to wait before responding to errors and retransmitting requests, and so on.

The standard interface routines are further classified as follows:

Top-level routines These APIs allow the application to specify the type of transport.

Intermediate-level routines These APIs are similar to the top-level APIs, but the user applications select the transport specific information using network selection APIs.

Expert-level routines These APIs allow the application to select which transport to use. These APIs are similar to the intermediate-level APIs with an additional control that is provided by using the name-to-address translation APIs.

Bottom-level routines The bottom level contains routines used for full control of transport options.

Other routines These APIs allow the various applications to work in coordination with the simplified, top-level, intermediate-level, and expert-level APIs.

The AIX V6.1 TI-RPC interface routines listed by classification level are documented in the “Transport Independent Remote Procedure Call” section of Chapter 8, “Remote Procedure Calls”, in *AIX Version 6.1 Communication Programming Concepts*, SC23-5258.

1.2 AIX tracing facilities review

AIX Version 6 has several tracing facilities available:

AIX system trace This is the main trace facility on AIX. It supports tracing of both applications and the kernel.

The AIX system trace facility is designed for tracing inside the kernel and kernel extensions. However, it also supports user-defined tracing in application code. It is based on compiled-in static trace hooks and is only enabled when needed. By default, all trace hooks are enabled when tracing is turned on. However, there are options to enable only a set of trace hooks or to disable some specific trace hooks. Both user and kernel tracing share the same system buffers. So, the application-level trace data is copied to the system buffer.

Light weight memory trace

Light weight memory trace (LMT) traces only key AIX kernel events and is not available in user mode. LMT is also based on compiled-in static trace hooks. It is enabled by default, but it uses a light weight mechanism to record trace data, so the performance impacts are minimal. The trace data is sent to per-CPU buffers and stays in memory until overwritten. There are commands to extract the traced data, and it is displayed using the same tools as AIX system trace. Alternatively, it can also be displayed with the **kdb** command or extracted from a system dump.

Truss

Truss is a tracing mechanism that allows tracing of all system calls and optionally all library calls executed by a specific process. So, traced events are limited to system subroutines calls. Trace output consists of the parameters passed into and the values returned from each system (and library) call. This is directly sent to the standard error of that process. There is no mechanism to save the trace data and there are no system-wide buffers.

Component trace facility

Component trace (CT) is a new tracing facility that became available in AIX starting with AIX V5.3 TL06. The component tracing facility can be used as an additional filter on AIX system trace. It can also be used to provide exclusive in-memory tracing, directed to use either system-wide LMT buffers, or component-specific buffers to save the trace data. Its primary purpose, similar to LMT, is for collecting First Failure Data Capture data for debugging purposes.

POSIX trace

AIX Version 6 implements the POSIX trace system that support tracing of user applications. The POSIX tracing facilities allow a process to select a set of trace event types to activate a trace stream of the selected trace events as they occur in the flow of execution and to retrieve the recorded trace events. Similar to system trace, POSIX trace is also dependent upon precompiled-in trace hooks in the application being instrumented.

1.3 POSIX threads tracing

The Portable Operating System Interface (POSIX) is a registered trademark of the Institute of Electrical and Electronics Engineers (IEEE). POSIX is simultaneously an IEEE standard, an ISO/IEC Standard, and an Open Group Technical standard.

All standards are subject to revision. For the most accurate information about this standard, visit the following Web site:

http://www.opengroup.org/onlinepubs/009695399/mindex.html

POSIX defines a standard operating system interface and environment and it is also referenced as IEEE Std 1003.1-2001 that has been approved by the Open Group under the name of "Single UNIX[1] Specification (version 3)". POSIX is drawn from the base documents:

- The IEEE Std 1003.1-1996 (POSIX-1), incorporating IEEE standards 1003.1-1990, 1003.1b-1993, 1003.1c-1995, and 1003.1i-1995
- The following amendments to the POSIX.1-1990 standard:
  - IEEE P1003.1, a draft standard (additional system services)
  - IEEE Std 10031d.1999 (additional Real-time extensions)
  - IEEE Std 10031g.2000 (Protocol Independent Interface (PII))
  - IEEE Std 10031j.2000 (advanced Real-time Extensions)
  - IEEE Std 10031q.2000 (Tracing)
- The IEEE Std 1003.2-1992 (POSIX-2), incorporating IEEE standards 1003.2a-1992
- The following amendment to the POSIX-2:1993 standard:
  - IEEE P1003.2b draft standard (additional utilities)
  - IEEE Std 1003.2d.1994 (batch environment)
- The Open Group Technical Standard, February 1997, the Base Specification (XBD5, XCU5 and XSH5 sections)
- The Open Group Technical Standard, January 2000, Networking Services (section XNS5.2)
- The ISO/IEC 9899:1999, Programming Languages - C

AIX Version 6 implements the Tracing Option Group, which is an optional function, defined within IEEE Std 1003.1-2001.

[1] UNIX is a registered trademark of The Open Group.

1.3.1 POSIX tracing overview

This section provides an overview of the POSIX tracing facilities as implemented within AIX in the newly POSIX trace library (libposixtrace.a).

The main purposes of tracing are:

- Application debugging during the development stage if the source code is pre-instrumented
- Fault analysis to discover a problem afterwards based on flight recorded data
- A performance measurement tool to check code efficiency

The POSIX trace model is based on two main data types:

Trace event The execution flow of the traced process generates information relative to the program step or action being executed. This program step or action is named a *trace point*, and the traced information a *trace event*. The recorded trace event is contained in the posix_trace_event_info structure, defined in the i/usr/include/trace.h include file.

Trace stream The collection of traced information must be kept, in order to be analyzed, in a place named a *trace stream* that is created for this traced process. It is not mandatory that the traced process creates its associated trace stream. A *trace stream identifier* is returned by the trace stream creation routines and is valid only for the process that made the creation subroutine call. The *trace stream identifier* (trid) is a trace_id_t type defined in the /usr/include/sys/types.h include file. When an offline analysis is required, a *trace log* can be associated with the trace stream.

The POSIX tracing operation relies on three logically different entities:

Traced process The process for which trace events are recorded is named the *traced process*. It is the instrumented code.

Controller process The controller process controls the recording of the trace events into the trace stream. Thus, the controller is in charge to initialize and create the stream, start and stop the tracing, manage the mapping between trace streams and traced processes, and to shut the trace stream down.

Analyzer process The analyzer process retrieves the traced events either at runtime from the trace stream, or at the end of execution as an analysis from a *trace pre-recorded stream* whose content has been obtained reloading the trace stream log.

Figure 1-1 shows the POSIX trace system overview for online analysis.

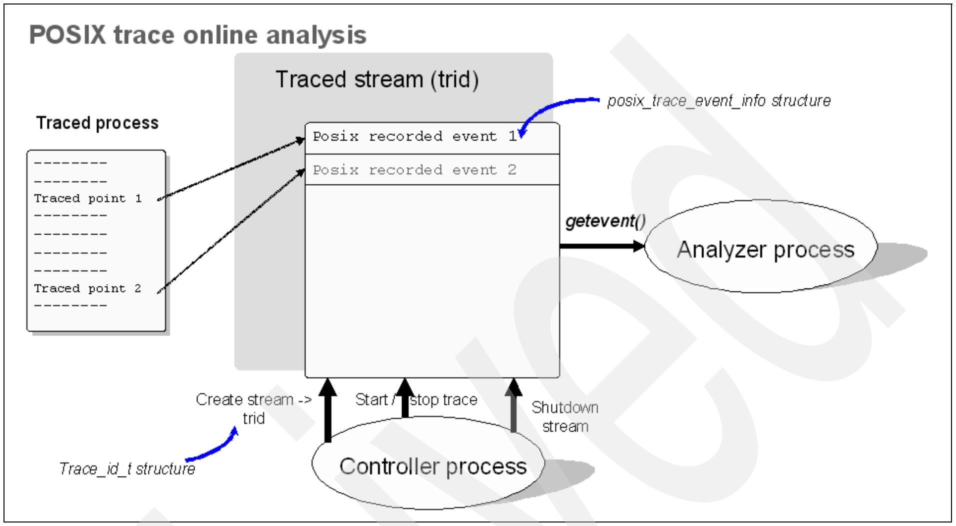


Figure 1-1 POSIX trace system overview: online analysis

Figure 1-2 shows the POSIX trace system overview for offline analysis.

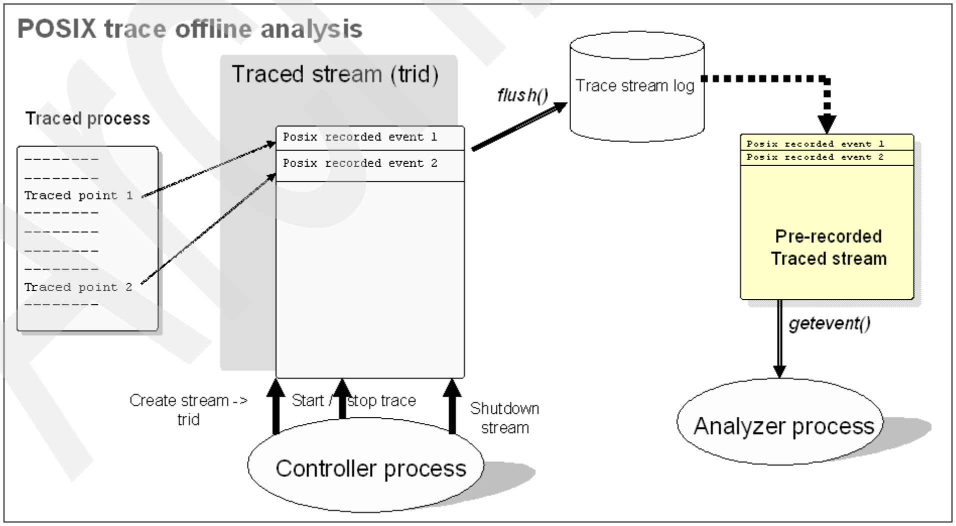


Figure 1-2 POSIX trace system overview: offline analysis

1.3.2 Trace event definition

Each event is identified by a *trace name* and a *trace event identifier* (an internal reference), defined as the trace_event_id_t type in the /usr/include/sys/types.h header file. It has also an associated name returned by the subroutine posix_trace_eventid_get_name().

The event name length in number of characters must be less than TRACE_EVENT_NAME_MAX (defined in the /usr/include/sys/types.h header file).

Trace events belong to two classes, namely:

User trace events Defined and generated by the traced process.

System trace events Defined and generated by the operating system.

User trace events

Each traced process has to define the mapping of the trace event names to trace event identifiers, achieved by calling the posix_trace_eventid_open() subroutine. This subroutine returns a unique trace event identifier to be used on the trace stream. Therefore, the mapping between user event types and user event names are private to the instrumented code and they last only during execution time.

The instrumented code uses this user trace identifier to set a traced point calling the posix_trace_event() subroutine. The execution of a traced point generates a trace event if the trace stream is created, started, and if this traced event identifier is not ignored by filtering (see “Trace stream filtering” on page 11).

Table 1-1 lists the subroutines to define a user trace event and to implement a trace point by an instrumented code.

Table 1-1 User trace event routines used by the instrumented code

Purpose	Subroutine name
Trace subroutine for instrumenting application code	posix_trace_eventid_open()
Trace subroutines for implementing a trace point	posix_trace_event()

A predefined user trace event exists if the limit of per-process user trace event names (TRACE_USER_EVENT_MAX constant) has been reached. Then this user trace event is returned, indicating that the instrumented application is registering more events than allowed.

> **Note:** By default, the instrumented code can define a number of user trace events up to the value of _POSIX_TRACE_USER_EVENT_MAX, constant defined in the file /usr/include/sys/limits.h.
>
> If the limit of the per-process user trace event defined in TRACE_USER_EVENT_MAX (/usr/include/sys/limits.h) has been reached, the POSIX_TRACE_UNNAMED_USEREVENT (/usr/include/trace.h) trace event identifier is returned, indicating that no more event mapping is available for the instrumented application.

Table 1-2 provides the predefined user trace event, defined in the /usr/include/trace.h include file.

Table 1-2 Predefined user trace event

Event ID-Constant	Event name
POSIX_TRACE_UNNAMED_USEREVENT	posix_trace_unnamed_userevent

The following program abstract demonstrates two user trace events names (EV001: snow call and EV002: white call) mapped with two trace event type identifiers to trace snow and white subroutine calls. Trace points use the user trace event data to differentiate the different calls done to the same subroutine:

```
#include /usr/include/sys/types.h
#include /usr/include/trace.h
{
      int ret;
      trace_event_id_t eventid1, eventid2;
      char * data_ptr;
      size_t data_len;
... lines omitted for clarity
/* Definition of user trace events */
      ret=posix_trace_eventid_open("EV001: snow call",&eventid1);
      ret=posix_trace_eventid_open("EV002: white call",&eventid2);
... lines omitted for clarity
/* Trace point EV001: snow call */
      data_ptr=“waking up”;
      data_len=strlen(data_ptr);
      ret=posix_trace_event(eventid1,data_ptr,data_len);
      ret=snow(1);
... lines omitted for clarity
/* Trace point EV002: white call*/
      data_ptr=“laundry white”;
      data_len=strlen(data_ptr);
```

```
      ret=posix_trace_event(eventid2,data_ptr,data_len);
      ret=white(3);
... lines omitted for clarity
/* Trace point EV001: snow call */
      data_ptr="sleeping well";
      data_len=strlen(data_ptr);
      ret=posix_trace_event(eventid1,data_ptr,data_len);
      ret=snow(0);
... lines omitted for clarity
return 0;
}
```

System trace events

The system trace events include a small set of events to correctly interpret the trace event information present in the stream.

Table 1-3 provides the names of defined system trace events.

Table 1-3 System trace events names

Event ID-Constant	Event name
POSIX_TRACE_ERROR	posix_trace_error
POSIX_TRACE_START	posix_trace_start
POSIX_TRACE_STOP	posix_trace_stop
POSIX_TRACE_FILTER	posix_trace_filter
POSIX_TRACE_OVERFLOW	posix_trace_overflow
POSIX_RESUME	posix_trace_resume
POSIX_TRACE_FLUSH_START	posix_trace_flush_start
POSIX_TRACE_FLUSH_STOP	posix_trace_flush_stop

Note: All system trace events identifiers are defined in the /usr/include/trace.h include file.

Trace event sets

The events can be gathered in a set. A set allows you to define which events may be ignored during tracing.

The event set is a trace_event_set_t object. This object must be initialized either by the posix_trace_eventset_empty() or posix_trace_eventset_fill() subroutine.

This event set, as an object, can be only manipulated by specific routines, as described in Table 1-4.

Table 1-4 Trace event sets routines used by instrumented code

Purpose	Subroutine name
Add a trace event type in a trace event type set.	posix_trace_eventset_add()
Delete a trace event type from a trace event type set.	posix_trace_eventset_del()
Empty a trace event type set.	posix_trace_eventset_empty()
Fill in a trace event type set.	posix_trace_eventset_fill()
Test if the trace event type is included in the trace event type set.	posix_trace_eventset_ismember()

There are predefined sets of system trace events, as described in Table 1-5.

Table 1-5 Predefined system trace event sets

Event Set ID	Description
POSIX_TRACE_WOPID_EVENTS	It includes all process independent trace event types.
POSIX_TRACE_SYSTEM_EVENTS	It includes all system trace events, but no AIX kernel events can be traced. It is limited to the available POSIX system trace events.
POSIX_TRACE_ALL_EVENTS	It includes all trace events: user and system.

Trace stream filtering

Traced events may be filtered. Filtering a trace event means to filter out (ignore) this selected trace event. Each traced stream is created without filtering any event type: all events are traced.

Note: By default, no trace events are filtered.

Filtering non-relevant information maintains the performance of the tracing subsystem. It prevents the tracing subsystem from processing a large number of events while the trace collection is generated or while the trace is analyzed.

The filtered events are gathered in a set of events (see “Trace event sets” on page 10). The set of events to be filtered out is attached to a stream: it has to be defined after the creation of the stream, but the stream may be either started or not.

With the posix_trace_set_filter() subroutine, the filtering set can be changed accordingly to the following values of the how parameter:

POSIX_TRACE_SET_EVENTSET
: The set of trace event types to be filtered is the trace event type set that the *set* parameter points to.

POSIX_TRACE_ADD_EVENTSET
: The set of trace event types to be filtered is the union of the current set and the trace event type set that the *set* parameter points to.

POSIX_TRACE_SUB_EVENTSET
: The set of trace event types to be filtered is the current trace event type set less each element of the specified set.

The system trace event POSIX_TRACE_FILTER indicates that the trace event filter set has changed while the trace stream was running. The trace event filter is managed by the controller process.

Table 1-6 lists the subroutines used to manage the filter set on the trace stream.

Table 1-6 Filter management routines on trace stream

Purpose	Subroutine name
Retrieves the filter of an initialized trace stream.	posix_trace_get_filter()
Sets the filter of an initialized trace stream.	posix_trace_set_filter()

Managing trace events

The results of the tracing operations are monitored and analyzed by the controller process and the analyzer process.

Table 1-7 lists the subroutines to manage trace events from a trace stream used by the trace controller and analyzer process.

Table 1-7 Management trace events routines used by controller and analyzer

Purpose	Subroutine name
Compares two trace event type identifiers.	posix_trace_eventid_equal()
Retrieves the trace event name from a trace event type identifier.	posix_trace_eventid_get_name()
Iterates over the list of trace event type.	posix_trace_eventtypelist_getnext_id()
Rewinds the list of event types.	posix_trace_eventtypelist_rewind()

Table 1-8 lists the subroutines to retrieve *trace events* from a trace stream used by the trace analyzer process.

Table 1-8 Retrieval trace events routines used by the analyzer process

Purpose	Subroutine name
Retrieves a trace event and block until available.	posix_trace_getnext_event()
Retrieves a trace event and block until the timeout expires.	posix_trace_timedgetnext_event()
Retrieves a trace event and returns if not available.	posix_trace_trygetnext_event()

1.3.3 Trace stream definition

A trace stream is the location where trace events are recorded. The following are the types of streams and objects, as noted by the POSIX standard:

The *active* stream The active stream is an initialized and created trace stream that is still not shutdown. The trace stream can still store trace events. As a trace stream can be located only in memory, if an analysis must be done after process execution, a log file has to be defined at the creation time of the trace stream.

The *Log* file The log file is a persistent location where the in-memory trace stream is written by a flush operation initiated by the controller process. No stored events can be retrieved directly from a log file. A log file is available for analysis only after the corresponding trace stream has been shut down.

Without a *Log* file Without a log file, a trace stream allows only online analysis.

The *pre-recorded* stream As stored events in a log file cannot be directly retrieved, they have to be re-loaded in a trace stream. This trace stream is named *pre-recorded stream*. Then the analyzer process doing the analysis can retrieve the traced events from this pre-recorded stream.

The *Event recording* The events are recorded in the stream as soon as the stream is started. The stream may be associated with a log file if any offline analysis is needed. The association of the stream with the log file is made at the stream creation. The log file is a persistent location where the in-memory trace is flushed by the controller process.

The *Event analysis* When the stream is not associated to a log file, the stream allows only online analysis. The log file is ready for an analysis as soon as the stream associated with a log file has been shut down. That means that no stored events can be retrieved for the analysis during the event recording. The stored events are re-loaded from the log file into a trace stream. Events are then retrieved as during online analysis.

Traced events have to be retrieved one by one from the traced stream (*active* or *pre-recorded*) with the oldest event being retrieved first. With AIX, trace stream is an in-memory area where trace events are recorded.

> **Note:** Trace analysis can be done concurrently while tracing the instrumented code or it can be done offline. Log files are not directly eligible for trace analysis: they must be reloaded into a stream.

Whatever it is, a trace stream or a trace log, an action policy has to be defined when the trace stream or the trace log will be full of traced events. These *full* policies are named respectively *trace stream policy* (see “Trace stream policy” on page 15) and *trace Log policy* (see “Trace log policy” on page 15).

A trace stream or trace log capacity to record events depends on numerous criteria as the size of stream/Log, the size of the recorded events, and the number of the recorded events named *inheritance*: either only the process events or the process and its child processes events are recorded. All these criteria, jointly with the full policies, are gathered into the attributes definition of a traced stream (see “Trace stream attributes” on page 16).

Selecting the types of events to be recorded also determines how fast the traced stream/log will be full (see “Trace stream filtering” on page 11).

Trace stream policy

The *stream policy* is one of the trace stream attributes. The stream attributes are described in “Trace stream attributes” on page 16.

The stream policy, also named *stream full policy*, defines the policy followed when the trace stream is full and has the following values:

POSIX_TRACE_LOOP
This policy permits automatic overwrite of the oldest events until the trace is stopped by the subroutines posix_trace_stop() or posix_trace_shutdown().

POSIX_TRACE_UNTIL_FULL
This policy requires the system to stop tracing when the trace stream is full. If the stream that is full is emptied by a call to posix_trace_flush() or partially emptied by calls to posix_trace_getnext_event(), the trace activity is resumed.

POSIX_TRACE_FLUSH
This policy is an extension of the previous policy POSIX_TRACE_UNTIL_FULL for trace stream associated to a log file. There is an automatic flush operation when the stream is full.

Trace log policy

The *log policy* is one of the trace stream attributes. The stream attributes are described in “Trace stream attributes” on page 16.

The log policy, also named *log full policy*, defines the policy followed when the trace log is full and has the following values:

POSIX_TRACE_LOOP
The trace log loops until the trace stream is stopped by the subroutines posix_trace_stop() or posix_trace_shutdown(). This policy permits automatic overwriting of the oldest events.

POSIX_TRACE_UNTIL_FULL
The trace stream is flushed to the trace log until the trace log is full. The last recorded trace event is the POSIX_TRACE_STOP trace event (see "System trace events" on page 10). The event collection stops when the trace stream or the trace log file becomes full.

POSIX_TRACE_APPEND
The trace stream is flushed to the trace log without log size limitation.

Trace stream attributes

A trace stream has the following *trace stream attributes*:

Version of the trace system
The generation-version attribute identifies the origin and version of the trace system. It is generated automatically by the trace system.

Name of the trace stream
A character string to identify the trace stream, defined by the trace controller.

Creation time The time of creation of the trace stream. It is generated automatically by the trace system.

Clock resolution The clock resolution of the clock used to generate time stamps. It is generated automatically by the trace system.

Stream_minsize The minimal size in bytes of the trace stream strictly reserved for the trace events. The maximum size has been set to a segment size.

Stream_fullpolicy The policy followed when the trace stream is full; it could be either to loop at the beginning of the stream or to stop tracing or to flush to a log file when it is full.

Max_datasize The maximum record size in bytes for a trace event. Traced data exceeding that limit will be recorded up to that limit.

Inheritance It specifies whether a newly created trace stream inherits tracing in its parent's process trace stream or not. It specifies either if the parent is being traced or if its child is concurrently traced using the same stream (POSIX_TRACE_INHERITED) or not (POSIX_CLOSE_FOR_CHILD).

Log_maxsize The maximum size in bytes of a trace log associated with an active stream.

Log_fullpolicy It defines the policy of a trace log associated with an active trace stream; it could be either loop, tracing until the log is full, or tracing until the maximum size defined for a file system is reached.

Before the trace stream is created, the *trace stream attributes*, contained in the trace_attr_t object must be initialized by the posix_trace_attr_init() subroutine.

This posix_trace_attr_init() subroutine initializes the trace stream attributes with the default values described in Table 1-9.

Table 1-9 Default values for trace stream attributes

Attribute field	Default value
stream_minsize	8192 bytes. This is the smallest AIX trace buffer size.
stream_fullpolicy	POSIX_TRACE_LOOP for a stream without a log POSIX_TRACE_FLUSH for a stream with a log
max_datasize	16 bytes
inheritance	POSIX_TRACE_CLOSE_FOR_CHILD
log_maxsize	1 MB
log_fullpolicy	POSIX_TRACE_LOOP
version	0.1
clock resolution	Clock resolution used to generate time stamps

The value of each attribute is set by calling posix_trace_attr_set...() subroutines that explicitly set the value of these attributes (see Table 1-10).

The value of each attribute is retrieved from this trace_attr_t object using the posix_trace_attr_get...() subroutines (see Table 1-11 on page 18).

Table 1-10 lists the subroutines used to set up and manage the *trace stream attributes* object by the controller process.

Table 1-10 Setting trace stream attribute routines used by the controller process

Purpose	Subroutine name
Initializes a trace stream attributes object.	posix_trace_attr_init()
Destroys a trace stream attribute object.	posix_trace_attr_destroy()
Sets the trace name.	posix_trace_attr_setname()

Purpose	Subroutine name
Sets the inheritance policy of a trace stream.	posix_trace_attr_setinherited()
Sets the stream full policy.	posix_trace_attr_setstreamfullpolicy()
Sets the maximum user trace event data size.	posix_trace_attr_setmaxdatasize()
Sets the trace stream size.	posix_trace_attr_setstreamsize()
Sets the size of the log of a trace stream.	posix_trace_attr_setlogsize()
Sets the log full policy of a trace stream.	posix_trace_attr_setlogfullpolicy()

Table 1-11 lists the subroutines used to retrieve the trace stream attributes used by the trace controller and analyzer process.

Table 1-11 Retrieval trace stream attribute routines used by the controller and analyzer

Purpose	Subroutine name
Retrieves the timestamping clock resolution.	posix_trace_attr_getclockres()
Retrieves the creation time of a trace stream.	posix_trace_attr_getcreatetime()
Retrieves the version of a trace stream.	posix_trace_attr_getgenversion()
Retrieves the inheritance policy of a trace stream.	posix_trace_attr_getinherited()
Retrieves the log full policy of trace stream.	posix_trace_attr_getlogfullpolicy()
Retrieves the size of the log of a trace stream.	posix_trace_attr_getlogsize()
Retrieves the maximum user trace event data size.	posix_trace_attr_getmaxdatasize()
Retrieves the maximum size of a system trace event.	posix_trace_attr_getmaxsystemeventsize()
Retrieves the maximum size of an user event for a given length.	posix_trace_attr_getmaxusereventsize()
Retrieves the trace stream name.	posix_trace_attr_getname()

Purpose	Subroutine name
Retrieves the stream full policy.	posix_trace_attr_getstreamfullpolicy()
Retrieves the trace stream size.	posix_trace_attr_getstreamsize()

Trace stream management

The trace stream is created for the traced process with the posix_trace_create() or posix_trace_create_withlog() subroutine by the controller process, depending on whether a log is associated with the active stream or with posix_trace_open() by the analyzer process.

These trace stream creation subroutines use the process identifier (pid_t type) of the traced process as an argument: a zero indicates the traced process is the caller itself.

A *trace stream identifier* is returned by the trace stream creation routines and is valid only for the process that made these calls. The *trace stream identifier* is defined as the trace_id_t type in the /usr/include/sys/types.h include file.

Table 1-12 lists the subroutines to retrieve the attribute and state of the trace stream used by the trace controller and analyzer process.

Table 1-12 Trace stream attributes and state routines

Purpose	Subroutine name
Retrieves trace attributes.	posix_trace_get_attr()
Retrieves trace status.	posix_trace_get_status()

Table 1-13 lists the subroutines to control the *trace stream* used by the trace controller process.

Table 1-13 Trace stream control routines used by the trace controller process

Purpose	Subroutine name
Creates an active trace stream.	posix_trace_create()
Creates an active trace stream and associates it with a trace log.	posix_trace_create_withlog()
Initiates a flush of the trace stream.	posix_trace_flush()
Shuts down a trace stream.	posix_trace_shutdown()
Clears the trace stream and trace log.	posix_trace_clear()

Purpose	Subroutine name
Starts a trace.	posix_trace_start()
Stops a trace.	posix_trace_stop()

Table 1-14 lists the subroutines to control the *trace stream* used by the trace analyzer process.

Table 1-14 Trace stream control routines used by the trace analyzer process

Purpose	Subroutine name
Opens a trace log.	posix_trace_open()
Re-initializes a trace log for reading.	posix_trace_rewind()
Closes a trace log.	posix_trace_close()

1.3.4 AIX implementation overview

With AIX Version 6, the process that manages streams and events is a daemon named *posixtrace*. It is the only process the operating system has to implement.

As posixtrace creates a a trace stream for all processes and records all events, posixtrace belongs to the root user. The posixtrace daemon is run as root (owner: root group:bin mode -r-sr-xr-x).

The posixtrace daemon is started by the first library load through the associated library initialization routine mechanism. This mechanism is implemented through the binitfini binder option. Thus, the libposixtrace.a library has been linked with the option -binitfini:posix_trace_libinit.

This posix_trace_libinit routine binds a dedicated socket to the file named /var/adm/ras/.pxt_sock and listens for one connection coming from the instrumented code linked with the libposixtrace library.

Another file named /var/adm/ras/.start_lock is used as a lock file in order to prevent several starts of the posixtrace daemon.

When the main daemon thread checks that there is no thread left, it closes the socket, unlocks, and unlinks /var/adm/ras/.pxt_sock, then exits.

1.4 ProbeVue

The first *dynamic* tracing facility, named *ProbeVue*, is introduced with AIX with Version 6.

A tracing facility is *dynamic* because it is able to gather execution data from applications without any modification of their binaries or their source code. *Dynamic* refers to this capability to insert trace points at runtime without the need to prepare the source code in advance. Inserting specific tracing calls and defining specific tracing events into the source code, which require you to re-compile the software and generate new executable, is referred as a *static* tracing facility.

The name *ProbeVue* is given by historical reference to the first dynamic tracing facility introduced by IBM within the OS/2® operating system in 1994 (using the OS/2 **dtrace** command). This dynamic tracing facility was ported to Linux and expanded under the DProbes name. There is no other similarity between these two dynamic tracing tools: they remain two different and distinct tracing frameworks that come from a similar background.

Interestingly, there are no standards in the area of dynamic tracing. POSIX has defined a tracing standard for static tracing software only, as described in 1.3.1, “POSIX tracing overview” on page 6.

Dynamic tracing benefits and considerations

Software debugging is often considered a dedicated task running on development systems or test systems trying to mimic real customer production systems.

However, this general state is currently evolving due to the recent advances in hardware capabilities and software engineering creating complex environments:

- The processing and memory capabilities of high-end servers with associated storage technologies have lead to huge systems being put into production.
- Dedicated solutions developed by system integrators based on ERP software, for example, implement numerous middleware and several application layers and lead also to complex software solutions.
- Most software is now multi-threaded and running on many processors. Thus, two executions can behave differently depending on the order of thread execution: multi-threaded applications are generally non deterministic. Erroneous behaviors are more difficult to reproduce and debug for such software.

Thus, to determine the root cause of a trouble in today's IT infrastructure, it has become a prohibitive high expense and a significant burden if troubleshooting is not achieved on the real production system.

With the ProbeVue dynamic tracing facility, a production system can be investigated: ProbeVue captures the execution data without installing dedicated instrumented versions of applications or the kernel, which require interrupting the service for the application relaunch or server reboot.

Additionally, ProbeVue helps find the root cause of troubles happening only on long running jobs where unexpected accumulated data, queues overflows, and others defects of the application or kernel are revealed only after many days or months of execution.

As ProbeVue is able to investigate any kind of applications as long as a Probe Manager is available (see "Probe manager" on page 28), it is a privileged tracing tool to analyze a complex trouble as a cascading failure between multiple sub-systems: with only one unique tracing tool, ProbeVue allows an unified instrumentation of a production system.

Of note, ProbeVue has the following considerations:

- To trace an executable without modifying it requires you to encapsulate the binary code with a control execution layer. This control layer will start and interrupt the binary execution to allow the context tracing. Due to the dynamic tracing aspect, it can only be an interpreted layer. Interpreter languages are known to be slower than compiled language: the dynamic interpreted tracing points are potentially slower than the static compiled ones.
- If system administrators and system integrators are expected to use a tool to investigate the software execution, the tool must give them the necessary knowledge of the application architecture to do an efficient investigation of the critical components that are in trouble. On the other hand, developers know where to set effective tracing points on the strategic data manipulated by the application on the earlier development stage, so this is more effective.

For these reasons, ProbeVue is a complimentary tracing tool to the static tracing methods, adding a new innovative tracing capability to running production systems.

ProbeVue dynamic tracing benefits

As a dynamic tracing facility, ProbeVue has the following main benefits:

- Trace hooks do not have to be pre-compiled. ProbeVue works on unmodified kernel or user applications.

- The trace points or probes have no effect (do not exist) until they are dynamically enabled.
- Actions (specified by the instrumentation code) to be executed at a probe point or the probe actions are provided dynamically at the time the probe is enabled.
- Trace data captured as part of the probe actions are available for viewing immediately and can be displayed as terminal output or saved to a file for later viewing.

ProbeVue can be used for performance analysis as well as for debugging problems. It is designed to be safe to run on production systems and provides protection against errors in the instrumentation code.

The section defines some of the terminology used. The subsequent sections introduce Vue, the programming language used by ProbeVue and the **probevue** command, which is used to start a tracing session.

1.4.1 ProbeVue terminology

ProbeVue introduces a terminology for the concepts used in dynamic tracing. The following is the description of the terms used with ProbeVue:

Probe
A software mechanism that interrupts normal system action to investigate and obtain information about current context and system state. This is also commonly referred to as *tracing*.

Tracing actions or **probe actions**
Refers to the actions performed by the probe. Typically, they include the capturing of information by dumping the current values of global and context-specific information to a trace buffer. The obtained information, thus captured in the trace buffer, is called *trace data*. The system usually provides facilities to consume the trace, that is, read the data out of the trace buffer and make it available to the users of the system.

A probe point
Identifies the points during normal system activity that are capable of being probed. With dynamic tracing, probe points do not have any probes installed in them unless they are being probed.

Enabling a probe is the operation of adding a probe to a probe point.

Disabling a probe is the operation of removing a probe from a probe point.

Triggering or **firing** of a probe refers to the condition where a probe is entered and the tracing actions are performed.

ProbeVue supports two kinds of probe points:

Probe location This is a location in user or kernel code where some tracing action like the capture of trace data is to be performed. Enabled probes at a probe location fire when any thread executing code reaches that location.

Probe event This is an abstract event at whose occurrence some tracing action is to be performed. Probe events do not easily map to a specific code location. Enabled probes that indicate a probe event fire when the abstract event occurs.

ProbeVue also distinguishes probe points by their type:

Probe type Identifies a set of probe points that share some common characteristics, for example, probes that, when enabled, fire at the entry and exit of system calls, or probes that when enabled fire when system statistics are updated.

Distinguishing probes by probe types induces a structure to a wide variety of probe points. So, ProbeVue requires a probe manager to be associated with each probe type:

Probe manager The software code that defines and provides a set of probe points of the same probe type, for example, "*the system calls*" probe manager.

1.4.2 Vue programming language

The Vue programming language is used to provide your tracing specifications to ProbeVue. The Vue programming language is often abbreviated to the *Vue language* or just to *Vue*.

A Vue script or Vue program is a program written in Vue. You can use a Vue script to:

- Identify the probe points where a probe is to be dynamically enabled.
- Identify the conditions, if any, which must be satisfied for the actions to be executed when a probe fires.

- Identify the actions to be executed, including what trace data to capture.
- Associate the same set of actions for multiple probe points.

In short, a Vue script tells ProbeVue where to trace, when to trace, and what to trace.

We recommend that Vue scripts have a file suffix of *.e* to distinguish them from other file types, although this is not a requirement.

1.4.3 The probevue command

The **probevue** command is used to start a dynamic tracing session or a ProbeVue session. The **probevue** command takes a Vue script as input, reading from a file or from the command line and activates a ProbeVue session. Any trace data that is captured by the ProbeVue session can be printed to the terminal or saved to a user-specified file as per options passed in the command line.

The ProbeVue session stays active until a Ctrl-C is typed on the terminal or an exit action is executed from within the Vue script.

Each invocation of the **probevue** command activates a separate dynamic tracing session. Multiple tracing sessions may be active at one time, but each session presents only the trace data that is captured in that session.

Running the **probevue** command is considered a privileged operation and privileges are required for non-root users who wish to initiate a dynamic tracing session.

1.4.4 The probevctrl command

The **probevctrl** command changes and displays the ProbeVue dynamic tracing parameters, the per-processor trace buffer size, the consumed pinned memory, the user owning the session, the identifier of the process that started the session, and the information about whether the session has kernel probes for the ProbeVue sessions.

1.4.5 Vue: an overview

Vue is both a programming and a script language. It is not an extension of C or a simple mix of C and **awk**. It has been specifically designed as a dedicated dynamic tracing language. Vue supports a subset of C and scripting syntax that is most beneficial for dynamic tracing purposes.

This section describes the structure of a Vue script.

Structure of a Vue script

A Vue script consists of one or more clauses. The clauses in a Vue script can be specified in any order. Figure 1-3 is a typical layout of a Vue script.

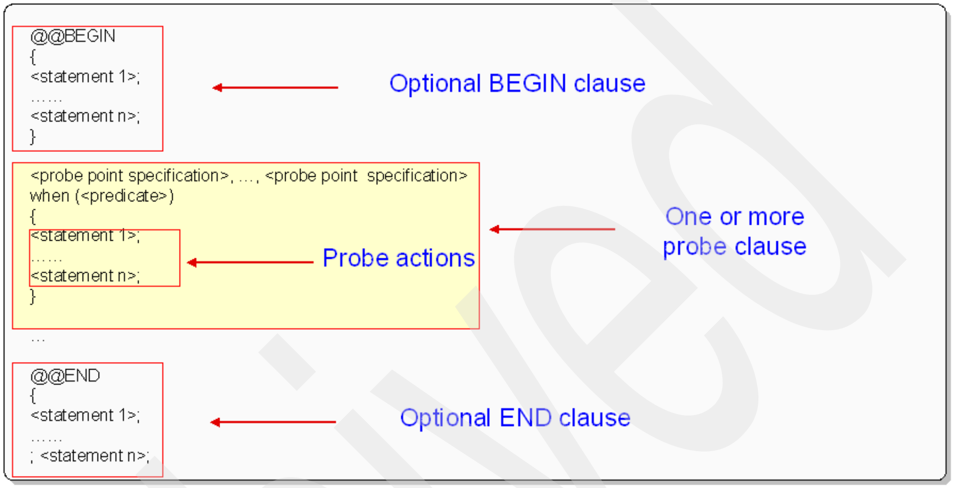


Figure 1-3 Structure of a Vue script

The following are two Vue scripts examples:

1. The following canonical Hello World program prints "Hello World" into the trace buffer and exits:

```
#!/usr/bin/probevue

   /* Hello World in probevue */
   /* Program name: hello.e */

   @@BEGIN
   {
      printf("Hello World\n");
      exit();
   }
```

2. The following Hello World program prints "Hello World" when Ctrl-C is typed on the keyboard:

```
#!/usr/bin/probevue

   /* Hello World 2 in probevue */
   /* Program name: hello2.e */
```

```
@@END
{
   printf("Hello World\n");
}
```

Each clause of a Vue script consists of the following three elements:

- Probe point specification

  The probe point specification identifies the probe points to be dynamically enabled.

- Action Block

  The action block is used to identify the set of probe actions to be performed when the probe fires.

- An optional predicate

  The predicate, if present, identifies a condition that is to be checked at the time the probe is triggered. The predicate must evaluate to TRUE for the probe actions of the clause to be executed.

Probe point specification

A probe point specification identifies the code location whose execution or the event whose occurrence should trigger the probe actions. Multiple probe points can be associated with the same set of probe actions and the predicate, if any, by providing a comma-separated list of probe specifications at the top of the Vue clause.

The format for a probe specification is probe-type specific. The probe specification is a tuple (a type of programming structure) of ordered list of fields separated by colons. It has the following general format:

```
@@<probetype>:<probetype field1>:...:<probetype fieldn>:<location>
```

AIX Version 6.1 supports the following probe types:

- User Function Entry probes (or uft probes)

  For example, a uft probe at the entry into any function called foo() (in the main executable or any of the loaded modules including libraries) in process with ID = 34568:

  ```
  @@uft:34568:*:foo:entry
  ```

- System Call Entry/Exit probes (or syscall probes)

  For example, a syscall probe at the exit of a read system call:

  ```
  @@syscall:*:read:exit
  ```

- Probes that fire at specific time intervals (or interval probes)

  For example, an interval probe that fires every 500 milliseconds (wall clock time):

  ```
  @@interval:*:clock:500
  ```

Action blocks

The action block identifies the set of actions to be performed when a thread hits the probe point. Supported actions are not restricted to the basic capturing and formatting of trace data but, the full power of Vue can be employed.

An action block in Vue is similar to a procedure in procedural languages. It consists of a sequence of statements that are executed in order. The flow of execution is essentially sequential. The only exceptions are that conditional execution is possible using the if-else statement and control may be returned from within the action block using the return statement.

Unlike procedures in procedural languages, an action block in Vue does not have an output or return value. Neither does it have inherent support for a set of input parameters. On the other hand, the context data at the point where a probe is entered can be accessed within the action block to parameterize the actions to be performed.

Predicates

Predicates should be used when execution of clauses at probe points must be performed conditionally.

The predicate section is identified by the presence of the when keyword immediately after the probe specification section. The predicate itself consists of regular C-style conditional expressions with the enclosing parentheses.

A predicate has the following format:

```
when ( <condition> )
```

For example, this is a predicate indicating that probe points should be executed for process ID = 1678:

```
when ( __pid == 1678 )
```

Probe manager

The probe manager is an essential component of dynamic tracing. Probe managers are the providers of the probe points that can be instrumented by ProbeVue.

Probe managers generally support a set of probe points that belong to some common domain and share some common feature or attribute that distinguishes them from other probe points. Probe points are useful at points where control flow changes significantly, at points of state change, or other similar points of significant interest. Probe managers are careful to select probe points only in locations that are safe to instrument.

ProbeVue currently supports the following three probe managers:

- System call (syscall) probe manager

  The syscall probe manager supports probes at the entry and exit of well-defined and documented base AIX system calls. The syscall probe manager accepts a 4-tuple probe specification in one of the following formats where the *<system_call_name>* field is to be substituted by the actual system call name:

  ```
  * syscall:*:<system_call_name>:entry
  * syscall:*:<system_call_name>:exit
  ```

  These indicate that a probe is to be placed at the entry and exit of system calls. Assigning the "*" to the second field indicates that the probe will be fired for all processes. Additionally, a process ID can be specified as the second field of the probe specification to support probing of specific processes:

  ```
  * syscall:<process_ID>:<system_call_name>:entry
  * syscall:<process_ID>:<system_call_name>:entry
  ```

- User function probe manager

  The user function tracing (uft) probe manager supports probing user space functions that are visible in the XCOFF symbol table of a process. These entry points, usable as probe points, are currently restricted to those written in C language text file. The uft probe manager currently accepts a 5-tuple probe specification only in the following format:

  ```
  uft:<processID>:*:<function_name>:entry
  ```

  Note that the uft probe manager requires the process ID for the process to be traced and the complete function name of the function at whose entry point the probe is to be placed. Further, the uft probe manager currently requires that the third field be set to '*" to indicate that the function name is to be searched in any of the modules loaded into the process address space, including the main executable and shared modules.

- Interval probe manager

  The interval probe manager supports probe points that fire at a user-defined time interval. The probe points are not located in kernel or application code, but instead are based on wall clock time interval based probe events. The interval probe manager accepts a 4-tuple probe specification in the following format:

  ```
  @@interval:*:clock:<# milliseconds>
  ```

  The second field is *, indicating that the probe can be fired in any process. Currently, the interval probe manager does not filter probe events by process IDs. For the third field, the only value supported currently is the clock keyword that identifies the probe specification as being for a wall clock probe. The fourth or last field, that is, the <# milliseconds> field, identifies the number of milliseconds between firings of the probe. Currently, the interval probe manager requires that the value for this field be exactly divisible by 100 and consist only of digits 0-9. Thus, probe events that are apart by 100 ms, 200 ms, 300 ms, and so on, are allowed.

Vue functions

Unlike programs written in the C or FORTRAN programming languages or in a native language, scripts written in Vue do not have access to the routines provided by the AIX system libraries or any user libraries. However, Vue supports its own special library of functions useful for dynamic tracing programs. Functions include:

- Tracing-specific functions:

get_function	Returns the name of the function that encloses the current probe.
time stamp	Returns the current time stamp.
diff_time	Finds the difference between two time stamps.

- Trace capture functions

printf	Formats and prints values of variables and expressions.
trace	Prints data without formatting.
stktrace	Prints and formats the stack trace.

- List functions

list	Instantiate a list variable.
append	Append a new item to a list.

sum, max, min, avg, count
Aggregation functions that can be applied on a list variable.

- C-library functions

atoi, strstr Standard string functions.

- Functions to support tentative tracing

start_tentative, end_tentative
Indicators for start and end of tentative tracing.

commit_tentative, discard_tentative
Commit or discard data in tentative buffer.

- Miscellaneous functions

exit Terminates the E-program.

get_userstring Read string from user memory.

The Vue string functions can be applied only on variables of string type and not on a pointer variable. Standard string functions like strcpy(), strcat(), and so on, are not necessary in Vue, because they are supported through the language syntax itself.

1.4.6 ProbeVue dynamic tracing example

This is a basic ProbeVue example to show how ProbeVue works and how to use ProbeVue on a running executable without restarting or recompiling it.

The following steps must be performed:

1. The C program shown in Example 1-1, named *pvue*, is going to be traced dynamically.

Example 1-1 Basic C program to be dynamically traced: pvue.c

```
#include <fcntl.h>
main()
{
int x, rc;
int buff[100];

for (x=0; x<5; x++){
        sleep(3);
        printf("x=%d\n",x);
}
sleep (3);
```

```
fd=open("./pvue.c",O_RDWR,0);
x =read(fd,buff,100);
printf("[%s]\n",buff);
}
```

2. Compile and execute the program in the background. For example:

```
# cc -q64 -o pvue pvue.c
# ./pvue &
[1]     262272
```

3. In order to trace dynamically the number of calls executed by the pvue process to the subroutines printf(), sleep(), entry of read(), exit of read(), we use the probevue script shown in Example 1-2, named pvue.e, which uses the process ID as an entry parameter ('$1').

Example 1-2 Sample Vue script, named pvue.e

```
#!/usr/bin/probevue
@@BEGIN
{
        printf("Tracing starts now\n");
}
@@uft:$1:*:printf:entry
{
        int count;
        count = count +1;
        printf("printf called %d times\n",count);
}
@@uft:$1:*:sleep:entry
{
        int count1;
        count1 = count1 +1;
        printf("sleep called %d times\n",count1);
}
@@syscall:*:read:exit
        when (__pid == $1)
{
        printf("read entered\n");
}
@@syscall:*:read:entry
        when (__pid == $1)
{
        printf("read exited\n");
}
@@END
{
```

```
        printf("Tracing ends now\n");
}
```

4. We use the Vue script named *pvue.e,* with the process ID to be traced as the parameter, by executing the **probevue** command:

```
# probevue ./pvue.e  262272
```

Example 1-3 shows the tracing output.

Example 1-3 Start Vue script providing pid

```
# ./pvue.e  262272
Tracing starts now
printf called 1 times
sleep called 1 times
printf called 2 times
sleep called 2 times
printf called 3 times
sleep called 3 times
printf called 4 times
sleep called 4 times
printf called 5 times
sleep called 5 times
read exited
read entered
printf called 6 times
^CTracing ends now
#
```

2

File systems and storage

This chapter contains the major AIX Version 6.1 enhancements that are part of the file system and connected storage, including:

2.1 Disabling JFS2 logging

AIX V6.1 allows you to mount a JFS2 file system with logging turned off. Disabling JFS2 logging can increase I/O performance. The following examples are typical situations where disabled logging may be helpful:

- While restoring a backup
- For a compiler scratch space
- During a non-migration installation

Improved performance is also found in situations where a series of I/O operations modify JFS2 metadata. Note that non-representative tests in a lab environment showed up to a ten percent performance improvement for a series of operations that *only* changed JFS2 metadata.

Be sure to balance the benefit of a performance advantage with the possible data exposures of a disabled file system log.

> **Important:** If a system abnormally stops during a JFS2 metadata operation with logging disabled, the **`fsck`** command might not be able to recover the file system into a consistent state. In such cases, the file system has to be recreated, and all the data will be lost.

You can disable JFS2 logging with the **`mount`** command. There is no SMIT or Web-based System Manager panel, since this feature is used only in rare cases. You cannot disable the logging while creating a file system. Every file system has to be created with a valid JFS2 log device or an inline log.

Use the following flag with the **`mount`** command to mount a JFS2 file system with logging disabled:

```
mount -o log=NULL /aix61diff
```

In order to make the mount setting persistent, modify the log attribute of the corresponding /etc/filesystems stanza to log=NULL.

2.2 JFS2 internal snapshot

With AIX 5L V5.2, the JFS2 snapshot was introduced. Snapshots had to be created into separate logical volumes. AIX V6.1 offers the ability to create snapshots within the source file system.

Therefore, starting with AIX V6.1, there are two types of snapshots:

- External snapshot
- Internal snapshot

Table 2-1 provides an overview of the differences between the two types of snapshots.

Table 2-1 Comparison of external and internal snapshots

Category	External snapshot	Internal snapshot
Location	Separate logical volume	Within the same logical volume
Access	Must be mounted separately	/fsmountpoint/.snapshot/snapshotname
Maximum generations	15	64
AIX compatibility	>= AIX 5L V5.2	>= AIX V6.1

Both the internal and the external snapshots keep track of the changes to the snapped file system by saving the modified or deleted file blocks. Snapshots provide point-in-time (PIT) images of the source file system. Often, snapshots are used to be able to create a consistent PIT backup while the workload on the snapped file system continues.

The internal snapshot introduces the following enhancements:

- No super user permissions are necessary to access data from a snapshot, since no initial mount operation is required.
- No additional file system or logical volume needs to be maintained and monitored.
- Snapshots are easily NFS exported, since they are in held in the same file system.

2.2.1 Managing internal snapshots

A JFS2 file system must be created with the new -a isnapshot=yes option. Internal snapshots require the use of the extended attributes v2 and therefore the **`crfs`** command will automatically create a v2 file system.

Existing file systems created without the isnapshot option cannot be used for internal snapshots. They have to be recreated or have to use external snapshots.

There are no new commands introduced with internal snapshots. Use the **snapshot**, **rollback**, and **backsnap** commands to perform operations. Use the new -n snapshotname option to specify internal snapshots. There are corresponding SMIT and Web-based System Manager panels available.

To create an internal snapshot:

```
# snapshot -o snapfrom=/aix61diff -n snap01
Snapshot for file system /aix61diff created on snap01
```

To list all snapshots for a file system:

```
# snapshot -q /aix61diff
Snapshots for /aix61diff
Current  Name        Time
*        snap01      Tue Sep 25 11:17:51 CDT 2007
```

To list the structure on the file system:

```
# ls -l /aix61diff/.snapshot/snap01
total 227328
-rw-r--r--    1 root     system     10485760 Sep 25 11:33 file1
-rw-r--r--    1 scott    staff       1048576 Sep 25 11:33 file2
-rw-r--r--    1 jenny    staff     104857600 Sep 25 11:33 file3
drwxr-xr-x    2 root     system          256 Sep 24 17:57 lost+found
```

The previous output shows:

- All snapshots are accessible in the /fsmountpoint/.snapshot/ directory.
- The data in the snapshot directories are displayed with their original file permission and ownership. The files are read only; no modifications are allowed.

> **Note:** The .snapshot directory in the root path of every snapped file system is not visible to the **ls** and **find** command. If the .snapshot directory is explicitly specified as an argument, they are able to display the content.

To delete an internal snapshot:

```
# snapshot -d -n snap01 /aix61diff
```

2.2.2 Error handling

There are two known conditions where a snapshot is unable to preserve the file system data:

- The file system runs out of space (for internal snapshots) or the logical volume is full (for external snapshots).
- Write operations to the snapshot are failing, for example, due to a disk failure.

In both cases, all snapshots are aborted and marked as INVALID. In order to recover from this state, the snapshots have to be deleted and a new one can be created. It is, therefore, important that you monitor the usage of the file system or logical volume:

- You can use the **`snapshot -q`** command and monitor the Free field for logical volumes of external snapshots that are not mounted.
- For internal snapshots, use the **`df`** command to monitor the free space in the file system.

If an error occurs while reading data from a snapshot, an error message is returned to the running command. The snapshot is still valid and continues to track changes to the snapped file system.

2.2.3 Considerations

The following applies for internal snapshots:

- A snapped file system can be mounted read only on previous AIX 5L versions. The snapshot itself cannot be accessed. The file system must be in a clean state; run the **`fsck`** command to ensure that this is true.
- A file system created with the ability for internal snapshots can still have external snapshots.
- Once a file system has been enabled to use internal snapshots, this cannot be undone.
- If the **`fsck`** command has to modify the file system, any internal snapshots for the file system will be deleted by **`fsck`**.
- Snapped file systems cannot be shrunk.
- The **`defragfs`** command cannot be run on a file system with internal snapshots.
- Existing snapshot Web-based System Manager and SMIT panels are updated to support internal snapshots.

The following items apply to both internal and external snapshots:

- A file system can use exclusively one type of snapshot at the same time.
- Typically, a snapshot will need two to six percent of the space needed for the snapped file system. For a highly active file system, 15 percent is estimated.
- External snapshots are persistent across a system reboot.
- During the creation of a snapshots, only read access to the snapped file system is allowed.
- There is reduced performance for write operations to a snapped file system. Read operations are not affected.
- Snapshots are not replacement for backups. A snapshot depends always on the snapped file system, while backups have no dependencies on the source.
- Neither the **`mksysb`** nor **`alt_disk_install`** commands will preserve snapshots.
- A file system with snapshots cannot be managed by DMAPI. A file system being managed by DMAPI cannot create a snapshot.

2.3 Encrypted File System

AIX V6.1 introduces the ability to encrypt files on a per file basis without the need of third-party tools. EFS should be used in environments where sensitive data requires additional protection.

AIX EFS has the following advantages over other encrypted file systems:

- Increased file level encryption granularity:

  Data is encrypted on a user/group level, compared to other implementations, where all users use the same keys. This is a useful protection on a per file system/disk level, but does not protect the data from being read by others in the same file system/disk.
- Seamless integration into traditional user administration commands and therefore transparent to users and administrators.
- Provides a unique mode that can protect against a compromised or malicious root user.

Additional information and extensive examples can be found in Chapter 2, “Encrypted File System”, in *AIX 6 Advanced Security Features: Introduction and Configuration*, SG24-7430:

`http://www.redbooks.ibm.com/abstracts/sg247430.html?Open`

2.3.1 Encryption

You can encrypt files on a per-file basis. Data is encrypted before it is written back to disk and decrypted after it is read from disk. Data held in memory is not encrypted, but the EFS access control is still in place. AIX uses a combination of symmetric and asymmetric encryption algorithms to protect the data.

A unique AES symmetric key is used to encrypt and decrypt every file. This symmetric key is encrypted with an RSA public key of the user and group and then added to the extended attributes of the file.

EFS uses an RSA private/public keypair to protect each symmetric key. These keys are stored in containers named *keystores*. The user keystores are password protected. The initial password of a user keystore is the user login password. Group keystores and admin keystores are not protected with a password; instead they have access key protection. Access keys are stored inside all user keystores that belong to this group.

The users keystore is loaded into the AIX kernel upon user login (associated with the login shell) or by invoking the new **`efskeymgr`** command and providing an argument to specify to which process the keys should be associated. All child processes of the associated process will have access to the keys.

2.3.2 Keystore modes

User keystores have two modes of operation, as discussed in the following sections.

Root admin mode

In root admin mode, the root user can:

- Get access to the user keystore
- Get access to the group keystore
- Reset the user keystore password
- Reset the group access key

Root admin mode is the default mode of operation. A consequence of root being able to get access to the user keystore is that root can get access to *all* encrypted files.

Root guard mode

All the privileges granted to root in the root admin mode are not valid in this mode.

This mode of operation offers protection against a malicious root user. It means that if the system is hacked and the hacker somehow manages to obtain root privilege, the hacker cannot have access to user or group keystores and therefore cannot have access to user encrypted files.

Important: If a user loses their keystore password, root cannot reset it. It means that *no one* can get access to that keystore anymore and the encrypted files owned by this user can no longer be decrypted.

2.3.3 File access permissions

It is important to understand that the traditional AIX file permissions do not overlap with the EFS mechanisms. EFS introduces another level of file access checking. The following steps are used when an encrypted file is being accessed:

1. The traditional file permissions are checked first.
2. Only if the check is passed will AIX continue to verify that only a user that has a private key that matches one of the public keys can gain access to the encrypted data.

If the traditional file permissions allow the user to read the file, but the user has no proper private key in his keystore, access is denied.

Note: Even the root user will not have access to all files as long as other users do not grant access to encrypted files with the following command:

```
efsmgr -a ./filename -u root
```

If the keystores are operated in root admin mode, the root user can load the private keys of other users to get access to all files.

2.3.4 Installation

This section discusses the prerequisites and commands used for the installation of EFS.

Prerequisites

In order to use EFS, you must meet the following prerequisites:

- The Crypto Library (CLiC) package clic.rte from the AIX V6.1 expansion pack must be installed.
- Role Based Access Control (RBAC) must be enabled.

- A JFS2 file system with the efs=yes option must be enabled.
- A JFS2 file system with the ea=v2 option must be enabled.

If necessary, use the **chfs** command to change the efs and ea options on previously created file systems. If you specify the efs option with the **crfs** or **chfs** command, it will automatically create or change the file system to use v2 extended attributes.

Commands

There are new commands introduced with EFS. All are part of the bos.rte.security package, which is installed by default in AIX. These commands are shown in Table 2-2.

Table 2-2 New EFS commands

Command	Description
/usr/sbin/efsenable	Prepares the system to use EFS. It creates the EFS administration keystore, the user keystore of the current user (root or an user with the RBAC role aix.security.efs), and the security group keystore in the /var/efs directory. This command needs to be executed only once on every AIX installation in order to use EFS.
/usr/sbin/efskeymgr	Dedicated to all key management operations needed by EFS.
/usr/sbin/efsmgr	Manages the file encryption and de-encryption.

Traditional commands have been modified to support EFS, as shown in Table 2-3.

Table 2-3 Commands modified for EFS

Commands	Enhancement
cp, **mv**	Moves/copies files from EFS <-> EFS and EFS <-> non-EFS file systems.
ls, **find**	Enabled to handle encrypted files.
backup, **restore**, **tar**, **pax**, **cpio**	Supports raw modes for EFS encrypted files. Files can be accessed in the encrypted form without the need for the private keys.
mkdir	Handles EFS inheritance.
mkuser, **chuser**, **mkgroup**, **chgroup**, **rmuser**, **rmgroup**	Enabled to modify the keystores and EFS user attributes.

Commands	Enhancement
chown, **chgrp**, **chmod**	Enabled to modify the EFS extended attributes.
passwd	Updates the key store password if it is the same as the login password.

For the new command options, refer to the man pages or the AIX product documentation.

2.3.5 Enable and create EFS file systems

This section describes the necessary steps to activate EFS. Example 2-1 shows the following tasks:

1. Enable EFS.
2. Create an EFS file system.
3. Shows the directory structure for the keystores.
4. Mount the file system.

All commands have to be run from the root user or a user with the appropriate RBAC roles assigned.

Example 2-1 Enabling EFS and creating an EFS file system

```
# efsenable -a
Enter password to protect your initial keystore:
Enter the same password again:

# crfs -v jfs2 -g rootvg -m /efs -A yes -a size=256M -a efs=yes
File system created successfully.
261932 kilobytes total disk space.
New File System size is 524288

# find /var/efs
/var/efs
/var/efs/users
/var/efs/users/.lock
/var/efs/users/root
/var/efs/users/root/.lock
/var/efs/users/root/keystore
/var/efs/groups
/var/efs/groups/.lock
/var/efs/groups/security
/var/efs/groups/security/.lock
```

```
/var/efs/groups/security/keystore
/var/efs/efs_admin
/var/efs/efs_admin/.lock
/var/efs/efs_admin/keystore
/var/efs/efsenabled

# mount /efs
```

2.3.6 File encryption and de-encryption

This section provides you an example of encrypting and decrypting files. Example 2-2 shows the following:

1. Display the loaded keys associated with the current login shell.
2. Create three test files.
3. Encrypt file2.
4. The **ls -U** command now indicates that the file is encrypted.
5. Use the **efsmgr -l** command to verify which keys are need to access the file.
6. Verify that user guest cannot read the file content that even the traditional file permissions would allow him to read.
7. Use the **ls**, **istat**, and **fsdb** commands to verify that the file is stored encrypted in the file system.
8. Decrypt file2.

Example 2-2 Encryption and de-encryption of files

```
# efskeymgr -V
List of keys loaded in the current process:
 Key #0:
                           Kind ..................... User key
                           Id   (uid / gid) ......... 0
                           Type ..................... Private key
                           Algorithm ................ RSA_1024
                           Validity ................. Key is valid
                           Fingerprint ..............
e34acd99:b1f22cdc:85f638e0:3fd56e78:e3c5a3a7
 Key #1:
                           Kind ..................... Group key
                           Id   (uid / gid) ......... 7
                           Type ..................... Private key
                           Algorithm ................ RSA_1024
                           Validity ................. Key is valid
```

```
                           Fingerprint .............
5e3e7305:203fce04:0e5a7339:4d688643:1e16beba
 Key #2:
                           Kind ..................... Admin key
                           Id   (uid / gid) ......... 0
                           Type ..................... Private key
                           Algorithm ................ RSA_1024
                           Validity ................. Key is valid
                           Fingerprint .............
fffa123f:cc615f5f:41b4dc2a:80e98a22:e50667a8

# cd /efs
# touch file1 file2 file3
# for i in file[1-3]
> do
> echo "content of $i" > $i
> done

# ls -U
total 24
-rw-r--r---    1 root     system           17 Sep 20 10:54 file1
-rw-r--r---    1 root     system           17 Sep 20 10:54 file2
-rw-r--r---    1 root     system           17 Sep 20 10:54 file3
drwxr-xr-x-    2 root     system          256 Sep 20 10:30 lost+found

# efsmgr -e file2

# ls -U
total 32
-rw-r--r---    1 root     system           17 Sep 20 10:54 file1
-rw-r--r--e    1 root     system           17 Sep 20 11:07 file2
-rw-r--r---    1 root     system           17 Sep 20 10:54 file3
drwxr-xr-x-    2 root     system          256 Sep 20 10:30 lost+found

# efsmgr -l file2
EFS File information:
 Algorithm: AES_128_CBC
List of keys that can open the file:
 Key #1:
  Algorithm       : RSA_1024
  Who             : uid 0
  Key fingerprint : e34acd99:b1f22cdc:85f638e0:3fd56e78:e3c5a3a7

# su - guest -c cat /efs/file[1-3]
content of file1
```

```
cat: 0652-050 Cannot open /efs/file2.
content of file3

#  ls -iU file2
    7 -rw-r--r--e    1 root     system          17 Sep 20 11:07 file2

# istat 7 /dev/fslv00
Inode 7 on device 10/11 File
Protection: rw-r--r--
Owner: 0(root)          Group: 0(system)
Link count:   1         Length 17 bytes

Last updated:   Thu Sep 20 11:07:09 CDT 2007
Last modified:  Thu Sep 20 11:07:09 CDT 2007
Last accessed:  Thu Sep 20 11:31:33 CDT 2007

Block pointers (hexadecimal):
2b

# fsdb /dev/fslv00
Filesystem /dev/fslv00 is mounted.  Modification is not permitted.

File System:                    /dev/fslv00

File System Size:               523864  (512 byte blocks)
Aggregate Block Size:           4096
Allocation Group Size:          8192    (aggregate blocks)

> display 0x2b
Block: 43     Real Address 0x2b000
00000000:  023173CC 00521DBD FDE0A433 556504CE   |.1s..R.....3Ue..|
00000010:  069AE78F 13610D78 7ECCB975 EDD9A258   |.....a.x~..u...X|
00000020:  F5E2DE6D AE16DEB9 4C9DF533 01F68EC1   |...m....L..3....|
00000030:  4A942ADA DD08A62D 86B3D4FF 0D7BA079   |J.*....-.....{.y|
00000040:  8A4A4D4E 3330F8B3 82640172 A830F7A4   |.JMN30...d.r.0..|
00000050:  85369398 10165D90 F57E1C90 023DD6E6   |.6....]..~...=..|
00000060:  9BAC97F3 AB308BA9 751AAA31 67167FFD   |.....0..u..1g...|
00000070:  11CDA7F1 BE590C7F D9E2C144 A0DFECE3   |.....Y.....D....|
00000080:  46B83CD8 01EB3133 1F1F2FAC 0E016BB0   |F.<...13../...k.|
00000090:  ED4055BA AA16D0F0 6BD1DEEA DE1D97ED   |.@U.....k.......|
000000a0:  BAC172E5 F4A0B05F 6DA06952 CC43D1F5   |..r...._m.iR.C..|
000000b0:  E023B89D E7F78E05 AB94246B 6602B394   |.#........$kf...|
000000c0:  3171B246 5C2AB5C7 B96CCF1E A78DE2BD   |1q.F\*...l......|
000000d0:  019C5735 AB71D7E8 12FB70F5 747F3DCA   |..W5.q....p.t.=.|
000000e0:  D1EA73FF 63746CE9 C4E5EAEB 7E2DD5A2   |..s.ctl.....~-..|
```

```
000000f0:  1FE58E32 AA82EB4F 104E72E4 EB69D87E   |...2...O.Nr..i.~|
-hit enter for more-

# efsmgr -d file2

# ls -iU file2
    5 -rw-r--r---    1 root     system          17 Sep 20 11:53 file2
```

Important: Encryption and de-encryption changes the position of the files on a file system. Files are copied during these operations and therefore the inode numbers will change.

2.3.7 Encryption inheritance

An EFS enabled file system does not imply that every file in it is encrypted. To achieve this encryption, the you must enable encryption inheritance. There are two levels of inheritance:

- Activated on the file system level
- Activated on the directory level

All new files and subdirectories will inherit the encryption settings of the parent directory. Directories themselves are never encrypted; they only inherit encryption.

Example 2-3 shows an example of encryption inheritance.

Example 2-3 Encryption inheritance

```
# ls -U
total 24
-rw-r--r---    1 root     system          17 Sep 20 13:49 file1
-rw-r--r---    1 root     system          17 Sep 20 11:53 file2
-rw-r--r---    1 root     system          17 Sep 20 10:54 file3
drwxr-xr-x-    2 root     system         256 Sep 20 10:30 lost+found

# mkdir inh_dir
# efsmgr -E inh_dir
# mv file[1-3] inh_dir/

# ls -U inh_dir/
total 48
-rw-r--r--e    1 root     system          17 Sep 20 13:49 file1
-rw-r--r--e    1 root     system          17 Sep 20 11:53 file2
```

```
-rw-r--r--e    1 root     system          17 Sep 20 10:54 file3
```

Of special note are the following:

- Inheritance can be deactivated with the **`efsmgr -D /path/directory`** command.
- Use the **`efsmgr -s -E /fsmountpoint`** command and the **`efsmgr -s -D /fsmountpoint`** command to set or unset inheritance on the file system level.

> **Note:** Enabling or disabling inheritance has no effect to already existing files in the directory or file system. You must use the **`efsmgr`** command to change the encryption settings.

2.3.8 Considerations

The following are general considerations:

- Make backups of your keystores.
- The RSA keys of the users keystore are automatically loaded into the kernel on login as long as the user login and keystore password are identical. If this is not the case, the user must run the **`efskeymgr -o ksh`** command and enter the user password. You can exchange the ksh shell with another shell if needed.
- In order to successfully encrypt or decrypt a file, there must be enough free space in the file system where free space >= filesize.
- An encrypted file does not occupy more file system space. Note that 4 KB is added to size because of the encrypted metadata in the extended attributes per file. In environments with large numbers of files, this might be relevant.
- Once a JFS2 file system is EFS enabled, it cannot be undone.
- DIO/CIO modes on encrypted files will not perform as well as on regular files.
- Performance of encryption should be verified in advance before activating EFS on a production environment to ensure it meets your requirements.
- System workload partitions (WPARs) are supported. After executing the **`efsenable`** command in the global environment, all system WPARs can use EFS.
- The file systems /, /usr, /var, and /opt cannot be EFS enabled.
- With AIX Version 6.1, you cannot store the RSA Keys on an LDAP server.
- NFS exports of a EFS file system are not supported.

- EFS is an AIX V6.1 or later feature and can be used only with JFS2. Previous AIX versions are not supported.
- To be able to do backups of encrypted data, the manufacturer of your backup software must provide support for EFS. Note that the AIX commands **backup**, **tar**, and **cpio** are already enabled to handle encrypted files.

2.4 iSCSI target mode software solution

As an enhancement of AIX V6.1, the iSCSI target software device driver can be used over a Gigabit (or higher speed) Ethernet adapter and host-located TCP/IP stack enabling AIX to act as one iSCSI target device or as several iSCSI target devices. The iSCSI target driver exports local disks or logical volumes to iSCSI initiators that connect to AIX using the iSCSI protocol that is defined in RFC 3720 and TCP/IP.

Each target device has an iSCSI Qualified Name and a set of logical unit numbers (LUNs) that are available to initiators that connect to the virtual iSCSI target. For each target device, you can specify which network interface and which TCP/IP port numbers the target driver can use to accept incoming connections.

> **Note:** The iSCSI target mode software solution is available in the AIX V6.1 Expansion Pack. Please refer to your AIX V6.1 release notes for more detailed information.

2.4.1 iSCSI software target considerations

The name for each virtual iSCSI virtual target is specified through the SMIT menus. It is recommended to use the iSCSI Qualified Name (IQN) convention to specify this name. There is no restriction on the name convention, but not using an IQN name might prevent initiators from logging to the defined target.

To display the current name of an iSCSI target device and verify if it uses the proper name convention, issue the following command and look for the iscsi_name field value:

```
# lsattr -E -l target0
```

In the previous example, target0 represents the name of the iSCSI software target device.

2.4.2 SMIT interface

iSCSI configuration is done by using SMIT menus. To configure the iSCSI target mode software driver, use the following SMIT path:

Devices → iSCSI → iSCSI Target Device

You can also use the SMIT menu shortcut **`smit tmiscsi`** to access the iSCSI software target menu.

> **Note:** For detailed configuration information about the iSCSI software target driver, refer to the AIX V6 Information Center and man pages.

In addition to the SMIT menus, the **`lsdev`** and **`rmdev`** commands can be used for listing and removing iSCSI target mode software devices.

3

Workload Partitions overview and resource management

This chapter discusses Workload Partitions (WPARs). WPARs are virtualized software-based partitions running within one AIX V6.1 operating system instance. This chapter contains the following sections:

3.1 Overview

WPARs are virtualized software based partitions running within one AIX V6.1 operating system instance.

WPAR virtualized hardware resources, such as memory and disk space, are partitioned by the operating system for the purpose of isolating specific applications or AIX workload environments. In contrast to LPARs, multiple WPARs can be created within a single OS copy, so a single LPAR running AIX V6.1 can contain multiple WPARs.

In general, LPARs are used to virtualize a system at the hardware level, while WPARs are used to virtualize a running AIX V6.1 system running at the software level.

There are two forms of workload partitions:

System WPAR Presents an environment that is most similar to a stand-alone AIX system. This WPAR type runs most of the system services that would be found in a stand-alone system and does not share writable file systems with any other WPAR or the global system.

Application WPAR Has all the process isolation that a system WPAR provides, except that it shares file system name space with the global system and any other application WPARs defined within the system. Other than the application itself, a typical Application WPAR only runs an additional light weight init process within the WPAR.

In this publication, we do not intent to cover all details of WPAR concepts, capabilities, and planning, but rather discuss specific AIX V6.1 features enabled for support of our WPAR environment.

> **Note:** For a detailed list of WPARs concepts and functionality, refer to *Introduction to Workload Partition Management in IBM AIX Version 6.1*, SG24-7431.

3.2 WPAR based system virtualization

WPAR provides a solution for partitioning one AIX operating instance in multiple encapsulated environments: each environment, called a workload partition, can host applications and execute them isolated from applications running within other WPARs.

Figure 3-1 illustrates how WPARs can be implemented within multiple AIX instances of the same physical server, whether they execute in dedicated LPARs or micro partitions.

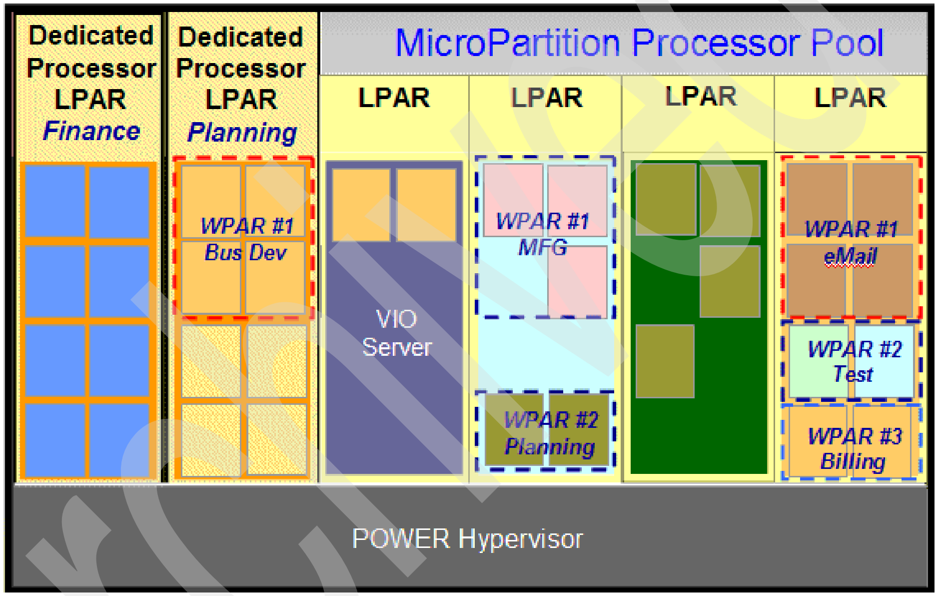


Figure 3-1 Multiple WPAR execution environments

Figure 3-1 shows how a WPAR can be implemented across a fully virtualized IBM System p server environment. Each process running on a WPAR is isolated from the rest of the environment.

3.3 Management tools

Table 3-1 lists the different AIX V6.1 WPAR management tools. For more details about each individual management tool listed in this publication, please refer to Chapter 3, “Overview of WPAR Management Tools”, in *Introduction to Workload Partition Management in IBM AIX Version 6.1*, SG24-7431.

Table 3-1 WPAR management options

Tool or function	Part of	Usage
AIX command-line interface	AIX Base	Creation, activation, modification, and deletion of WPARs.
SMIT/smitty	AIX Base	Identical to CLI usage.
WLM	AIX Base	WLM provides the underlying technology for WPAR resource management, but is not directly used by system administrators to manage WPARs.
WPAR checkpoint and relocation command-line interface	IBM Workload Partitions Manager™ for AIX	A checkpoint of the runtime status of a WPAR that can be used to resume a workload at a specific point of its execution, and, optionally, to move it to a different server.
WPAR Manager GUI	IBM Workload Partitions Manager for AIX	Automation of WPAR relocation, load balancing, metering, inventory, performance data collection, and policy based mobility.

3.3.1 Packaging

The WPAR management tools features provided in Table 3-1 that are listed as AIX Base are part of the AIX built-in operating system features.

The WPAR built-in AIX features are provided by the bos.wpars filesets.

The WPAR Manager, an additional program, consists of the following filesets:

mcr.rte The support for WPAR mobility

wparmgt.agent.rte The WPAR agent executing in all LPARs containing managed WPARs

wparmgt.mgr.rte The WPAR manager executing in the management LPAR

wparmgt.cas.agent The Common Access Service agent executing in all LPARs containing managed WPARs

wparmgt.cas.agentmgr
The Common Access Service agent executing in the management LPAR

lwi.rte Eclipse based Light Weight Infrastructure (LWI) runtime.

There is no fileset providing the WPAR Agent Console role. The console can be accessed by any Web browser running on a workstation with an IP connection to the WPAR manager.

3.4 System trace support

This section discusses WPAR metrics support for system trace and disk Input/Ouput metrics.

3.4.1 Overview

The trace facility helps isolate system problems by monitoring selected system events. Events that can be monitored include entry and exit to selected subroutines, kernel routines, kernel extension routines, and interrupt handlers. When the trace facility is active, information about these events is recorded in a system trace log file.

The trace facility includes commands for activating and controlling traces and generating trace reports.

Applications and kernel extensions can use several subroutines to record additional events. These trace reports can then be used by performance tools to make evaluations of system performance and activity.

3.4.2 WPAR tracing capabilities

In AIX Version 6.1, system trace is WPAR aware. Trace entries are able to be correlated to the WPAR that the trace belongs to. This allows administrators and performance tools to determine usage based on WPARs.

The following functions have been added in AIX V6.1 for WPAR support of trace capabilities:

- Launch a trace from within a WPAR
- Ability to correlate a trace entry to a WPAR
- Filtering which WPARs trace entries to log (global only)
- Filtering which WPARs entries to report (global only)
- Running more than one kernel trace at the same time
- Additional trace utility hooks
- Ability to run more than one kernel trace at the same time

Both the **trace** and **trcrpt** commands support filtering based on WPARs. This helps the global system from collecting unnecessary trace entries for WPARs, and the opposite, which helps reducing the amount of trace entries in the trace buffer. Also, when displaying a report, the user is now able to only display trace entries for desired WPARs.

3.4.3 Trace WPAR filtering from the global environment

The **trace** command now supports the parameters for filtering WPAR specific system traces provided in Table 3-2.

Table 3-2 New trace command WPAR filtering options

Filtering option	Description
-W	Includes the workload partition's configured ID (CID) for the current process with each hook. This flag is only valid in the Global system in a workload partition environment
-@ WPARName [,WPARName]	Traces on the listed workload partitions. Multiple WPAR names can be separated by commas or enclosed in quotes and separated by spaces. Specify *Global* to include the current Global system into the trace. This flag is only valid in the Global system in a workload partition environment.

> **Note:** The **`trcrpt`** command can report the WPAR's CID whether or not the W option is specified as long as the following events are being traced: 134, 139, 210, 465, 5D8, or the hooks group WPAR (-J wpar).

SMIT trace fast path

The SMIT panel that starts a trace is now updated to include the additional options for tracing only certain WPARs and including the WPAR CID in the trace entries. Figure 3-2 shows the SMIT panel changes. New in this version are the last two fields. To access this SMIT panel, you must issue the following command:

```
# smitty trcstart
```

```
                                START Trace

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

                                                       [Entry Fields]
  EVENT GROUPS to trace                               []                       +
  ADDITIONAL event IDs to trace                       []                       +
  Event Groups to EXCLUDE from trace                  []                       +
  Event IDs to EXCLUDE from trace                     []                       +
  Process IDs to Trace                                []                       +
  Program to Trace                                    []
  Propagate Tracing to                                [new processes and thr>  +
  Trace MODE                                          [alternate]              +
  STOP When Logfile full?                             [no]                     +
  LOG FILE                                            [/var/adm/ras/trcfile]
  SAVE PREVIOUS logfile?                              [no]                     +
  Omit PS/NM/LOCK HEADER to log file?                 [yes]                    +
  Omit DATE-SYSTEM HEADER to log file?                [no]                     +
  Run in INTERACTIVE mode?                            [no]                     +
  Trace BUFFER SIZE in bytes                          [262144]                  #
  LOG FILE SIZE in bytes                              [2621440]                 #
  Buffer Allocation                                   [automatic]              +
  WPAR names to Trace                                 []                       +
  Save WPAR's CID in trace entries?                   [no]                     +


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

Figure 3-2 SMIT trcstart fast path menu options

SMIT trace panel field details

These fields are not valid if ran within a WPAR. If values are specified from within a WPAR, then the command will fail. Table 3-3 describes the newly added fields.

Table 3-3 New trace fields for WPAR smitty trcstart panel

Field name	Description	Values
WPAR names to Trace	Specify the WPAR names of the currently running WPARs to be traced. Specify keyword Global to include this system in the trace. If you do not specify a value, all WPARs and the Global will be traced.	A list of WPAR names to be included in the trace. Global should be used to indicate this system. If no value is specified, then all WPARs and the Global will be traced. If just WPAR names are specified, then only those WPARs will be traced and the Global will not.
Save WPAR's CID in trace entries?	Select yes to save the WPAR's configuration ID (CID) in each trace entry. These CIDs can then be displayed and used in filtering in the **trcrpt** command.	A default value of no means do not include the WPAR's CID in trace. The field can be toggled between no and yes using the Tab key.

3.4.4 Trace report filtering from the Global environment

Similar to **trace** command filtering, the **trcrpt** command is able to filter which WPARs it is interested. This requires trace entries that are placed in the trace log to be able to be correlated to the appropriate WPAR and reducing the amount of data reported. Additionally, there is also an option to display the CID or the WPAR name for each trace entry in the report.

The **trcrpt** command now supports a new -@ <WPARList> as well as new -O options for filtering WPAR specific system traces. Table 3-4 contains detailed descriptions of these new parameters.

Table 3-4 New trcrpt command WPAR filtering options

<table>
<tr><th>Filtering option</th><th>Description</th></tr>
<tr><td>-@ <WPARList></td><td>Will only display trace entries that were collected for the indicated WPARs. The WPARList can contain either WPAR names or WPAR IDs. A WPAR ID of '0' or WPAR name of 'Global' will display the trace entries for the Global system.</td></tr>
<tr><td>-O wparname=[on|off]</td><td>The new -O option displays the workload partition names in the trace report. The default value is off.</td></tr>
<tr><td>-O cid=[on|off]</td><td>The new -O option displays the workload partition's configured ID (CID) in the trace report. The default value is off.</td></tr>
</table>

SMIT trace report fast path

The SMIT panel for trace reports is now updated to include the additional options of filtering on WPARs and displaying the WPAR name or the WPAR CID in the report. Figure 3-3 shows the SMIT panel display changes for the new panel that include these new options (*highlighted*). To access this SMIT panel, you must issue the following command:

```
# smitty trcrpt
```

```
                              Generate a Trace Report

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

                                                         [Entry Fields]
  Show exec PATHNAMES for each event?                 [yes]                  +
  Show PROCESS IDs for each event?                    [no]                   +
  Show THREAD IDs for each event?                     [no]                   +
  Show WPAR Names for each event?                     [no]                   +
  Show WPAR CIDs for each event?                      [no]                   +
  Show CURRENT SYSTEM CALL for each event?            [yes]                  +
  Time INCREMENTS for report                          [elapsed only]         +
  Event Groups to INCLUDE in report                   []                     +
  IDs of events to INCLUDE in report                  []                     +X
  Event Groups to EXCLUDE from report                 []                     +
  IDs of events to EXCLUDE from report                []                     +X
  WPARs to include in report                          []
  STARTING time                                       []
  ENDING time                                         []
  LOG FILE to create report from                      [/var/adm/ras/trcfile]
  FILE NAME for trace report (default is stdout)      []



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

Figure 3-3 SMIT panel for smitty trcrpt panel fast path option

SMIT trace report panel field details

The highlighted fields are not valid if run within a WPAR. If the values are specified from within a WPAR, then the command will fail. Table 3-5 on page 63 describes the newly added fields.

Table 3-5 New trace report filtering fields for WPAR in the smitty trcrpt panel

Field name	Description	Values
Show WPAR Names for each event?	Select whether you wish the WPAR Names to be displayed (or printed) in the trace report. The default is no.	The default value of no means do not show WPAR Names in report. The field can be toggled between no and yes using the Tab key.
Show WPAR CIDs for each event?	Select whether you wish the WPAR configured IDs (CID) to be displayed (or printed) in the trace report. The default is no.	The default value of no means do not show WPAR CIDs in report. The field can be toggled between no and yes using the Tab key.
WPARs to INCLUDE in report	Specify a list of WPARs to be included in the trace report.	A list of WPAR names or configured IDs (CID) that the trace report should use for filtering. If not specified, trace entries for all WPARs will be reported. Use Global or 0, if filtering on WPARs and you want the Global's trace entries.

3.4.5 Tracing from within a WPAR

The ability to filter and trace based on a WPAR is beneficial to the administrator of the Global system. However, it is also very beneficial for the administrator of the WPAR itself to run trace and collect trace reports based on its WPAR activities.

The ability to trace from within a WPAR is an AIX V6.1 supported feature. The **trace** and **trcrpt** commands work the same as in the Global environment with the exception of the WPAR specific options available for WPAR filtering from the Global. Those are not required or valid from within a WPAR.

Enabling trace

In order to be able to start trace from within a WPAR, this privilege needs to be enabled, because it is not enabled by default. The trace facility can be enabled during and after the creation of a WPAR using both the command line and the SMIT panels.

To enable trace using the command-line interface, use the **mkwpar** or **chwpar** commands with the following syntax:

```
# chwpar -n WPARname -S privs+=PV_KER_RAS
```

In the example, WPARname is the name of your existing WPAR.

To enable trace using the SMIT panel menus, you can use the following sequence; for this example, we use a System WPAR:

1. Run **# smitty wpar** and select **Administer SYSTEM Workload Partitions → Change / Show System Workload Partition Characteristics → Change / Show General Characteristics**.
2. Select the WPAR to change and include the PV_KER_RAS privilege in the Privileges field.

Multi-session trace support

Due the nature of WPARs, administrators for multiple WPARs may want to run a trace based performance tool at the same time or at the same time as the Global.

> **Important:** The current implementation of system trace allows one instance of trace to be run in the Global environment and seven for WPARs concurrently.
>
> Tracing capabilities are not available for Application type WPARs. The trace command system services are not extended to it. The same is true for trace based commands such as **filemon**, **netpmon**, **pprof**, **tprof**, and **curt**.

3.5 File system metrics support

I/O statistics are heavily used by system administrators to determine if excess I/O is causing system performance issues. This data is then used by the administrators to reorganize their activities to increase their systems utilization.

To provide meaningful I/O file system metrics for users of WPAR and AIX V6.1 systems in general, file system metrics are collected at the logical file system (LFS) layer. Since System WPARs have a separate mount for each file system that it has, even if shared from the Global (namefs), each mounted file system has its metrics specific to that mounted file system. In addition to that, there are also metrics information for remotely mounted file systems to give a logical view of the clients activity on that file system.

For WPAR specific usage, a given WPAR only collect metrics for the file systems that belong to it.

For more information about collecting I/O related statistics for WPARs, see 3.7.3, “Updates for the iostat command” on page 71 and 3.7.9, “Updates for the topas command” on page 84.

3.6 Network metrics support

Network metrics are very important statistical data gathered and analyzed by AIX administrators. This metric data is also consumed by user applications to make decisions based in the network performance of the system.

Existing network statistics are gathered from the network adapter all the way up to the UDP/TCP layer. WPARs, however, do not have access to the physical devices. Network activity for a WPAR is managed by utilizing aliases in the Global environment to attach the WPAR's IP to an existing Global environment interface. These alias-based IPs are attached to the appropriate WPAR's socket, thus enabling a WPAR to access its packets in the network.

To display and monitor the network statistics for WPARs, the **netstat** command has been updated with the following capabilities:

- Ability to display network statistics for a given WPAR from the Global environment through the new -@ WPARname flag
- Ability to run the command from within a WPAR and display statistics relevant to its isolated environment

The following is a list the supported **netstat** flags from the WPAR environment:

```
netstat [-Aaon] [-f address_family]
            [-inrsu] [-f address_family] [-p proto]
            [-n] [-I interface] [interval]
```

Flags not included in the list are not supported from the Global environment with the -@ flag.

3.7 Performance tools updates for WPAR support

Performance monitoring is an important task for AIX system administrators. The addition of WPAR in this version of the operating system facilitates the gathering and filtering of performance related statistics of selective applications and workloads isolated in both the Global environment, and WPARs.

This section discusses the following AIX V6.1 performance tools updates to support proper tracking and filtering of system performance data for WPARs, listed in alphabetical order:

Due to the extensive amount of information about each one of the commands, this publication does not describe all the details of the changes, but rather provides a few examples the authors considered useful.

For a detailed list of changes on a specific AIX updated command, refer to the man pages or the AIX V6.1 product documentation.

3.7.1 Updates for the curt command

The **curt** command is used to convert an AIX trace file into a number of statistics related to CPU utilization (application, kernel, NFS, Flih, Slih, and Wait), and either process, thread, or pthread activity.

The following enhancements have been made to this performance tool to support WPAR specific metrics:

- Ability to filter statistics for a given WPAR, or WPAR list from the Global environment
- Ability to display organized statistics for all active WPARs from the global environment
- Ability to run the command from within a WPAR and display statistics relevant to its isolated environment

Important: In order to use this command within a WPAR, trace privileges must be enabled in the WPAR. Refer to "Enabling trace" on page 63.

Table 3-6 describes the updates made to this command for support of WPARs.

Table 3-6 Option changes for curt command

Flag or argument	Behavior in WPAR	Behavior in Global
"-@ Wparlist"	Fails with a usage message as the -@ Wparlist option is illegal inside the WPAR.	Prints relevant information for the given WPAR only. If the specified WPAR does not exist, or is not active, or the trace have been taken in a non-WPAR system, then it fails with a `workload partition not found message` unless the workload partition name is Global.
"-@ ALL"	Fails with a usage message as the -@ ALL option is made illegal inside the WPAR.	Executes normally and prints the summary of all WPARs. A workload partition name is displayed for each record.
"-@"	Fails with a usage message as the -@ option is made illegal inside the WPAR.	Executes normally and prints the process tree with related WPARs and Global dependency. A workload partition name is displayed for each record. If trace have been taken in a non-WPAR system, then it fails with the `trace file contains no WPAR` error.
"-b"	Fails with a usage message as the -b option is made illegal inside the WPAR.	Will print the WPAR ID of each process in the processes table, and an additional WPAR table associating active WPARs to their CIDs.

When using **`curt`**, be aware of the following details in the reports:

- Kproc Summary (by Tid), shows no indication of the WPAR name because all the kernel processes are branded to the Global environment.

- The **curt** command reports summaries of all the WPARs that existed on the system during the time of a trace collection and their CPU consumption (one line per WPAR).

  For each category (application, syscall, hcall, kproc, nfs, flih, and slih), the amount of CPU time is expressed as a percentage of total processing time.

The total amount of CPU time is expressed as percentage of the total processing time (of the system) in milliseconds.

3.7.2 Updates for the filemon command

The **filemon** command monitors the performance of the file system, and reports the I/O activity on behalf of logical files, virtual memory segments, logical volumes, and physical volumes. The command will always behave the same way in post-processing mode, regardless of whether it runs inside a WPAR or not.

The following enhancements have been made to this command to support WPAR specific metrics:

- The ability to filter I/O traced statistics for a given WPAR from the Global environment
- The ability to display organized statistics for all active WPARs from the global environment
- The ability to run the command from within a WPAR and display statistics relevant to its isolated environment

> **Important:** In order to use this command within a WPAR, trace privileges must be enabled in the WPAR. Refer to "Enabling trace" on page 63.

Table 3-7 describes the updates made to the **filemon** command for support of WPARs.

Table 3-7 Option changes for filemon command

Flag or argument	Behavior in WPAR	Behavior in Global
none	Executes the default report and displays information specific to the WPAR.	Executes normally with no changes from previous versions of AIX.

Flag or argument	Behavior in WPAR	Behavior in Global
"-@ Wparlist"	Fails with a usage message as the -@ Wparlist option is made illegal inside the WPAR.	Prints relevant information for the given WPAR only. If the specified WPAR does not exist or is not active, then it fails with the `workload partition not found` message unless the workload partition name is Global.
"-@ ALL"	Fails with a usage message as the -@ ALL option is made illegal inside the WPAR.	Executes normally and prints the summary of all WPARs. A workload partition name is displayed for each record.
"-@"	Fails with a usage message as the -@ option is made illegal inside the WPAR.	Executes normally and prints additional WPAR information. A workload partition name is displayed for each record.

Important: When running the **`filemon`** command from the Global environment with any of the -@ options, always use the -O lf option. This is due to WPAR restrictions. For example:

```
# filemon -O lf -@ mywpar1
```

Example 3-1 demonstrates the output of the **`filemon`** command ran without any parameters within the mywpar1 WPAR.

Example 3-1 The filemon command example

```
Cpu utilization: 100.0%
Cpu allocation:   75.5%
[filemon: Reporting started]

Most Active Files
------------------------------------------------------------------------
  #MBs  #opns   #rds   #wrs  file                     volume:inode
------------------------------------------------------------------------
   0.4      1     91      0  unix                     <major=10,minor=5>:9565
   0.0      1      2      0  ksh.cat                  <major=10,minor=5>:17456
   0.0      1      2      0  cmdtrace.cat             <major=10,minor=5>:17280
   0.0      9      2      0  SWservAt                 <major=10,minor=11>:123
   0.0      9      2      0  SWservAt.vc              <major=10,minor=11>:124
```

```
------------------------------------------------------------------------
Detailed File Stats
------------------------------------------------------------------------

FILE: /unix  volume: <major=10,minor=5>  inode: 9565
opens:                  1
total bytes xfrd:       372736
reads:                  91      (0 errs)
  read sizes (bytes):   avg  4096.0 min    4096 max    4096 sdev     0.0
  read times (msec):    avg   0.004 min   0.004 max   0.006 sdev   0.000
lseeks:                 172

FILE: /usr/lib/nls/msg/en_US/ksh.cat  volume: <major=10,minor=5>  inode: 17456
opens:                  1
total bytes xfrd:       8192
reads:                  2       (0 errs)
  read sizes (bytes):   avg  4096.0 min    4096 max    4096 sdev     0.0
  read times (msec):    avg   0.005 min   0.004 max   0.006 sdev   0.001
lseeks:                 5

FILE: /usr/lib/nls/msg/en_US/cmdtrace.cat  volume: <major=10,minor=5>  inode:
17280
opens:                  1
total bytes xfrd:       8192
reads:                  2       (0 errs)
  read sizes (bytes):   avg  4096.0 min    4096 max    4096 sdev     0.0
  read times (msec):    avg   0.005 min   0.005 max   0.005 sdev   0.000
lseeks:                 8

FILE: /etc/objrepos/SWservAt  volume: <major=10,minor=11>  inode: 123
opens:                  9
total bytes xfrd:       796
reads:                  2       (0 errs)
  read sizes (bytes):   avg   398.0 min     328 max     468 sdev    70.0
  read times (msec):    avg   0.003 min   0.003 max   0.004 sdev   0.001
lseeks:                 1

FILE: /etc/objrepos/SWservAt.vc  volume: <major=10,minor=11>  inode: 124
opens:                  9
total bytes xfrd:       80
# reads:                        2       (0 errs)
  read sizes (bytes):   avg    40.0 min      40 max      40 sdev     0.0
  read times (msec):    avg   0.003 min   0.002 max   0.003 sdev   0.000
lseeks:                 1
[filemon: Reporting completed]
```

```
[filemon: 6.999 secs in measured interval]
```

As shown in Example 3-1 on page 69, the **filemon** command now is WPAR aware and reports I/O statistics relevant to the WPAR where is being run.

3.7.3 Updates for the iostat command

The **iostat** command is used to display and monitor I/O statistics. Such statistics are frequently used by system administrators to analyze system I/O throughput and potential bottlenecks.

The following enhancements have been made to this command to support WPAR specific metrics:

- The ability to filter I/O activities for a given WPAR from the Global environment
- The ability to display organized statistics for all active WPARs from the global environment
- The ability to run the command from within a WPAR and display statistics relevant to its isolated environment
- A new command line option -f, which displays the file systems utilization report
- A new command line option -F, which displays the file systems utilization report, and turns off other utilization reports
- Support for the -s, -T, -l, -V, -f, -F options within a WPAR

Table 3-8 describes the updates made to the **iostat** command for support of WPARs.

Table 3-8 Option changes for iostat command

Flag or argument	Behavior in WPAR	Behavior in Global
none	Executes the default report and displays an @ above the metrics specific to the WPAR.	Executes normally with no changes from previous versions of AIX.

Flag or argument	Behavior in WPAR	Behavior in Global
"-@ Wparname"	Fails with a usage message as the -@ Wparname option is made illegal inside the WPAR.	Printsrelevant information for the given WPAR only. If the specified WPAR does not exist or is not active, then it fails with a `workload partition not found` message unless the workload partition name is Global.
"-@ ALL"	Fails with a usage message as the -@ ALL option is made illegal inside the WPAR.	Executes normally and prints the summary of all WPARs. A workload partition name is displayed for each record.
"-a"	Fails with a usage message as the -a option is illegal inside the WPAR.	Executes normally and prints adapter throughput information associated with Global. This option cannot be used with the -@ option.
"-A"	Fails with a usage message as the -A option is illegal inside the WPAR.	Executes normally and prints asynchronous IO utilization information associated to Global. This option cannot be used with the -@ option.
"-d"	Fails with a usage message as the -d option is illegal inside the WPAR.	Executes normally, turning off the display of the TTY utilization report or the CPU utilization report associated with Global. This option cannot be used with the -@ option.
"-D"	Fails with a usage message as the -D option is illegal inside the WPAR.	Executes normally and prints extended tape/drive utilization information associated with Global. This option cannot be used with the -@ option.
"-f"	Displays the file system report appended to the default O/P.	Displays the file system report only along with the System configuration.

Flag or argument	Behavior in WPAR	Behavior in Global
“-F”	Displays the file system report only along with the System configuration.	Displays the file system report only along with the System configuration.
“-m”	Fails with a usage message as the -m option is illegal inside the WPAR.	Executes normally and prints path utilization information associated with Global. This option cannot be used with the -@ option.
“-P”	Fails with a usage message as the -P option is illegal inside the WPAR.	Executes normally and prints tape utilization information associated with Global. This option cannot be used with the -@ option.
“-q”	Fails with a usage message as the -q option is illegal inside the WPAR.	Executes normally and prints AIO queues and their request count information associated with Global. This option cannot be used with the -@ option.
“-Q”	Fails with a usage message as the -Q option is illegal inside the WPAR.	Executes normally and prints a list of all the mounted file systems and the associated queue numbers with their request counts associated with Global. This option cannot be used with the -@ option.
“-s”	Displays the system throughput report.	Displays only TTY and CPU.
“-t”	Fails with a usage message as the -t option is illegal inside the WPAR.	Executes normally, turning off of the display of the disk utilization report associated with Global. This option cannot be used with the -@ option.

Flag or argument	Behavior in WPAR	Behavior in Global
“-z”	Fails with a usage message as the -z option is illegal inside the WPAR.	Executes normally, resetting the disk input/output statistics associated with Global. Only root users can use this option. This option cannot be used with the -@ option.

The following example shows the output of the -@ ALL option when used in the Global environment:

```
# iostat -@ ALL

System configuration: lcpu=2 ent=0.30

tty:      tin         tout    avg-cpu: % user % sys % idle % iowait physc % entc
          1.5          6.6              11.9  57.0   31.1      0.1   0.0    0.0

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0           0.2       3.3       0.6      70685    100556
cd0              0.0       0.0       0.0          0         0
-------------------------------mywpar1-------------------------------------
tty:      tin         tout    avg-cpu: % user % sys % idle % iowait physc % entc
           -            -               29.9  70.1    0.0      0.0   0.0    0.0
-------------------------------mywpar2-------------------------------------
tty:      tin         tout    avg-cpu: % user % sys % idle % iowait physc % entc
           -            -               30.3  69.7    0.0      0.0   0.0    0.0
                          ==========================
#
```

As shown in the previous example, the **iostat** command now is WPAR aware and reports WPAR I/O relevant information from the Global environment.

3.7.4 Updates for the netpmon command

The **netpmon** command monitors a trace of system events, and reports on network activity and performance during the monitored interval, such as CPU utilization, network device-driver I/O, Internet sockets calls, and NFS I/O. The command will always behave the same way in post-processing mode, regardless of whether it runs inside a WPAR or not.

The following enhancements have been made to this command to support WPAR specific metrics:

- The ability to filter network traced statistics for a given WPAR from the Global environment.

- The ability to display organized statistics for all active WPARs from the global environment.
- The ability to run the command from within a WPAR and display statistics relevant to its isolated environment.

Important: In order to use this command within a WPAR, trace privileges must be enabled in the WPAR. Refer to "Enabling trace" on page 63.

Table 3-9 describes the updates made to the **`netpmon`** command for support of WPARs.

Table 3-9 Option changes for netpmon command

Flag or argument	Behavior in WPAR	Behavior in Global
none	Executes the default report and displays information specific to the WPAR.	Executes normally with no changes from previous versions of AIX.
"-@ Wparname"	Fails with a usage message, as the -@ Wparlist option is made illegal inside the WPAR.	Prints relevant information for a given WPAR only. If the specified WPAR does not exist or is not active, then it fails with a `workload partition not found` message unless the workload partition name is Global.
"-@ ALL"	Fails with a usage message as the -@ ALL option is made illegal inside the WPAR.	Executes normally and prints a summary of all WPARs. A workload partition name is displayed for each record.
"-@"	Fails with a usage message as the -@ option is made illegal inside the WPAR.	Executes normally and prints additional WPAR information. A workload partition name is displayed for each record.

Example 3-2 demonstrates the output of the **netpmon -@** command when ran within the Global environment.

Example 3-2 The netpmon command in a global environment

```
Fri Oct  5 15:05:21 2007
System: AIX 6.1 Node: server5 Machine: 00C0F6A04C00


========================================================================
=

Process CPU Usage Statistics:
-----------------------------
                                                         Network
Process (top 20)             PID  CPU Time   CPU %   CPU %  WPAR
--------------------------------------------------------------------
trcstop                   454690    0.0029   9.182   0.000  Global
getty                     303290    0.0014   4.419   0.000  Global
wlmsched                   65568    0.0012   3.725   0.000  Global
ksh                       381130    0.0009   2.739   0.439  Global
xmgc                       49176    0.0008   2.632   0.000  Global
gil                        61470    0.0008   2.356   2.356  Global
swapper                        0    0.0007   2.125   0.000  Global
java                      270528    0.0005   1.491   0.000  Global
netpmon                   393260    0.0005   1.418   0.000  Global
sched                      12294    0.0003   0.977   0.000  Global
netpmon                   454688    0.0002   0.779   0.000  Global
lockd-1                   426196    0.0002   0.741   0.000  Global
rpc.lockd                 139406    0.0001   0.465   0.000  Global
sendmail:                 332014    0.0001   0.204   0.000  mywpar1
init                      368830    0.0001   0.189   0.000  mywpar1
sendmail:                 204900    0.0001   0.182   0.000  Global
pilegc                     45078    0.0000   0.079   0.000  Global
aixmibd                   123008    0.0000   0.069   0.000  Global
rmcd                      266378    0.0000   0.052   0.000  Global
netm                       57372    0.0000   0.046   0.046  Global
----------------------------------------------------------
Total (all processes)               0.0108  33.871   2.841
Idle time                           0.0083  25.906

========================================================================
=

First Level Interrupt Handler CPU Usage Statistics:
```

```
----------------------------------------------------
                                                  Network
FLIH                              CPU Time   CPU %   CPU %
-----------------------------------------------------------
PPC decrementer                     0.0089  27.944   0.000
data page fault                     0.0016   5.026   0.000
external device                     0.0003   1.086   0.011
queued interrupt                    0.0000   0.055   0.000
-----------------------------------------------------------
Total (all FLIHs)                   0.0109  34.112   0.011

========================================================================
=

Second Level Interrupt Handler CPU Usage Statistics:
-----------------------------------------------------
                                                  Network
SLIH                              CPU Time   CPU %   CPU %
-----------------------------------------------------------
<addr=      0x40cf618>              0.0006   1.740   0.077
-----------------------------------------------------------
Total (all SLIHs)                   0.0006   1.740   0.077

========================================================================
=

Detailed Second Level Interrupt Handler CPU Usage Statistics:
--------------------------------------------------------------

SLIH: <addr=      0x40cf618>
count:                   42
  cpu time (msec):       avg 0.013   min 0.009   max 0.035   sdev 0.005

COMBINED (All SLIHs)
count:                   42
  cpu time (msec):       avg 0.013   min 0.009   max 0.035   sdev 0.005
```

As shown in the previous example, the **netpmon** command is now WPAR aware and displays CPU and network related statistics relevant to the Global and WPAR environments.

3.7.5 Updates for the pprof command

The **pprof** command is used to report CPU usage of all kernel threads over a period of time. This tool uses the trace facility, allowing for the generation of reports for previously ran traces. The command will always behave the same way in post-processing mode, regardless of whether it runs inside a WPAR or not.

The following enhancements have been made to this command to support WPAR specific metrics:

- The ability to filter processes for a given WPAR, or WPAR list from the Global environment
- The ability to display organized statistics for all active WPARs from the global environment
- The ability to run the command from within a WPAR and display statistics relevant to its isolated environment

> **Important:** In order to use this command within a WPAR, trace privileges must be enabled in the WPAR. Refer to “Enabling trace” on page 63.

Table 3-10 provides the updates made to this command for support of WPARs.

Table 3-10 Option changes for pprof command

Flag or argument	Behavior in WPAR	Behavior in Global
“-@ Wparlist”	Fails with a usage message as the -@ Wparlist option is made illegal inside the WPAR.	Prints relevant information for the given WPAR only. If the specified WPAR does not exist or is not active, then it fails with a `workload partition not found` message unless the workload partition name is Global.
“-@ ALL”	Fails with a usage message as the -@ ALL option is made illegal inside the WPAR.	Executes normally and prints the summary of all WPARs. A workload partition name is displayed for each record.

Flag or argument	Behavior in WPAR	Behavior in Global
“-@”	Fails with a usage message as the -@ option is made illegal inside the WPAR.	Executes normally and prints the process tree with related the WPARs and Global dependency. A workload partition name is displayed for each record.

Example 3-3 demonstrates the pprof.cpu file output of the **pprof 2 -@** command.

Example 3-3 pprof.cpu output file sample

```
              Pprof CPU Report

              Sorted  by  Actual CPU  Time

              From: Fri Oct  5 07:34:38 2007
              To:   Fri Oct  5 07:34:40 2007


     E = Exec'dF = Forked
     X = ExitedA = Alive (when traced started or stopped)
     C = Thread Created

               Pname       PID     PPID  BE       TID     PTID  ACC_time  STT_time  STP_time   STP-STT  WPARs
               =====     =====    ===== ===     =====    =====  ========  ========  ========  ======== ======
               syncd    102564        1  AA    209013        0     0.010     1.015     1.126     0.111  Global
                wait      8196        0  AA      8197        0     0.006     0.009     2.020     2.011  Global
               pprof    430170   491578  AA   1831047        0     0.001     0.009     2.028     2.019  Global
                  sh    524344   430170  EE   1315027  1831047     0.001     0.018     0.019     0.001  Global
                nfsd    278674        1  AA    364727        0     0.001     1.737     1.737     0.001  Global
                xmgc     49176        0  AA     61471        0     0.001     1.027     1.028     0.001  Global
            wlmsched     65568        0  AA     98353        0     0.001     0.048     1.952     1.903  Global
               getty    315560        1  AA    577593        0     0.001     0.028     2.012     1.984  Global
             swapper         0        0  AA         3        0     0.001     0.078     1.979     1.901  Global
               pprof    377082   491578  AX   1622045        0     0.000     0.009     0.009     0.000  Global
               pprof    524346   430170  FE   1315029  1831047     0.000     2.020     2.021     0.000  Global
               pprof    524344   430170  FE   1315027  1831047     0.000     0.010     0.010     0.000  Global
                java    290966   311468  AA    983265        0     0.000     0.908     1.128     0.220  Global
      /usr/bin/sleep    524344   430170  EX   1315027  1831047     0.000     0.020     2.020     2.000  Global
                rmcd    327764   303290  AA   1188057        0     0.000     1.387     1.387     0.000  mywpar1
                wait     53274        0  AA     65569        0     0.000     0.009     0.178     0.169  Global
                 gil     61470        0  AA     90157        0     0.000     0.105     1.938     1.833  Global
                 gil     61470        0  AA     86059        0     0.000     0.298     1.605     1.307  Global
                rmcd    262334   213132  AA    552979        0     0.000     0.072     0.072     0.000  Global
               sched     12294        0  AA     12295        0     0.000     0.978     0.978     0.000  Global
           sendmail:    122980   213132  AA    274587        0     0.000     0.077     0.077     0.000  Global
                java    290966   311468  AA    532489        0     0.000     0.736     1.737     1.001  Global
                nfsd    278674        1  AA    368827        0     0.000     0.038     1.841     1.803  Global
                java    290966   311468  AA    975069        0     0.000     0.737     1.737     1.001  Global
                 gil     61470        0  AA     81961        0     0.000     0.098     1.898     1.800  Global
           rpc.lockd    266418        1  AA    389365        0     0.000     0.038     1.841     1.803  Global
             lockd-1    401640        1  AA    708773        0     0.000     0.038     1.841     1.803  Global
             lockd-2    364698        1  AA   1269907        0     0.000     0.038     1.841     1.803  Global
                 gil     61470        0  AA     94255        0     0.000     0.411     1.698     1.288  Global
                nfsd    278674        1  AA    405711        0     0.000     0.479     1.681     1.202  Global
             lockd-2    364698        1  AA   1351829        0     0.000     0.429     1.631     1.202  Global
             lockd-1    401640        1  AA   1056843        0     0.000     0.248     1.451     1.203  Global
           rpc.lockd    266418        1  AA    422099        0     0.000     0.411     1.618     1.207  Global
                java    290966   311468  AA    634935        0     0.000     0.737     1.737     1.001  Global
              pilegc     45078        0  AA     69667        0     0.000     1.090     1.090     0.000  Global
              pilegc     45078        0  AA     45079        0     0.000     1.090     1.090     0.000  Global
                netm     57372        0  AA     73765        0     0.000     1.153     1.153     0.000  Global
             lockd-2    364698        1  AA   1335445        0     0.000     1.972     1.972     0.000  Global
   /usr/bin/trcstop    524346   430170  EA   1315029  1831047     0.000     2.028     2.028     0.000  Global
                  sh    524346   430170  EE   1315029  1831047     0.000     2.023     2.023     0.000  Global
                                                                ========
```

```
0.024
```

As shown in the previous example, the **pprof** command is now WPAR aware and reports individual processes relevant to WPARs.

3.7.6 Updates for the procmon plug-in

The procmon plug-in is part of the Performance WorkBench graphical user interface. This plug-in helps to monitor the processes running on the AIX system and displays such information as CPU, memory, and entitlement on the current partition.

The following enhancements have been made to this command to support WPAR specific metrics:

- The partition performance tab has been updated to display the number of WPARs and their state in the current LPAR.
- There is a new tab displaying existing WPARs on the current LPAR with more detailed estate information, such as name, host name, and type.
- The Processes tab now indicates processes and their relationship to the Global or WPAR environments.

Figure 3-4 on page 81 shows processes in the virtual environment they belong to displayed in the WPAR column. If a process belongs to the Global environment, the field will read Global. If the process belongs to a WPAR, then it will display the WPAR name.

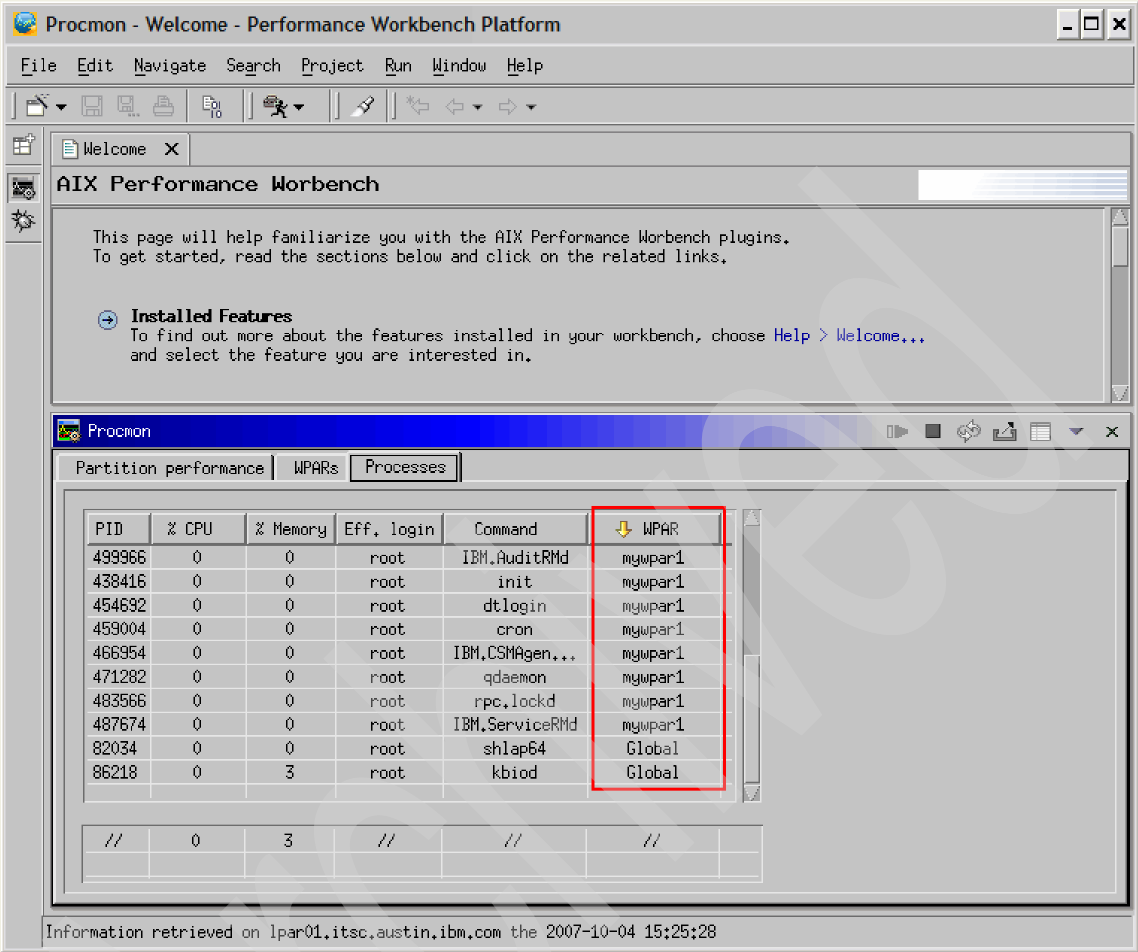


Figure 3-4 Performance Workbench - Processes tab view

The new WPAR column (*highlighted*) shown in Figure 3-4 allows the user to sort the processes by WPAR if desired.

3.7.7 Updates for the proctree command

The **proctree** command is used to print the process tree contained in a hierarchy specified by a given process or user ID. The ouput shows children processes indented from their respective parent processes. An argument of all digits is taken to be a process ID; otherwise, it is assumed to be a user login name.

The following enhancements have been made to this command to support WPAR specific metrics:

- The ability to filter processes for a given WPAR from the Global environment
- The ability to display organized statistics for all active WPARs from the global environment
- The ability to run the command from within a WPAR and display statistics relevant to its isolated environment

Table 3-11 describes the updates made to this command for support of WPARs.

Table 3-11 Option changes for proctree command

Flag or argument	Behavior in WPAR	Behavior in Global
none	Executes the default report and displays only the process tree specific to the WPAR.	Executes normally with no changes from the previous versions of AIX.
"-@ Wparname"	Fails with a usage message as the -@ Wparname option is made illegal inside the WPAR.	Prints the relevant information for the given WPAR only. If the specified WPAR does not exist or is not active, then it fails with the `workload partition not found` message unless the workload partition name is Global.
"-@"	Fails with a usage message as the -@ option is made illegal inside the WPAR.	Executes normally and prints the process tree with the related WPARs and Global dependency. A workload partition name is displayed for each record.

The following example demonstrates the output of the -@ option when used in the Global environment:

```
# proctree -@ mywpar1
mywpar1    438416   /etc/init
mywpar1       348294   /usr/sbin/srcmstr
mywpar1          188466   /usr/sbin/biod 6
mywpar1          299142   /usr/sbin/syslogd
mywpar1          356354   sendmail: accepting connections  nnections
mywpar1          372776   /usr/sbin/portmap
mywpar1          389218   /usr/sbin/rsct/bin/rmcd -a IBM.LPCommands -r
```

```
mywpar1          393216   /usr/sbin/writesrv
mywpar1          409616   /usr/sbin/inetd
mywpar1          413718   /usr/sbin/nfsrgyd
mywpar1          454692   /usr/dt/bin/dtlogin
mywpar1          466954   /usr/sbin/rsct/bin/IBM.CSMAgentRMd
mywpar1          471282   /usr/sbin/qdaemon
mywpar1          483566   /usr/sbin/rpc.lockd -d 0
mywpar1          487674   /usr/sbin/rsct/bin/IBM.ServiceRMd
mywpar1          499966   /usr/sbin/rsct/bin/IBM.AuditRMd
mywpar1       217218   /usr/lib/errdemon
mywpar1       459004   /usr/sbin/cron
#
```

As shown in the previous example, the **proctree** command is now WPAR aware and reports the process tree relevant to the specified WPAR.

3.7.8 Updates for the svmon command

The **svmon** command is used to report in-depth memory state information from the kernel in terms of pages.

The following enhancements have been made to this command to support WPAR specific metrics:

- The ability to filter memory information for a given WPAR or WPAR list from the Global environment
- The ability to display organized statistics for all active WPARs from the global environment
- The ability to run the command from within a WPAR and display statistics relevant to its isolated environment

Table 3-12 describes the updates made to this command for support of WPARs.

Table 3-12 Option changes for svmon command

Flag or argument	Behavior in WPAR	Behavior in Global
none	Executes the default report and displays an @ with information specific to the WPAR	Executes normally with no changes from the previous versions of AIX.

Flag or argument	Behavior in WPAR	Behavior in Global
"-@ Wparlist"	Fails with a usage message as the -@ Wparlist option is made illegal inside the WPAR.	Prints relevant information for a given WPAR only. If the specified WPAR does not exist or is not active, then it fails with a `workload partition not found` message unless the workload partition name is Global.
"-@ ALL"	Fails with a usage message as the -@ ALL option is made illegal inside the WPAR.	Executes normally and prints the summary of all WPARs. A workload partition name is displayed for each record.

The following example demonstrates the output of the **svmon -@ mywpar1** command ran from the Global environment:

```
# svmon -@ mywpar1

################################################################################
######## WPAR : mywpar1
################################################################################
               size       inuse        free         pin     virtual
memory       262144       31899       52482       31899      148643
pg space     131072        2656

               work        pers        clnt       other
pin             144           0           0       10322
in use         3671           0       28228

PageSize   PoolSize       inuse        pgsp         pin     virtual
s    4 KB         -       31819        2656          80       67091
m   64 KB         -           5           0           4        5097
#
```

As shown in the previous example, the **svmon** command is now WPAR aware and reports memory information relevant to WPARs.

3.7.9 Updates for the topas command

The **topas** command is used to monitor and report system wide metrics about the state of the local system. The command displays its output in a 80x25 character-based display format or in a window of at least the same size on a graphical display. The **topas** command requires the bos.perf.tools and perfagent.tools filesets to be installed on the system.

The following enhancements have been made to this command to support WPAR specific metrics:

- The ability to display statistics for a given WPAR or WPAR list from the Global environment
- The ability to display organized statistics for all active WPARs from the Global environment
- The ability to run the command from within a WPAR and display statistics relevant to its isolated environment

Table 3-13 describes the updates made to this command for the support of WPARs.

Table 3-13 Option changes for topas command

Flag or argument	Behavior in WPAR	Behavior in Global
none	Executes the default display specific to the WPAR. Main Panel: Replace disk statistics by file systems; network statistics are provided per WPAR.	Executes normally with no changes from the previous versions of AIX.
"-@ WPARname"	Fails with a usage message as the -@ WPARname option is made illegal inside the WPAR.	Executes the default display specific to the WPAR. Main Panel: Replace disk statistics by file systems; network statistics are provided per WPAR.
"-P" Processes Screen	Executes normally displaying the processes window for the WPAR.	Executes normally with no changes from the previous versions of AIX.
"-D" Disk Screen	Fails with a usage message as the -D option is made illegal inside the WPAR.	Executes normally with no changes from the previous versions of AIX.
"-C" CEC Screen	Fails with a usage message as the -C option is made illegal inside the WPAR.	Executes normally with no changes from the previous versions of AIX.

Flag or argument	Behavior in WPAR	Behavior in Global
“-L” Partition Screen	Fails with a usage message as the -L option is made illegal inside the WPAR.	Executes normally with no changes from the previous versions of AIX.
“-W” WLM Screen	Fails with a usage message as the -W option is made illegal inside the WPAR.	Executes normally by displaying WLM classes and active WPARs along with the sub-process panel.
“-F” Filesystem Screen	Displays statistics about the file systems belonging to the WPAR.	Executes normally by displaying the statistics for the file system that belongs to the Global environment and all WPARs, running in the system, tagged with their respective names.
“-R” CEC-Recording	Fails with a usage message as the -R option is made illegal inside the WPAR.	Executes normally with no changes from the previous versions of AIX.

Figure 3-5 on page 87 demonstrates the output of the **topas** command ran from the Global environment.

```
Topas Monitor for host:    server5             EVENTS/QUEUES    FILE/TTY
Fri Oct  5 01:31:54 2007   Interval:  2        Cswitch     191  Readch        6
                                               Syscall      48  Writech     218
Kernel    0.8   |#                          |  Reads         0  Rawin         2
User      0.1   |#                          |  Writes        1  Ttyout      218
Wait      0.0   |                           |  Forks         0  Igets         0
Idle     99.1   |###########################|  Execs         0  Namei         1
Physc =  0.00                  %Entc=    1.4   Runqueue    0.0  Dirblk        0
                                               Waitqueue   0.0
Network  KBPS   I-Pack  O-Pack   KB-In  KB-Out                  MEMORY
Total     0.7      6.5     2.0     0.4     0.4 PAGING           Real,MB    1024
                                               Faults        0  % Comp     63.3
Disk    Busy%      KBPS     TPS KB-Read KB-Writ Steals       0  % Noncomp  17.7
Total     0.0       0.0     0.0     0.0     0.0 PgspIn       0  % Client   17.7
                                               PgspOut       0
FileSystem         KBPS     TPS KB-Read KB-Writ PageIn       0  PAGING SPACE
Total               0.0     0.0     0.0     0.0 PageOut      0  Size,MB     512
                                               Sios          0  % Used      1.1
WPAR                    CPU%    Mem%  Blk-I/O%                  % Free     99.9
mywpar1                    0       3        0  NFS (calls/sec)
                                               SerV2         0  WPAR Activ    1
                                               CliV2         0  WPAR Total    1
Name            PID  CPU%  PgSp Wpar           SerV3         0  Press: "h"-help
topas        311544   0.1   1.9 Global         CliV3         0         "q"-quit
wlmsched      65568   0.0   0.5 Global
getty        192662   0.0   0.5 Global
gil           61470   0.0   0.9 Global
java         303232   0.0  46.2 Global
rmcd         176302   0.0   2.6 Global
lockd-1      352270   0.0   1.2 Global
sendmail     188566   0.0   1.3 Global
rpc.lock     135326   0.0   1.2 Global
sendmail     405678   0.0   1.3 mywpar1
java         307348   0.0  16.1 Global
netm          57372   0.0   0.4 Global
xmgc          49176   0.0   0.4 Global
syncd        127100   0.0   0.5 Global
xmwlm        291008   0.0   8.6 Global
```

Figure 3-5 The topas command output in a WPAR environment

As shown in Figure 3-5, the **topas** command is now WPAR aware and reports relevant WPAR information (*circled*).

3.7.10 Updates for the tprof command

The **tprof** command is used to report CPU usage for individual programs and the system as a whole. This command is a useful tool for anyone with a JAVA, C, C++, or FORTRAN program that might be CPU-bound and who wants to know which sections of the program are most heavily using the CPU.

The following enhancements have been made to this command to support WPAR specific metrics:

- The ability to filter processes for a given WPAR or WPAR list from the Global environment
- The ability to display organized statistics for all active WPARs from the global environment

- The ability to run the command from within a WPAR and display statistics relevant to its isolated environment

> **Important:** In order to use this command within a WPAR, trace privileges must be enabled in the WPAR. Refer to "Enabling trace" on page 63.

Table 3-14 describes the updates made to this command for support of WPARs.

Table 3-14 Option changes for tprof command

Flag or argument	Behavior in WPAR	Behavior in Global
"-@ Wparlist"	Fails with a usage message as the -@ Wparlist option is made illegal inside the WPAR.	Prints the relevant information for a given WPAR only. If the specified WPAR does not exist or is not active, then it fails with a `workload partition not found` message unless the workload partition name is Global.
"-@ ALL"	Fails with a usage message as the -@ ALL option is made illegal inside the WPAR.	Executes normally and prints the summary of all WPARs. A workload partition name is displayed for each record.
"-@"	Fails with a usage message as the -@ option is made illegal inside the WPAR.	Executes normally and prints the process tree with related the WPARs and Global dependency. A workload partition name is displayed for each record.

The following example demonstrates the sleep.prof file output of the **`tprof -x sleep 10`** command when ran from within the mywpar1 WPAR:

```
Configuration information
=========================
System: AIX 6.1 Node: mywpar1 Machine: 00C0F6A04C00
Tprof command was:
    tprof -x sleep 10
Trace command was:
    /usr/bin/trace -ad -M -L 66476851 -T 500000 -j
00A,001,002,003,38F,005,006,134,210,139,5A2,5A5,465,234,5D8, -o
sleep.trc
Total Samples = 3
Traced Time = 10.04s (out of a total execution time of 10.04s)
```

```
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
<

Process                               Freq  Total Kernel   User Shared  Other
=======                               ====  ===== ======   ==== ======  =====
/usr/bin/sh                              1  33.33  33.33   0.00   0.00   0.00
/usr/bin/trcstop                         1  33.33  33.33   0.00   0.00   0.00
PID-1                                    1  33.33  33.33   0.00   0.00   0.00
=======                               ====  ===== ======   ==== ======  =====
Total                                    3 100.00 100.00   0.00   0.00   0.00

Process                    PID      TID  Total Kernel   User Shared  Other
=======                    ===      ===  ===== ======   ==== ======  =====
/usr/bin/sh             450812   647309  33.33  33.33   0.00   0.00   0.00
/usr/bin/trcstop        450812   647309  33.33  33.33   0.00   0.00   0.00
PID-1                       -1   647305  33.33  33.33   0.00   0.00   0.00
=======                    ===      ===  ===== ======   ==== ======  =====
Total                                   100.00 100.00   0.00   0.00   0.00
```

As shown in the previous example, the **tprof** command is now WPAR aware and reports the processes' usage relevant to WPARs.

3.7.11 Updates for the vmstat command

The **vmstat** command is used to monitor and display VMM, I/O, and CPU utilization statistics. VMM and CPU statistics are frequently used by system administrators to analyze system throughput and potential bottlenecks.

The following enhancements have been made to the **vmstat** command to support WPAR specific metrics:

- The ability to filter VMM and CPU activities for a given WPAR from the Global environment
- The ability to display organized statistics for all active WPARs from the global environment
- The ability to run the command from within a WPAR and display statistics relevant to its isolated environment

Table 3-15 describes the updates made to the **vmstat** command performance tool for support of WPARs.

Table 3-15 Option changes for vmstat command

Flag or argument	Behavior in WPAR	Behavior in Global
None	Executes the default report and displays an @ above the metrics specific to the WPAR.	Executes normally with no changes from the previous versions of AIX.
"-@ Wparname"	Fails with a usage message as the -@ Wparname option is made illegal inside the WPAR.	Prints relevant information for the given WPAR only. If the specified WPAR does not exist or is not active, then it fails with the `workload partition not found` message unless the workload partition name is Global.
"-@ ALL"	Fails with a usage message as the -@ ALL option is made illegal inside the WPAR.	Executes normally and prints the summary of all WPARs. A workload partition name is displayed for each record.
"-i"	Fails with a usage message as the -i option is made illegal inside the WPAR.	Executes normally and prints interrupt information associated with Global. This option cannot be used with the -@ option.
"-s"	Displays an @ by the side of the metrics associated with the WPAR	Executes normally and prints the sum structure and count of paging events associated with the Global environment. If it is used in combination with any of thee -@ options, it will print the relevant information summarizing the specified WPARs.

Flag or argument	Behavior in WPAR	Behavior in Global
"-v"	Displays an @ by side of the metrics associated with the WPAR	Executes normally and prints the VMM statistics associated with the Global environment. If used in combination with any of thee -@ options, it will print the relevant information summarizing the specified WPARs

Example 3-4 demonstrates the output of the -v option when used in the Global environment combined with the -@ ALL option.

Example 3-4 VMM statistics combined output from the global environment

```
# vmstat -@ ALL -v
WPAR:System
              262144 memory pages
              232510 lruable pages
               82435 free pages
                   1 memory pools
               83218 pinned pages
                80.0 maxpin percentage
                 3.0 minperm percentage
                90.0 maxperm percentage
                10.3 numperm percentage
               24041 file pages
                 0.0 compressed percentage
                   0 compressed pages
                10.3 numclient percentage
                90.0 maxclient percentage
               24041 client pages
                   0 remote pageouts scheduled
                   0 pending disk I/Os blocked with no pbuf
                   0 paging space I/Os blocked with no psbuf
                2484 filesystem I/Os blocked with no fsbuf
                   0 client filesystem I/Os blocked with no fsbuf
                   0 external pager filesystem I/Os blocked with no fsbuf
                   0 Virtualized Partition Memory Page Faults
                0.00 Time resolving virtualized partition memory page faults
----------------------------------------
WPAR: Global
              262144 memory pages
              232510 lruable pages
               82435 free pages
               83118 pinned pages
               23094 file pages
```

```
                    0 compressed pages
                23094 client pages
                    0 remote pageouts scheduled
                    0 paging space I/Os blocked with no psbuf
                 2484 filesystem I/Os blocked with no fsbuf
                    0 client filesystem I/Os blocked with no fsbuf
                    0 external pager filesystem I/Os blocked with no fsbuf
                    0 Virtualized Partition Memory Page Faults
----------------------------------------
WPAR: mywpar1
                  100 pinned pages
                  947 file pages
                    0 compressed pages
                  947 client pages
                    0 remote pageouts scheduled
                    0 paging space I/Os blocked with no psbuf
                    0 filesystem I/Os blocked with no fsbuf
                    0 client filesystem I/Os blocked with no fsbuf
                    0 external pager filesystem I/Os blocked with no fsbuf
                    0 Virtualized Partition Memory Page Faults
#
```

As shown in Example 3-4 on page 91, this combined command shows a breakdown summary report for the entire system, Global, and all active WPARs in the LPAR.

Note: Filtering options using the -@ will only show information for *active* WPARs. Some reports will not show a different output if there are no active WPARs in the system.

3.8 Standard command updates for WPAR support

In order to support and filter WPAR relevant information, many standard AIX commands have been enhanced to support workload partitions. Many of the commands have different behaviors inside a WPAR and in the Global environment. Table 3-16 on page 93 provides a summarized list of these commands and their changes.

Table 3-16 Command updates for WPAR support

Command	Flags or argument	Behavior in WPAR	Behavior in Global
acctcom	- @ wparname	Fails with a usage message as the -@ wparname option is made illegal inside the WPAR.	Executes normally. displaying accounting records for specified WPAR.
	- @ no argument	Fails with a usage message as the -@ option is made illegal inside the WPAR.	Executes normally. displaying accounting records for all WPARs. A WPAR name is displayed for each record.
clogin	wparName [-l user] [command [args]]	Not allowed within a WPAR.	Prompts for a password and runs a command in the WPAR or login if no command is specified.
df	All options	Displays information about WPAR mounted file systems only. Paths are displayed relative to the WPAR root.	Displays information about all the file systems. The paths are absolute.
domainname	None	Displays the domain name of the WPAR.	Displays the domain name for the system.
	{new domain name}	If executed by root, it sets the domain name of WPAR.	If executed by root, it sets the domain name of the Global environment
hostid	None	Displays the host ID of WPAR.	Displays the host ID of the Global environment.
	{IP address| hex number}	If executed by root, it sets the host ID of WPAR.	If executed by global root, it sets the host ID of the Global environment.
hostname	None	Displays the host name of WPAR.	Displays the host name of the system.
	{newhostname}	If executed by root and the host name privilege is allowed for WPAR, it sets the host name of WPAR.	If executed by root, it sets the host name of the Global environment.

Command	Flags or argument	Behavior in WPAR	Behavior in Global
`ifconfig`	All display options (-a -l)	Displays information about the WPAR.	Displays information about the Global environment.
`ioo`		Non-functional in WPAR.	No change in behavior.
`ipcrm`	None	Removes IPC objects associated with the WPAR.	Removes IPC objects associated with the Global environment.
	"-@ wparname"	Invalid unless the WPAR name = Global.	Removes IPC objects associated with the WPAR wparname.
`ipcs`	None	Displays information about the IPC objects created by the processes within the WPAR.	Displays information about the IPC objects created by the Global environment processes. No WPAR associated objects are displayed.
	"-@"	Displays IPC information within the WPAR.	Displays information about all IPC objects in the system. The name of the WPAR associated with the object is listed.
	"-@ wparname"	Displays no IPC information unless wparname = Global.	Displays information about IPC objects associated with the processes within the specified WPAR.
`mkclass`	All options	This command will only update the /etc/wlm directory. It will fail while updating the kernel data.	No change in behavior.
`mount`	None	Displays only the WPAR mounted file systems relative to the WPAR root.	Displays all the mounted file systems with absolute paths.
	With arguments	Only NFS mounts without CacheFS™ are allowed. nosuid and nodev are forced.	No change in behavior.

Command	Flags or argument	Behavior in WPAR	Behavior in Global
`netstat`	All options except -c, -C, -g, -m, -M, -P, -r, -v, -Z	Fails with usage message as the -c, -C, -g, -m, -M, -P, -r, -v, and -Z options are made illegal inside the WPAR.	Displays information about the whole system.
	new Argument "-@ wparname”	Fails with a usage message as the -@ wparname option is made illegal inside the WPAR.	Displays either connection or address information for the specified WPAR.
`nfso`		Only a subset of tunables are displayed within a WPAR.	No change.
`no`		Fails with a usage message. The -a option executes normally.	Executes normally if user has the correct privilege.
`projctl`	All options except qproj(s)	Fails with a "not owner" message. qproj(s) executes normally.	Executes normally if the user has the correct privilege.

Command	Flags or argument	Behavior in WPAR	Behavior in Global
`ps`	"-e"	Displays everything within the WPAR.	Displays everything within the system. Processes are not screened from view unless a specific -@ <wparname> is also included.
	"-@"	Displays the process information for processes in the WPAR. The WPAR name is included in the output.	Displays the process information for all processes in the system. The name of the WPAR is displayed in the output.
	"-@ wparname"	Displays no process information unless wparname = Global. Global case displays information about the processes within the WPAR. The name of the WPAR is provided in the output.	Displays information about the processes associated with the WPAR named wparname. The name of the WPAR is provided in the output.
	"-o wpar"	Produces a WPAR name header and the name of the WPAR associated with the process. This name is always Global.	Produces a WPAR name header and the name of the WPAR in which the process is executing.
`schedo`		Non-functional in WPAR.	No change in behavior.
`uname`	"-n"	Displays the name of the WPAR.	Displays the node name of the system.
`vmo`		Non-functional in WPAR.	No change in behavior.
`wlmstat`		The -B option is not allowed within a WPAR.	No change in behavior.
	"-@"	Will not work in the WPAR.	Will display data for a meta class (WPAR class).

Command	Flags or argument	Behavior in WPAR	Behavior in Global
`wlmtune`	All options	Not allowed within a WPAR.	No change in behavior.
`wlmcntrl`	All options	Not allowed within a WPAR.	No change in behavior.

3.9 Network file system support for WPARs

In this section, we discuss Network File System (NFS) interface implementation and support for WPARs.

3.9.1 Overview

Most applications running within a WPAR will operate with no difference than running in previous versions of AIX. This is because, within the WPAR, applications have a private execution environment isolated in terms of processes, signals, and file system space. They run with unique security privileges and have dedicated network addresses, and interprocess communication is restricted to processes executing in the same WPAR.

When AIX is installed and started, a special workload partition is created. This WPAR is referred to as the Global partition, which is the same as a default single instance of the OS.

Although WPARs are isolated, it is a common practice for network resources to be shared across different systems. AIX Network File Systems (NFS) allows for the distribution of local file systems in a server for the use of remote systems, LPARS, and now in AIX V6.1, WPARs.

The following list summarizes the NFS features enabled for WPAR support:

- Operation of NFS Version 2, 3, and Version 4 clients, AutoFS and CacheFS within a WPAR including the Global environment
- Implementation of per WPAR NFS client statistics and tunables
- Implementation of per WPAR NFS commands
- Implementation of per WPAR CacheFS commands
- Loading of NFS, AutoFS, and CacheFS kernel extensions from a WPAR
- RAS enhancements to NFS, AutoFS, and CacheFS for field support

3.9.2 NFS user interface

This section discusses the different NFS user interfaces updates for WPARs. Refer to the product documentation and man pages for a detailed description of parameters and their usage for each one of the commands discussed in this section.

Updates for the nfso command

The **nfso** command is used by NFS versions 2, 3, and 4 to set parameters and configuration options.

The **nfso** command has been changed to accept an additional argument, -@, that when invoked in the Global environment can be used to set or retrieve an optional value for a specific WPAR. Only a subset of **nfso** options within a WPAR are tunable for security reasons. For example, the command below demonstrates the output list of NFS tunables and their values for the WPAR named *mywpar1* when **nfso -a** is executed within the WPAR:

```
# nfso -a
           client_delegation = 0
                 nfs_rfc1323 = 1
      nfs_use_reserved_ports = 0
    nfs_v4_fail_over_timeout = 0
             utf8_validation = 1
```

Updates for the nfsstat command

The **nfsstat** command is used to display a wide range of NFS statistics. This command has been updated to accept an additional argument, -@, when invoked in the Global environment in order to obtain statistics for a specific WPAR. In the global partition when no "@" option is used, cumulative statistics for all workload partitions, including the Global, will be reported. The -@ option is not valid within WPARs.

Updates for the nfs4cl command

The **nfs4cl** command displays and modifies current NFSv4 statistics and properties. The command has been updated to work within a WPAR. When invoked from within a WPAR, it would display or modify the current NFSv4 statistics and properties for that particular WPAR.

> **Note:** The -@ parameter is not valid within the Global environment for this command.

3.9.3 AutoFS user interface

AutoFS relies on the use of the **automount** command to propagate the automatic mount configuration information to the AutoFS kernel extension and start the **automountd** daemon. The **automount** command is used as an administration tool for AutoFS. It installs AutoFS mount points and associates an automount map with each mount point.

Each WPAR runs a separate instance of the user mode daemon. The **automount** command works as in the previous version for both the Global environment and WPARs.

> **Note:** These commands only work on a System WPAR.

3.9.4 CacheFS user interface

This section discusses the different CacheFS user interfaces updates for WPARs. Refer to the product documentation and man pages for a detailed description of parameters and their usage for each one of the commands discussed in this section.

> **Note:** The -@ parameter is not valid within the Global environment for any of the CacheFS commands.

Updates for the cfsadmin command

The **cfsadmin** command provides maintenance tasks for disk space used for caching file systems. It allows for the following functions:

- Cache creation
- Deletion of cached file systems
- Listing of cache contents and statistics
- Resource parameter adjustment when the file system is un-mounted.

The **cfsadmin** command is now enhanced so it can be executed from within the WPAR. When executed within a WPAR, the command will only allow the -c, -d, -l, and -s options. The -o parameter is only allowed from the Global environment, as it affects global CacheFS parameters.

Updates for the cachefsstat command

The **cachefsstat** command displays statistical information about a cache file system. The command is now enhanced so it can be used within a WPAR and display statistics relevant to its environment.

Updates for the cachefslog command

The **cachefslog** command controls and displays the logging of a cache file system. This command has been enhanced to work within a WPAR and display information relevant to its environment.

Updates for the cachefswssize command

The **cachefswssize** command displays the work size for a cache file system. This command has been enhanced to work within a WPAR and display information relevant to its environment.

Packaging

The CacheFS functionality is not installed with the default AIX V6.1 system installation. The following fileset needs to be installed in order for CacheFS commands to be available:

bos.net.nfs.cachefs Supports CacheFS functionality.

3.9.5 Continuous availability enhancements for NFS

This section discusses the different continuous availability enhancements to support WPARs.

Tracing

Tracing capabilities have been extended to support System type WPARs. Details for using and enabling trace within a system WPAR are covered in 3.4, “System trace support” on page 57.

KDB support

The **kdb** command has been enhanced for the user mode KDB module to extract per-WPAR data structures from the Global environment.

The NFS KDB module has been updated with the following commands:

nfsgv Displays the global variable data structure. This command accepts the following syntax:

```
nfsvar variable-name|subsystem-name|all
```

nfswpar [n] Displays the addresses of the per-WPAR variable structures when used without arguments. A per-WPAR variable structure (corral ID, or structure address) can be passed as an argument.

nfsvar Displays a NFS global variable, a group of variables, or all the variables. This command accepts the following syntax:

`nfsvar variable-name|subsystem-name|all`

With the argument of all, it displays all variables. With the name of a variable as an argument, it displays that variable. With the group name (for example, krpc, clnt, or klm) as an argument, it displays the variable related to that subsystem.

4

Continuous availability

This chapter discusses the topics related to continuous availability, including:

4.1 Storage protection keys

Memory overlays and addressing errors are a difficult problem to diagnose and service. This problem is intensified and becoming more prominent by growing software size and complexity.

A new POWER6™ processor feature called storage protection keys, or storage keys for short, provides the hardware foundation to prevent inadvertent memory overlays in both the kernel and the application space. Storage protection keys are a new and strategic element of the AIX continuous availability framework.

AIX 5L Version 5.3 Technology Level 06 (5300-06) introduced the storage protection keys application programming interface (API) for user space applications that assists application programmers in utilizing the hardware storage protection keys on IBM System p POWER6 processor-based servers running this technology level. Additional background information about this user-mode storage key exploitation and an in-depth discussion of the API's use can be found in the white paper *Storage Protection Keys on AIX Version 5.3*, found at:

http://www.ibm.com/systems/p/library/wp_aix_lit.html

Beginning with AIX V6.1, the operating system kernel and kernel extensions inherently exploit the hardware storage keys for enhanced memory allocation and memory access reliability characteristics. To externalize this kernel-mode storage key support, AIX V6.1 also provides the kernel-mode storage protection key API, enabling kernel extension programmers to write code that makes use of the hardware storage protection keys.

Storage-keys were introduced into the PowerPC® architecture to provide memory isolation while still permitting software to maintain a flat address space. The concept was adopted from the z/OS® and S/390® systems. Storage-keys allow an address space to be assigned context specific protection. Access to the memory regions can be limited to prevent or identify illegal storage references.

Under AIX V6.1, storage-keys are used to capture bad storage references of the kernel and kernel extension that previously overwrote memory, thereby providing a transparent protection mechanism.

Additional background information about the new kernel-mode storage protection mechanism, and how to take advantage of storage protection keys to improve the Reliability, Availability, and Serviceability (RAS) characteristics of an existing device driver or kernel extension, can be found in the white paper *Key-enabling kernel extensions for the IBM AIX Version 6.1 operating system*, found at:

`http://www.ibm.com/developerworks/aix/library/au-keykernext/index.html`

4.2 Component trace and RTEC adoption

The AIX enterprise Reliability Availability Serviceability (RAS) infrastructure defines a component definition framework. This framework supports three distinct domains:

- Runtime Error Checking (RTEC)
- Component Trace (CT)
- Component Dump (CD)

This framework is shown in Figure 4-1.

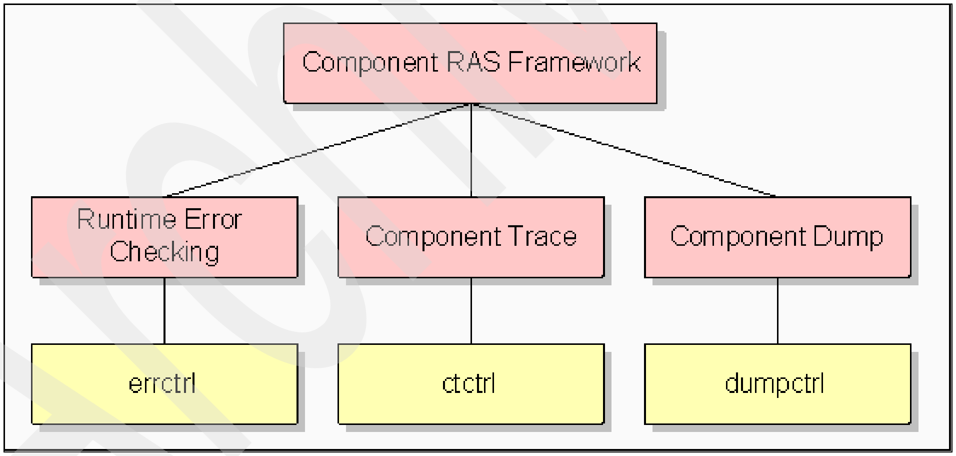


Figure 4-1 Component RAS Framework overview

AIX 5L Version 5.3 with the TL 5300-05 Technology Level package enabled the first operating system components to exploit the runtime error checking and the component trace services. AIX V6.1 introduces the third domain, the component dump services, and significantly increases the number of components that utilize the runtime error checking and component trace services. This section describes the operating system components that are enhanced in AIX V6.1 to leveraged the component trace and runtime error checking framework. The following provides an overview of the affected areas and the related base component names are given in parenthesizes.

- Areas of component trace and runtime error checking adoption:
  - Virtual Memory Manager (vmm and ipc)
  - AIX storage device drivers (scdisk, sisraid_dd, and sissas_dd)
  - Virtual SCSI disk drivers (vscsi_initdd)
  - Multiple Path I/O and AIX default path control module (mpio# and pcm#)
  - InfiniBand® device driver (if_ib, gxibdd, icmdd, and tsibdd)
  - LAN device driver (vioentdd, goentdd, headd, and kngentdd)
  - TCP kernel and netinet kernel extension (netisr)
- Areas of component trace adoption:
  - Internet Protocol Security (ipsec)
  - PCI device driver (pci)
  - Virtual bus device driver (vdev)
  - USB system device driver (usb_system)
  - USB audio device driver (usb_audio)
  - 2D graphics device drivers (lanaidd and cortinadd)
- Areas of runtime error checking adoption:
  - System loader (ldr)
  - NFS version 4 (nfs.nfs4)
  - Cache File System (cachefs)
  - Watchdog timer (watchdog)
  - System memory allocator (alloc)

(The # character denotes a place holder for configuration dependent integer values.)

These components are organized following a hierarchy by base component and subcomponents. Component names are built upon this hierarchy: <base component name>.<specific subcomponents>. These components belong to a *type/subtype* classification and have RAS properties.

The following example shows the component/subcomponent hierarchy for the tracing properties of the lfs component:

```
# ctctrl -c lfs -q -r
---------------------------------------+-------+-------+-------+---------------
                                       | Have  |Mem Trc|Sys Trc| Buffer size
     Component name                    | alias | /level| /level| /Allocated
---------------------------------------+-------+-------+-------+---------------
lfs                                    |   NO  |  ON/3 |  ON/3 |         0/ NO
     filesystem
          ._0                          |   NO  |  ON/3 |  ON/3 |         0/ NO
          .__1                         |   NO  |  ON/3 |  ON/3 |         0/ NO
          ._admin_9                    |   NO  |  ON/3 |  ON/3 |         0/ NO
          ._home_8                     |   NO  |  ON/3 |  ON/3 |         0/ NO
          ._opt_11                     |   NO  |  ON/3 |  ON/3 |         0/ NO
          ._proc_10                    |   NO  |  ON/3 |  ON/3 |         0/ NO
          ._tmp_5                      |   NO  |  ON/3 |  ON/3 |         0/ NO
          ._usr_2                      |   NO  |  ON/3 |  ON/3 |         0/ NO
          ._var_4                      |   NO  |  ON/3 |  ON/3 |         0/ NO
     .kdm                              |   NO  |  ON/3 |  ON/3 |         0/ NO
     .pile                             |   NO  |  ON/3 |  ON/3 |         0/ NO
```

In order to be used by the kernel when the system needs to communicate between various commands to each component (RAS callbacks), these components must be registered and unregistered to AIX.

There are two kernel services that are exported and can be called from the process environment with the ras_register() and ras_unregister() kernel service calls.

The following code is an example of registering a parent(base) component: It creates and registers a base component named *ethernet* of the network type with the Ethernet subtype:

```
ras_block_t rasb_eth;
kerrno_t err;
...lines missing for clarity
err=ras_register( &rasb_eth, “ethernet”,NULL,
RAS_TYPE_NETWORK_ETHERNET, “All ethernet devices”, RASF_TRACE_AWARE,
eth_call back, NULL);
```

Note: The flag RASF_TRACE_AWARE indicates what type of RAS systems this component is aware of. With RASF_TRACE_AWARE, this component is a tracing component.

The type/subtype field is associated with the component at registration time. The component characteristics can be modified from their default properties by the ras_control() exported kernel service call. For example, one component characteristic can be the size of a buffer area used to report error data or traced data or dump data.

To put the component on a *usable state* by the AIX kernel, the customization step, which loads the reboot persistent customized properties, is mandatory. The customization step is realized by calling the ras_customize() exported kernel service call.

The following example modifies some default properties of the previous *ethernet* component and then executes the mandatory customization call:

```
ras_block_t rasb_eth;
kerrno_t err;
...lines missing for clarity

/* set a buffer size (default size is 0) */
err=ras_control(rasb_eth, RASCT_SET_MEMBUFSIZE, size, 0);

/* allocate a private buffer size */
err=ras_control(rasb_eth, RASCT_SET_ALLOC_BUFFER, 0, 0);

/* activate memory trace mode */
err=ras_control(rasb_eth, RASCT_SET_MEMTRC_RESUME, 0, 0);

/* customization step to be usable component */
err=ras_customize(rasb_eth);
```

Persistence of component attributes

The three control commands **errctrl**, **ctctrl**, and **dumpctrl** are used to modify the RAS attribute values of individual components. With AIX Version 6, these commands are enhanced so that RAS attribute values can be specified for components not yet created. In addition, it will be possible to specify the RAS attribute values that will persist across reboots.

Persistence of component attributes is required for two reasons:

- RAS components can be created dynamically, such as when a file system is mounted. A method is needed to specify custom RAS attributes for components that have not yet been created, so that the desired RAS property takes effect as soon as the component is created.

- RAS components are also created before a system administrator can log in and run a control command. By allowing customized attribute values to be specified as part of the boot image, all components can be controlled, included those created early in the boot process.

This persistence capability is also essential for allowing an administrator to specify customizations required for a given configuration. Persistence is specified by using the -P or -p flag with the control commands:

The -P flag Specifies attribute values that apply to the next reboot. The -P flag results in the modification of the /var/adm/ras/raspertune file. Lines are added to or deleted from the file. In addition, the **`bosboot`** command processes the raspertune file.

The -p flag Specifies attribute values that apply to newly-created components. It will not affect an existing component.

Both flags can be used at the same time, with the expected result.

The -n flag Specifies attribute values to apply immediately to existing components. To apply changes to both current and newly created components, use the -n and -p flags.

The -x flag Specifies a permanent persistence specification that must be deleted. The -x flag must be used with -P or -p.

For example, the following command sets the error checking level to normal for the hdisk0 component with its alias -l flag:

```
# errctrl -p -l hdisk0 errchecknormal
```

As RAS components are organized under a hierarchy, the specified attributes can be set recursively to all component descendants with the -r flag or to all ancestors with the -u flag.

For example, the following command sets the error checking level to *minimal* for the `nfs` component and its descendants:

```
# errctrl -p -r -c nfs errcheckminimal
```

The following command set the error checking level to *detail* for the nfs.nfs4.nfs4_server component and its ancestors:

```
# errctrl -p -u -c nfs.nfs4.nfs4_server errcheckdetail
```

The following sections detail the enhancements in AIX Version 6.

4.2.1 VMM component trace and RTEC adoption

In previous AIX releases, the VMM does tracing using either system trace or light weight memory trace. AIX V6.1 extends the component trace adoption to the virtual memory manager (VMM) kernel subsystem and provides a VMM component tree for the component trace domain of the AIX enterprise RAS infrastructure. The related base component is named *vmm* and the integration into the component trace framework enables both the memory trace mode (private or light weight memory trace) and the user trace mode (system trace) for the new base component and its sub-components. In AIX V6.1, the VMM component tree is also utilized by the runtime error checking (RTEC) domain of the AIX enterprise RAS infrastructure. The VMM RTEC adoption also extends to the Inter Process Communication (IPC) services for which the base component name *ipc* has been defined.

The VMM component hierarchy of a given AIX configuration and the current settings for the memory trace mode (private or light weight memory trace) and the user trace mode (system trace) can be listed by the **ctctrl** command. The **ctctrl** command also allows you to modify the component trace related configuration parameters:

```
hhaix6:root:/root # ctctrl -c vmm -q -r
---------------------------------------+-------+-------+-------+---------------
                                       | Have  |Mem Trc|Sys Trc| Buffer size
     Component name                    | alias | /level| /level| /Allocated
---------------------------------------+-------+-------+-------+---------------
vmm
     .dr                               |  NO   | ON/3  | ON/3  |     65536/YES
     .internal                         |  NO   | ON/3  | ON/3  |      4096/YES
     .memp
          .mempLRU                     |  NO   | OFF/3 | ON/3  |         0/ NO
               .mempLRU0               |  YES  | ON/3  | ON/3  |     16384/YES
               .mempLRU1               |  YES  | ON/3  | ON/3  |     16384/YES
          .mempPSMD                    |  NO   | OFF/3 | ON/3  |         0/ NO
               .mempPSMD0              |  YES  | ON/3  | ON/3  |     16384/YES
               .mempPSMD1              |  YES  | ON/3  | ON/3  |     16384/YES
     .pdt                              |  YES  | OFF/3 | ON/3  |         0/ NO
          .pdt0                        |  YES  | ON/3  | ON/3  |         0/ NO
          .pdt80                       |  YES  | ON/3  | ON/3  |         0/ NO
          .pdt81                       |  YES  | ON/3  | ON/3  |         0/ NO
          .pdt82                       |  YES  | ON/3  | ON/3  |         0/ NO
          .pdt83                       |  YES  | ON/3  | ON/3  |         0/ NO
          .pdt84                       |  YES  | ON/3  | ON/3  |         0/ NO
          .pdt85                       |  YES  | ON/3  | ON/3  |         0/ NO
          .pdt86                       |  YES  | ON/3  | ON/3  |         0/ NO
          .pdt87                       |  YES  | ON/3  | ON/3  |         0/ NO
          .pdt88                       |  YES  | ON/3  | ON/3  |         0/ NO
          .pdt89                       |  YES  | ON/3  | ON/3  |         0/ NO
```

```
        .pdt8A                              |  YES  |  ON/3 |  ON/3 |        0/ NO
        .pdtbufx                            |   NO  | OFF/3 |  ON/3 |        0/ NO
             .pdtbufx0                      |  YES  |  ON/3 |  ON/3 |    65536/YES
   .services                                |   NO  |  ON/3 |  ON/3 |        0/ NO
```

The component tree elements of the previous listing are defined as follows (The # character denotes a place holder for given integer values and the XXX character sequence denotes a place holder for a given alphanumeric label):

vmm Base component for virtual memory manager kernel subsystem. This component has no private trace buffer.

- **.dr** Component for memory dynamic re-configuration (DR) trace. The DR component has a 64 KB default private trace buffer.
- **.internal** Component for internal VMM trace. This component has no private trace buffer.
- **.memp** Parent component for memory pools. This component has no private trace buffer.
  - **.mempXXX** Dynamic component for individual memory pool types. AIX V6.1 supports parent components for VMM pager (LRU) memory pools and Page Size Management Daemon (PSMD) pools. LRU and PSMD are the respective memory pool IDs that replace the XXX place holder.
  - **.mempLRU#** Subcomponent for LRU memory pool trace that has an associated 16 KB private buffer.
  - **.mempPSMD#** Sub-component for PSMD memory pool trace that has an associated 16 KB private buffer.
- **.pdt** Parent component for paging space devices and their related paging device tables (PDT). This component has no private trace buffer.
  - **.pdt#** Dynamic component for individual paging devices and their related PDTs. This component has no private trace buffer.
  - **.pdtbufx** Parent component for I/O tracking bufx structures.
- **.services** Component for VMM kernel services. This component has no private trace buffer.

Since there can be as many as 256 memory pools, the maximum amount of pinned memory consumed by VMM component private buffers at their default size is 256*2*16 KB for the memory pool buffers + 64 KB for the DR buffer, for a total of about 8.5 MB. In most system configurations, the amount will be substantially less than this, since the number of memory pools scales with the number of CPUs: By default, there is one memory pool per eight CPUs.

The RTEC vmm component hierarchy of a given AIX configuration and the current settings for error checking level, disposition for low-severity errors, and disposition for medium-severity errors can be listed by the **errctrl** command. The **errctrl** command also allows you to modify the runtime error checking related configuration parameters. The following **errctrl** command output shows that the default error checking level for all VMM components is normal (level=3), that low-severity errors are ignored (LowSevDis=48), and medium-severity errors are logged (collect service data and continue) (MedSevDisp=64):

```
hhaix6:root:/root # errctrl -c vmm -q -r
---------------------------------------------+-------+-------+-------+--------
                                             | Have  |ErrChk |LowSev |MedSev
    Component name                           | alias | /level| Disp  | Disp
---------------------------------------------+-------+-------+-------+--------
vmm
     .adsp                                   |  NO   | ON /3 |  48   |  64
     .dr                                     |  NO   | ON /3 |  48   |  64
     .frs                                    |  YES  | ON /3 |  48   |  64
          .frs0                              |  YES  | ON /3 |  48   |  64
          .frs1                              |  YES  | ON /3 |  48   |  64
          .frs2                              |  YES  | ON /3 |  48   |  64
          .frs3                              |  YES  | ON /3 |  48   |  64
     .internal                               |  NO   | ON /3 |  48   |  64
     .memp                                   |  YES  | ON /3 |  48   |  64
          .memp0                             |  YES  | ON /3 |  48   |  64
          .memp1                             |  YES  | ON /3 |  48   |  64
          .mempLRU                           |  NO   | ON /3 |  48   |  64
               .mempLRU0                     |  YES  | ON /3 |  48   |  64
               .mempLRU1                     |  YES  | ON /3 |  48   |  64
          .mempPSMD                          |  NO   | ON /3 |  48   |  64
               .mempPSMD0                    |  YES  | ON /3 |  48   |  64
               .mempPSMD1                    |  YES  | ON /3 |  48   |  64
     .pdt                                    |  YES  | ON /3 |  48   |  64
          .pdt0                              |  YES  | ON /3 |  48   |  64
          .pdt80                             |  YES  | ON /3 |  48   |  64
          .pdt81                             |  YES  | ON /3 |  48   |  64
          .pdt82                             |  YES  | ON /3 |  48   |  64
          .pdt83                             |  YES  | ON /3 |  48   |  64
          .pdt84                             |  YES  | ON /3 |  48   |  64
          .pdt85                             |  YES  | ON /3 |  48   |  64
          .pdt86                             |  YES  | ON /3 |  48   |  64
```

```
        .pdt87                                  |  YES  | ON /3 |  48   |  64
        .pdt88                                  |  YES  | ON /3 |  48   |  64
        .pdt89                                  |  YES  | ON /3 |  48   |  64
        .pdt8A                                  |  YES  | ON /3 |  48   |  64
        .pdt8B                                  |  YES  | ON /3 |  48   |  64
        .pdtbufx                                |  NO   | ON /3 |  48   |  64
             .pdtbufx0                          |  YES  | ON /3 |  48   |  64
   .power                                       |  NO   | ON /3 |  48   |  64
   .services                                    |  NO   | ON /3 |  48   |  64
   .vmpool                                      |  YES  | ON /3 |  48   |  64
        .vmpool0                                |  YES  | ON /3 |  48   |  64
        .vmpool1                                |  YES  | ON /3 |  48   |  64
   .wlm                                         |  NO   | ON /3 |  48   |  64
```

The RTEC ipc component hierarchy of a given AIX configuration and the current settings for error checking level, disposition for low-severity errors, and disposition for medium-severity errors can be listed by the **errctrl** command. The **errctrl** command also allows you to modify the runtime error checking related configuration parameters. The following **errctrl** command output shows that the default error checking level for all ipc components is normal (level=3), that low-severity errors are ignored (LowSevDis=48), and medium-severity errors are logged (collect service data and continue) (MedSevDisp=64):

```
hhaix6:root:/root # errctrl -c ipc -q -r
---------------------------------------------+-------+-------+-------+--------
                                             | Have  |ErrChk |LowSev |MedSev
    Component name                           | alias | /level| Disp  | Disp
---------------------------------------------+-------+-------+-------+--------
ipc
     .msg                                    |  YES  | ON /3 |  48   |  64
     .sem                                    |  YES  | ON /3 |  48   |  64
     .shm                                    |  YES  | ON /3 |  48   |  64
```

The vmm and the ipc components are also enabled for the component dump and live dump services of the AIX enterprise RAS infrastructure.

4.2.2 AIX storage device driver component trace and RTEC support

Beginning with AIX 5L V5.3 TL5, selected critical AIX storage device drivers started to exploit the AIX RAS infrastructure with regards to component naming and registration to the runtime error checking and component trace domains. AIX V6.1 enables three additional storage device drivers to exploit the component trace and the RTEC framework. Table 4-1 provides an overview of the AIX storage device drivers that utilize the AIX enterprise RAS infrastructure for trace and RTEC. The device drivers are listed by their base component names and the AIX release of introduction is provided.

Table 4-1 AIX storage device driver base component names

AIX release	Component name	Description
5300-05	scsidiskdd	CSI disk (scsidisk) device driver for Fibre Channel and iSCSI disks, except for FAStT (DS4000™)
5300-05	fcparray	Disk device driver for FAStT(DS4000)
5300-05	efcdd	Adapter device driver for Emulex Fibre Channel controllers
5300-05	efscsidd	SCSI protocol device driver for Emulex Fibre Channel controllers
6100-00	scdisk	Disk device driver for parallel SCSI disks and optical
6100-00	sisraid_dd	Adapter device driver for SIS based Ultra™ 320 SCSI and SCSI RAID controllers
6100-00	sissas_dd	Adapter device driver for SIS SAS RAID controller

The storage device driver component hierarchy of a given AIX configuration and the current settings for the memory trace mode (private or light weight memory trace) and the user trace mode (system trace) can be listed by the **ctctrl** command. The **ctctrl** command also allows you to modify the component trace related configuration parameters:

```
# ctctrl -c scdisk -q -r
# ctctrl -c sisraid_dd -q -r
# ctctrl -c sissas_dd -q -r
```

The RTEC storage device driver component hierarchy of a given AIX configuration and the current settings for error checking level, disposition for low-severity errors, and disposition for medium-severity errors can be listed by the **errctrl** command. The **errctrl** command also allows you to modify the runtime error checking related configuration parameters:

```
# errctrl -c scdisk -q -r
# errctrl -c sisraid_dd -q -r
# errctrl -c sissas_dd -q -r
```

4.2.3 Virtual SCSI device driver component trace and RTEC adoption

The AIX virtual SCSI client device driver was enhanced in AIX V6.1 to exploit the enterprise RAS infrastructure with regards to component naming and registration. In addition to registration, the driver also adds support for enterprise RAS component tracing and kernel runtime error checking. Component tracing and runtime error checking are referred to as domains supported by the AIX RAS infrastructure services.

During the virtual SCSI device driver configuration and initialization, the new base component *vscsi_initdd* is registered with the component trace framework. Each instance of a virtual adapter controlled by the virtual SCSI device driver becomes a sub-component specifying the base component (device driver) as the parent. The adapter sub-component is named vscsi#, where # designates a place holder for configuration dependent integer values. Finally, each open device on a given adapter defines a sub-component, specifying the adapter instance as the parent. An example hierarchy would be vscsi_initdd.vscsi0.lun8100000000000000, where lun8100000000000000 represents an hdisk instance that has been started (opened) on the client adapter vscsi0.

The virtual SCSI device driver component hierarchy of any given AIX configuration and the current settings for the memory trace mode (private or light weight memory trace) and the user trace mode (system trace) can be listed by the **ctctrl** command. The **ctctrl** command also allows you to modify the component trace related configuration parameters:

```
hhaix6:root:/root # ctctrl -c vscsi_initdd -q -r
-----------------------------------+-------+-------+-------+--------------
                                   | Have  |Mem Trc|Sys Trc| Buffer size
    Component name                 | alias | /level| /level| /Allocated
-----------------------------------+-------+-------+-------+--------------
vscsi_initdd
    .vscsi0                        |  YES  |  ON/3 |  ON/3 |      8192/YES
        .lun8200000000000000       |   NO  |  ON/3 |  ON/3 |      8192/YES
        .lun8300000000000000       |   NO  |  ON/3 |  ON/3 |      8192/YES
```

The RTEC virtual SCSI device driver component hierarchy of a given AIX configuration and the current settings for error checking level, disposition for low-severity errors, and disposition for medium-severity errors can be listed by the **errctrl** command. The **errctrl** command also allows you to modify the runtime error checking related configuration parameters. The following **errctrl** command output shows that the default error checking level for all net components is minimal (level=1), that low-severity errors are ignored (LowSevDis=48), and medium-severity errors are logged (collect service data and continue) (MedSevDisp=64):

```
hhaix6:root:/root # errctrl -c vscsi_initdd -q -r
---------------------------------------------+-------+-------+-------+--------
                                             | Have  |ErrChk |LowSev |MedSev
    Component name                           | alias | /level| Disp  | Disp
---------------------------------------------+-------+-------+-------+--------
vscsi_initdd
     .vscsi0                                 |  YES  | ON /1 |  48   |  64
          .lun8100000000000000               |  NO   | ON /1 |  48   |  64
          .lun8300000000000000               |  NO   | ON /1 |  48   |  64
```

The vscsi_initdd component is also enabled for the component dump and live dump services of the AIX enterprise RAS infrastructure.

4.2.4 MPIO and RAS component framework integration

AIX V6.1 enhances the AIX Multiple Path I/O (MPIO) framework and the AIX default Path Control Module (PCM) to exploit the AIX enterprise RAS infrastructure with regards to component tracing and runtime error checking. Component tracing and runtime error checking are referred to as domains supported by the AIX enterprise RAS infrastructure services.

The following outlines the parent/child hierarchy for MPIO and PCM component registration and naming:

- In previous AIX releases, the disk head driver registers to the enterprise RAS framework as the parent and each device controlled by the driver registers as a child of the disk head driver. AIX V6.1 establishes a parent/child hierarchy among the device instance of the disk head driver and the AIX MPIO framework. The MPIO framework registers to the enterprise RAS infrastructure as a child of the device instance controlled by the device driver. For example, the component name scsidiskdd.hdisk1.mpio1 shows that hdisk1 is the child of scsidiskdd and mpio1 is the child of hdisk1. hdisk1 is the device controlled by the parent SCSI disk, and mpio1 is the MPIO framework for that hdisk1.

- No hierarchy is needed within a PCM; however, the PCM itself registers to the AIX enterprise RAS infrastructure as a child of the MPIO framework. This extends the previously mentioned example component name to scsidiskdd.hdisk1.mpio1.pcm1. Here pcm1 is the PCM for hdisk1.

The MPIO and PCM component hierarchy of any given AIX configuration and the current settings for the memory trace mode (private or light weight memory trace) and the user trace mode (system trace) can be listed by the **ctctrl** command:

```
hhaix6:root:/root # ctctrl -c scsidiskdd -q -r
---------------------------------------+-------+-------+-------+---------------
                                       | Have  |Mem Trc|Sys Trc| Buffer size
     Component name                    | alias | /level| /level| /Allocated
---------------------------------------+-------+-------+-------+---------------
scsidiskdd
     .cd0                              |  YES  |  ON/3 |  ON/3 |       6400/YES
     .hdisk0                           |  YES  |  ON/3 |  ON/3 |       6400/YES
          .mpio0                       |   NO  |  ON/3 |  ON/3 |       4096/YES
               .pcm0                   |   NO  |  ON/3 |  ON/3 |       6400/YES
     .hdisk1                           |  YES  |  ON/3 |  ON/3 |       6400/YES
          .mpio1                       |   NO  |  ON/3 |  ON/3 |       4096/YES
               .pcm1                   |   NO  |  ON/3 |  ON/3 |       6400/YES
```

The RTEC MPIO and PCM component hierarchy of a given AIX configuration and the current settings for error checking level, disposition for low-severity errors, and disposition for medium-severity errors can be listed by the **errctrl** command. The **errctrl** command also allows you to modify the runtime error checking related configuration parameters. The following **errctrl** command output shows that the default error checking level for all net components is minimal (level=1), that low-severity errors are ignored (LowSevDis=48), and medium-severity errors are logged (collect service data and continue) (MedSevDisp=64):

```
hhaix6:root:/root # errctrl -c scsidiskdd -q -r
---------------------------------------------+-------+-------+-------+--------
                                             | Have  |ErrChk |LowSev |MedSev
    Component name                           | alias | /level| Disp  | Disp
---------------------------------------------+-------+-------+-------+--------
scsidiskdd
     .cd0                                    | YES   | ON /1 |  48   |  64
     .hdisk0                                 | YES   | ON /1 |  48   |  64
          .mpio0                             | NO    | ON /1 |  48   |  64
               .pcm0                         | NO    | ON /1 |  48   |  64
     .hdisk1                                 | YES   | ON /1 |  48   |  64
          .mpio1                             | NO    | ON /1 |  48   |  64
               .pcm1                         | NO    | ON /1 |  48   |  64
```

4.2.5 InfiniBand device driver component trace and RTEC support

Beginning with AIX V6.1, several of the AIX InfiniBand device drivers exploit the AIX enterprise RAS infrastructure with regards to component (device driver) naming and registration to the runtime error checking and component trace domains. The following AIX InfiniBand device drivers listed by their respective base component names are enhanced to utilize the component framework for trace and runtime error checking:

if_ib IP over InfiniBand interface
gxib InfiniBand Host Channel Adapter (gxibdd)
icm InfiniBand Connection Manager (icmdd)
tsib 4X InfiniBand PCI-X/ PCI-E card (tsibdd)

The following AIX filesets are impacted by the component framework adoption:

devices.chrp.IBM.lhca.rte
InfiniBand Host Channel Adapter device driver and ODM predefinitions

devices.common.IBM.ib.rte
InfiniBand Connection Manager (ICM), the IP over InfiniBand (IPoIB) interface and InfiniBand kernel libraries, as well as ODM predefinitions

devices.pci.b315445a.rte
4X InfiniBand PCI-X adapter device driver and ODM predefinitions

The InfiniBand device driver component hierarchy of a given AIX configuration and the current settings for the memory trace mode (private or light weight memory trace) and the user trace mode (system trace) can be listed by the **ctctrl** command (The **ctctrl** command also allows you to modify the component trace related configuration parameters.):

```
# ctctrl -q -r -t network_ib
-----------------------------------------+-------+-------+-------+---------------
                                         | Have  |Mem Trc|Sys Trc| Buffer size
     Component name                      | alias | /level| /level| /Allocated
-----------------------------------------+-------+-------+-------+---------------
gxib                                     |   NO  |  ON/3 |  ON/3 |      1024/YES
     .gxib_spec                          |   NO  |  ON/3 |  ON/3 |     64000/YES
     .iba0                               |   NO  |  ON/3 |  ON/3 |     64000/YES
     .iba1                               |   NO  |  ON/3 |  ON/3 |     64000/YES
ibka                                     |   NO  |  ON/3 |  ON/3 |    128000/YES
icm                                      |   NO  |  ON/3 |  ON/3 |    128000/YES
if_ib                                    |   NO  |  ON/3 |  ON/3 |      1024/YES
     .ib0                                |  YES  |  ON/3 |  ON/3 |     64000/YES
     .ib1                                |  YES  |  ON/3 |  ON/3 |     64000/YES
```

```
     .ib2                                │  YES  │  ON/3 │  ON/3 │      64000/YES
     .ib3                                │  YES  │  ON/3 │  ON/3 │      64000/YES
tsib                                     │   NO  │  ON/3 │  ON/3 │       1024/YES
     .iba0                               │   NO  │  ON/3 │  ON/3 │      64000/YES
     .tsib_spec                          │   NO  │  ON/3 │  ON/3 │      64000/YES
```

The base components and the related child components of the previous example command output are defined as follows:

gxib InfiniBand host channel adapter parent component and global trace component

gxib.gxib_spec InfiniBand host channel adapter (code name Galaxy) specific library

gxib.iba# Individual InfiniBand adapter driver instances (The # character denotes a place holder for the device instance number.)

ibka InfiniBand connection manager kernel library

icm InfiniBand connection manager

if_ib IPoIB parent component and global traces

if_ib.ib# IPoIB Interface instances (The # character denotes a place holder for the device instance number.)

tsib InfiniBand Cisco PCI/PCI-E parent and global trace component

tsib.iba# InfiniBand adapter's driver instance (The # character denotes a place holder for the device instance number.)

tsib.tsib_spec InfiniBand Cisco specific library

The RTEC InfiniBand device driver component hierarchy of a given AIX configuration and the current settings for error checking level, disposition for low-severity errors, and disposition for medium-severity errors can be listed by the **errctrl** command (The **errctrl** command also allows you to modify the runtime error checking related configuration parameters.):

```
# errctrl -c if_ib -q -r
# errctrl -c gxib -q -r
# errctrl -c icm -q -r
# errctrl -c tsib -q -r
```

4.2.6 LAN device driver component trace and RTEC support

Beginning with AIX V6.1, several of the AIX local area network (LAN) device drivers exploit the AIX enterprise RAS infrastructure with regards to component (device driver) naming and registration to the runtime error checking and component trace domains. The following AIX LAN device drivers listed by their respective base component names are enhanced to utilize the component framework:

vioentdd Virtual Ethernet device driver for LPAR clients and VIOS

goentdd Device driver for the 1-Port, 2-Port, and 4-Port Gigabit Ethernet PCI-X/PCIe adapter family

headd Device driver for the HEA 1/10 Gb Ethernet GX bus-attached integrated device

kngentdd Device driver for the 10 Gigabit Ethernet PCI-X DDR adapter

The following AIX filesets are impacted by the component framework adoption:

devices.pci.14106902.rte
1-Port, 2-Port, and 4-Port Gigabit Ethernet PCI-X/PCIe adapter family device driver

devices.pci.1410ec02.rte
LR/SR version of the 10 Gigabit Ethernet PCI-X DDR adapter

devices.chrp.IBM.lhea
HEA 1/10 Gb Ethernet device driver

devices.vdevice.IBM.l-lan.rte
Virtual Ethernet device driver

For the virtual Ethernet device driver component framework adoption, the base component name *vioentdd* designates a global head or anchor node with no associated resources. The vioentdd component is merely a marker that identifies the device driver to which the relevant sub-components are related. The vioentdd anchor node hat two child nodes, *dd* and *ffdc*. The vioentdd.dd sub-component designates the global component for the device driver and is used to log generic non-device-specific data. The vioentdd.ffdc sub-component records all global errors for First Failure Data Capture purposes. Under the dd component, the individual devices register their own component labeled by the respective *device logical names* (for example, ent0 for an Ethernet adapter). Finally, each device component has a dedicated *ffdc*, *managers*, *other*, *receive*, and *transmit* sub-component. The device specific ffdc component only logs errors that are usually hard errors that are non-recoverable. The managers component is dedicated to log data about the memory managers used for the vioentdd device driver. The other component captures generic errors and information that is device-specific. As suggested by their names, the receive and transmit components are utilized to capture data and trace errors related to receive and transmit operations respectively. Note that the ffdc, managers, other, receive, and transmit sub-components are only created when a device is opened, that is, has a TCP/IP interface assigned and is in the available state.

In respect to component trace, the **ctctrl** command can be used to list and control all LAN base components and their individually registered sub-components, for example:

```
hhaix6:root:/root # ctctrl -c vioentdd -q -r
---------------------------------------+-------+-------+-------+---------------
                                       | Have  |Mem Trc|Sys Trc| Buffer size
     Component name                    | alias | /level| /level| /Allocated
---------------------------------------+-------+-------+-------+---------------
vioentdd
     .dd                               |  NO   | ON/3  | ON/3  |     524288/YES
          .ent0                        |  YES  | ON/3  | ON/3  |      32768/YES
               .ffdc                   |  NO   | ON/3  | ON/3  |      65536/YES
               .managers               |  NO   | ON/3  | ON/3  |      65536/YES
               .other                  |  NO   | ON/3  | ON/3  |      32768/YES
               .receive                |  NO   | ON/3  | ON/3  |      32768/YES
               .transmit               |  NO   | ON/3  | ON/3  |      32768/YES
     .ffdc                             |  NO   | ON/3  | ON/3  |    2097152/YES
```

The previous listing shows the sub-components for a virtual Ethernet adapter with the device logical name ent0. For your convenience, you can refer to the component vioentdd.dd.ent0 directly by the alias ent0. As you can see, each virtual Ethernet device has its own First Failure Data Capture (ffdc), managers, other, packet receive (receive), and packet transmit (transmit) component.

The RTEC virtual Ethernet device driver component hierarchy of a given AIX configuration and the current settings for error checking level, disposition for low-severity errors, and disposition for medium-severity errors can be listed by the **errctrl** command. The **errctrl** command also allows you to modify the runtime error checking related configuration parameters.

The following **errctrl** command example output shows that the default error checking level for all vioentdd components is normal (level=3), that low-severity errors are ignored (LowSevDis=48), and medium-severity errors are logged (collect service data and continue) (MedSevDisp=64):

```
hhaix6:root:/root # errctrl -c vioentdd -q -r
---------------------------------------------+-------+-------+-------+--------
                                             | Have  |ErrChk |LowSev |MedSev
    Component name                           | alias | /level| Disp  | Disp
---------------------------------------------+-------+-------+-------+--------
vioentdd
     .dd                                     |  NO   | ON /3 |  48   |  64
          .ent0                              |  YES  | ON /3 |  48   |  64
```

The component hierarchy for Ethernet device drivers and device instances that are not related to the VIOS or the virtual Ethernet device drivers for LPAR clients differ from the vioentdd virtual Ethernet device driver component hierarchy as follows.

For the 1 and 10 Gb Ethernet adapters, and the Host Ethernet Adapters (HEA, also known as Integrated Virtual Ethernet adapter (IVE)), one global base component for each device driver is registered with the AIX component framework. The component names are defined by the AIX names of the respective device driver, *goentdd*, *headd*, or *kngentdd*. Each individual component is used to capture all traces related to configuration time before the actual device is configured. From there, each device is a child of the global component and has its own component labeled by the *device logical name* (for example, ent0 for an Ethernet adapter). This device component is used to capture all traces and runtime data that is not related to packet transmit or packet receive operations. Examples would be general errors, important data trace points, and I/O control (ioctl) paths. The packet transmit and packet receive related data and trace errors are recorded by two additional device sub-components, TX and RX respectively. The TX and RX components are only allocated and used for each individual device when the related adapter has been opened and is able to transmit and receive data. An Ethernet adapter is defined to be open when a TCP/IP interface has been assigned and the adapter is in the available state.

The following **ctctrl** and **errctrl** command example outputs show the component hierarchy and the component trace and runtime error detection related configuration parameters for two different Gigabit Ethernet PCI-X/PCIe adapter configurations.

Note that the **ctctrl** output lists multiple devices, of which only the ent1 adapter has been opened to receive or transmit data:

```
# ctctrl -c goentdd  -q -r
---------------------------------------+-------+-------+-------+---------------
                                       | Have  |Mem Trc|Sys Trc| Buffer size
     Component name                    | alias | /level| /level| /Allocated
---------------------------------------+-------+-------+-------+---------------
goentdd                                |  NO   | OFF/3 |  ON/3 |         0/ NO
     .ent0                             |  YES  |  ON/3 |  ON/3 |     131072/YES
     .ent1                             |  YES  |  ON/3 |  ON/3 |     131072/YES
          .RX                          |  NO   |  ON/3 |  ON/3 |     131072/YES
          .TX                          |  NO   |  ON/3 |  ON/3 |     131072/YES
     .ent3                             |  YES  |  ON/3 |  ON/3 |     131072/YES
     .ent4                             |  YES  |  ON/3 |  ON/3 |     131072/YES
     .ent5                             |  YES  |  ON/3 |  ON/3 |     131072/YES

# errctrl -c goentdd  -q -r
---------------------------------------------+-------+-------+-------+--------
                                             | Have  |ErrChk |LowSev |MedSev
    Component name                           | alias | /level| Disp  | Disp
---------------------------------------------+-------+-------+-------+--------
goentdd                                      |  NO   | ON /7 |  48   |  64
     .ent0                                   |  YES  | ON /7 |  48   |  64
     .ent1                                   |  YES  | ON /7 |  48   |  64
     .ent3                                   |  YES  | ON /7 |  48   |  64
     .ent4                                   |  YES  | ON /7 |  48   |  64
     .ent5                                   |  YES  | ON /7 |  48   |  64
```

The following **ctctrl** and **errctrl** command example outputs show the component hierarchy and the component trace and runtime error detection related configuration parameters for a given Host Ethernet Adapter configuration:

```
# ctctrl -c headd -q -r
---------------------------------------+-------+-------+-------+---------------
                                       | Have  |Mem Trc|Sys Trc| Buffer size
     Component name                    | alias | /level| /level| /Allocated
---------------------------------------+-------+-------+-------+---------------
headd                                  |  NO   |  ON/3 |  ON/3 |     131072/YES
     .ent1                             |  YES  |  ON/3 |  ON/3 |     131072/YES
          .RX                          |  NO   |  ON/3 |  ON/1 |     131072/YES
          .TX                          |  NO   |  ON/3 |  ON/1 |     131072/YES

# errctrl -c headd -q -r
---------------------------------------------+-------+-------+-------+--------
                                             | Have  |ErrChk |LowSev |MedSev
    Component name                           | alias | /level| Disp  | Disp
---------------------------------------------+-------+-------+-------+--------
headd
```

```
     .ent1                                  |  YES  | ON /3 |  48   |  64
```

The following **ctctrl** and **errctrl** command example outputs show the component hierarchy and the component trace and runtime error detection related configuration parameters for a given 10 Gigabit Ethernet PCI-X DDR adapter configuration:

```
# ctctrl -c kngentdd -q -r
---------------------------------------+-------+-------+-------+---------------
                                       | Have  |Mem Trc|Sys Trc| Buffer size
     Component name                    | alias | /level| /level| /Allocated
---------------------------------------+-------+-------+-------+---------------
kngentdd                               |   NO  | OFF/3 |  ON/3 |          0/ NO
     .ent0                             |  YES  |  ON/3 |  ON/3 |     131072/YES
          .RX                          |   NO  |  ON/3 |  ON/3 |     131072/YES
          .TX                          |   NO  |  ON/3 |  ON/3 |     131072/YES

# errctrl -c kngentdd -q -r
---------------------------------------------+-------+-------+-------+--------
                                             | Have  |ErrChk |LowSev |MedSev
    Component name                           | alias | /level| Disp  | Disp
---------------------------------------------+-------+-------+-------+--------
kngentdd                                     |  NO   | ON /3 |  48   |  64
     .ent0                                   |  YES  | ON /3 |  48   |  64
```

4.2.7 Error level checking for TCP kernel and kernel extension

The AIX operating system will be forced to halt when the TCP kernel and kernel extension code encounters unexpected paths or unrecoverable errors (debug asserts). However, at certain places in the TCP kernel code of previous AIX releases, it would have been not required to force a system halt, as the return error could have been handled by the running process itself. In AIX V6.1, these conditions and asserts were identified and are either replaced with component traces along with proper return code or are moved under an appropriate runtime error level. AIX V6.1 collects the required debug information using component trace and, with the relevant return code, errors can be handled effectively by the calling process.The trace can be saved in component memory, system memory, or both. The size of the memory buffer can be changed dynamically.

In summary, AIX V6.1 enhances the exploitation of the AIX enterprise RAS infrastructure by the TCP kernel and netinet kernel extension in two ways:

1. Provides runtime error checking information instead of a system halt at the default level.
2. Provides additional component trace framework integration.

The following routines were modified to improve the failure robustness of the TCP kernel and kernel extension code: m_copym(),m_copydata(), m_copymext(), uipc_usrreq(), udp_usrreq(), rip_usrreq(), if_attach(), in_control, in6_control(), and netintr(). Also, a new sub-component *netisr* was implemented under the parent component net for the component trace and the runtime error checking domain. And finally, component traces are added to all network software interrupt routines (netisr) in AIX V6.1.

The net component hierarchy of any given AIX configuration and the current settings for the memory trace mode (private or light weight memory trace) and the user trace mode (system trace) can be listed by the **ctctrl** command. The **ctctrl** command also allows you to modify the component trace related configuration parameters:

```
hhaix6:root:/root # ctctrl -c net -q -r
---------------------------------------+-------+-------+-------+---------------
                                       | Have  |Mem Trc|Sys Trc| Buffer size
     Component name                    | alias | /level| /level| /Allocated
---------------------------------------+-------+-------+-------+---------------
net                                    |   NO  |  ON/1 |  ON/7 |       1024/YES
     .cdli                             |   NO  |  ON/1 |  ON/7 |      10240/YES
     .loop                             |   NO  |  ON/1 |  ON/7 |      10240/YES
     .netisr                           |   NO  |  ON/1 |  ON/7 |      10240/YES
     .route                            |   NO  |  ON/1 |  ON/7 |      40960/YES
```

The RTEC net component hierarchy of a given AIX configuration, current settings for error checking level, and disposition for low-severity errors can be listed by the **errctrl** command. The **errctrl** command also allows you to modify the runtime error checking related configuration parameters. The following **errctrl** command output shows that the default error checking level for all net components is minimal (level=1), that low-severity errors are ignored (LowSevDis=48), and medium-severity errors are logged (collect service data and continue) (MedSevDisp=64):

```
hhaix6:root:/root # errctrl -c net -q -r
---------------------------------------------+-------+-------+-------+--------
                                             | Have  |ErrChk |LowSev |MedSev
    Component name                           | alias | /level| Disp  | Disp
---------------------------------------------+-------+-------+-------+--------
net                                          |  NO   | ON /1 |  48   |  64
     .cdli                                   |  NO   | ON /1 |  48   |  64
     .loop                                   |  NO   | ON /1 |  48   |  64
     .netisr                                 |  NO   | ON /1 |  48   |  64
     .route                                  |  NO   | ON /1 |  48   |  64
```

4.2.8 IPsec component trace exploitation

What is generally thought of as the Internet Protocol Security (IPsec) subsystem is actually a collection of several kernel extensions. To enable component trace exploitation for IP security, AIX V6.1 introduces an IPsec base component, named *ipsec*, and one sub-component for each of the *capsulate*, *crypto*, *filter*, and *tunnel* IPsec kernel extensions. The IPsec component hierarchy and the current settings for the memory trace mode (private or light weight memory trace) and the user trace mode (system trace) can be listed by the **ctctrl** command:

```
hhaix6:root:/root # ctctrl -c ipsec -q -r
---------------------------------------+-------+-------+-------+---------------
                                       | Have  |Mem Trc|Sys Trc| Buffer size
     Component name                    | alias | /level| /level| /Allocated
---------------------------------------+-------+-------+-------+---------------
ipsec                                  |   NO  |  ON/3 |  ON/3 |      40960/YES
     .capsulate                        |   NO  |  ON/3 |  ON/3 |      10240/YES
     .crypto                           |   NO  |  ON/3 |  ON/3 |      10240/YES
     .filter                           |   NO  |  ON/3 |  ON/3 |      10240/YES
     .tunnel                           |   NO  |  ON/3 |  ON/3 |      10240/YES
```

The trace buffers are structured so that all IPsec components dump trace into a single large buffer that goes to system trace. Smaller buffers will be used for component specific memory mode tracing. This tracing will be turned on by default. The following describes the different IPsec trace buffers being created:

Parent ipsec buffer This buffer of 40 KB size is used as a global IPsec trace buffer to collect almost all of the component trace information.

Capsulate buffer This private trace buffer of 10 KB buffer size is used only for capsulate kernel extension memory mode tracing.

Crypto buffer This private trace buffer of 10 KB size is used only for the crypto kernel extension memory mode tracing.

Filter buffer This private trace buffer of 10 KB size is used only for the filter kernel extension memory mode tracing.

Tunnel buffer This private trace buffer of 10 KB size is used only for the tunnel kernel extension memory mode tracing.

AIX V6.1 defines IPsec trace hook IDs in the ipsp_trchk.h header file and their corresponding trace formatting in /etc/trcfmt.

4.2.9 PCI device driver component trace adoption

In AIX V6.1, several new RAS features were implemented within the PCI bus device driver. These features include component trace, use of storage keys, and new private heaps for memory allocated by the driver to improve data isolation. On the first call to the configuration and initialization kernel service, the PCI bus driver registers a *pci* base component and a *pci.eeh* sub-component with the component framework of AIX. Each call to the configuration and initialization routine also results in the registration of a pci.pci# sub-component, where # designates a place holder for integer values. On each call to the kernel service, which allocates and initializes resources for performing Direct Memory Access (DMA) with PCI devices (d_map_init), an additional sub-component is registered in the form of pci.pci#.handle#, where # designates a place holder for configuration dependent integer values. This implementation allows drivers with multiple handles to have separate component trace buffers to trace the DMA activities of each handle separately.

The PCI component hierarchy can be represented as follows:

```
pci
    .pci0
    .pci1
        .handle1
        .handle2
    .pci2
        .handle1
        .handle2
        .handle3
        .handle4
    .pci3
        .handle1
...
    .eeh
```

Also, the alias *eeh* has been created to refer to the sub-component pci.eeh for convenience.

The PCI component hierarchy of any given AIX configuration and the current settings for the memory trace mode (private or light weight memory trace) and the user trace mode (system trace) can be listed by the **ctctrl** command. The **ctctrl** command also allows you to modify the component trace related configuration parameters:

```
# ctctrl -c pci -q -r
---------------------------------------+-------+-------+-------+---------------
                                       | Have  |Mem Trc|Sys Trc| Buffer size
     Component name                    | alias | /level| /level| /Allocated
---------------------------------------+-------+-------+-------+---------------
pci
     .eeh                              |  YES  |  ON/3 |  ON/3 |       2048/YES
     pci2
          .handle1                     |   NO  |  ON/3 |  ON/3 |        512/YES
          .handle2                     |   NO  |  ON/3 |  ON/3 |        512/YES
     pci3
          .handle1                     |   NO  |  ON/3 |  ON/3 |        512/YES
          .handle2                     |   NO  |  ON/3 |  ON/3 |        512/YES
```

4.2.10 Virtual bus device driver component trace adoption

In AIX V6.1, several new RAS features were implemented within the virtual bus device driver. These features include component trace, use of storage keys and new private heaps for memory allocated by the driver to improve data isolation.

On the first call to the configuration and initialization kernel service, the virtual bus driver registers with the AIX RAS component framework under the base component name *vdev*. Each call to the configuration and initialization routine also results in the registration of a vdev.vio# sub-component, where # designates a place holder for integer values. On each call to the kernel service, which allocates and initializes resources for performing Direct Memory Access (DMA) with virtual bus devices, an additional sub-component is registered in the form of vdev.vio#.handle#, where # designates a place holder for configuration dependent integer values. This implementation allows drivers with multiple handles to have separate component trace buffers to trace the DMA activities of each handle separately.

The virtual bus device driver component hierarchy of any given AIX configuration and the current settings for the memory trace mode (private or light weight memory trace) and the user trace mode (system trace) can be listed by the **ctctrl** command:

```
hhaix6:root:/root # ctctrl -c vdev -q -r
---------------------------------------+-------+-------+-------+---------------
                                       | Have  |Mem Trc|Sys Trc| Buffer size
     Component name                    | alias | /level| /level| /Allocated
---------------------------------------+-------+-------+-------+---------------
vdev
     vio0
          .handle1                     |   NO  |  ON/3 |  ON/3 |        512/YES
          .handle10                    |   NO  |  ON/3 |  ON/3 |        512/YES
          .handle2                     |   NO  |  ON/3 |  ON/3 |        512/YES
          .handle3                     |   NO  |  ON/3 |  ON/3 |        512/YES
          .handle4                     |   NO  |  ON/3 |  ON/3 |        512/YES
          .handle5                     |   NO  |  ON/3 |  ON/3 |        512/YES
          .handle6                     |   NO  |  ON/3 |  ON/3 |        512/YES
          .handle7                     |   NO  |  ON/3 |  ON/3 |        512/YES
          .handle8                     |   NO  |  ON/3 |  ON/3 |        512/YES
          .handle9                     |   NO  |  ON/3 |  ON/3 |        512/YES
```

4.2.11 Component trace for USB system driver

The USB system driver is enhanced in AIX V6.1 to exploit the AIX enterprise RAS component trace framework. The base component name for the USB system driver is *usb_system* and one single component specific node with the sub-component name of *usb0* will be defined during the driver configuration and initialization process. Note that no matter how many USB host controllers or USB devices are attached to the system, there is only one USB system driver instance in the customized devices CuDv ODM database that always has the name usb0. USB system driver component tracing will utilize private buffer memory trace mode and user trace mode. The USB system driver trace hook ID is 0x738.

The usb_system parent node is not component framework domain aware; that is, it will simply be a place holder. The usb0 node is component trace aware, but is not enabled for the runtime error checking or component dump domain of the AIX enterprise RAS component framework. The following customizations are performed for the usb0 node during driver configuration:

1. A 8192 byte private trace buffer will be allocated.
2. Memory trace mode will be enabled.
3. An alias of usb0 will be created for the usb_system.usb0 component.

The USB system driver component hierarchy for a given configuration and the current settings for the memory trace mode (private memory trace) and the user trace mode (system trace) can be listed by the **ctctrl** command:

```
# ctctrl -c usb_system -q -r
---------------------------------------+-------+-------+-------+---------------
                                       | Have  |Mem Trc|Sys Trc| Buffer size
     Component name                    | alias | /level| /level| /Allocated
---------------------------------------+-------+-------+-------+---------------
usb_system
     .usb0                             |  YES  |  ON/3 |  ON/3 |      8192/YES
```

The **ctctrl** command also allows you to modify the component trace related configuration parameters.

4.2.12 Component trace for USB audio

The USB audio device driver will use component trace in a way that allows traces to be associated with either the entire parent driver or with any of the existing sub-components. The entire parent driver will be identified by the registered component name *usb_audio*. The USB audio driver trace hook ID is 0x61E. The sub-components are selected to be the devices as listed in the ODM that have device special files in /dev and are identified by their logical device name. Code that is related to the USB audio device driver but that is not associated with a specific sub-component falls under the parent driver usb_audio base component. The **ctctrl** command can be used to list and control the USB audio driver parent component and all sub-components registered under the usb_audio base component name.

The following listing represents one USB audio device composed of three USB interfaces. Each of these three USB interfaces have a device in ODM and /dev.

When the USB audio driver is loaded and called to configure the first device, it configures the parent device driver base component first, and then the sub-component devices:

```
# ctctrl -c usb_audio -q -r
---------------------------------------+-------+-------+-------+---------------
                                       | Have  |Mem Trc|Sys Trc| Buffer size
Component name                         | alias | /level| /level| /Allocated
---------------------------------------+-------+-------+-------+---------------
usb_audio                              |   NO  |  ON/1 |  ON/3 | 4096/YES
     .paud0                            |   NO  |  ON/1 |  ON/3 | 4096/YES
     .paudas0                          |   NO  |  ON/1 |  ON/3 | 4096/YES
     .paudas1                          |   NO  |  ON/1 |  ON/3 | 4096/YES
```

The list of the audio devices can be displayed by the **lsdev** command as follows:

```
# lsdev -C -c audio
paud0 Available 0.2.1 USB Audio Device, AudioControl Interface
paudas0 Available 0.2.1 USB Audio Device, AudioStreaming Interface
paudas1 Available 0.2.1 USB Audio Device, AudioStreaming Interface
```

4.2.13 Component trace for 2D graphics device drivers

Beginning with AIX V6.1, the 2D graphics device drivers for the GXT130P and GXT145 graphics adapters are instrumented to leverage the component trace services. The integration into the component trace framework enables both the memory trace mode (private or light weight memory trace) and the user trace mode (system trace) for the named graphics device drivers. This enhancements provides advanced First Failure Data Capture (FFDC) and Second Failure Data Capture (SFDC) capabilities and will potentially supersede the need for custom-built debug drivers. The hierarchy within 2D graphics device drivers defines the device driver itself as the base component and each adapter controlled by that device driver is implemented as a sub-component identified by the logical device name. The following base component names are introduced:

lanaidd The GXT130P graphics adapter 2D device driver delivered by the devices.pci.2b102005.rte fileset.

cortinadd The GXT145 graphics adapter 2D device driver delivered by the devices.pci.2b102725.rte fileset.

The base component names are references to the development code names Lanai (sixth-largest of the Hawaiian Islands) and Cortina (Cortina d'Ampezzo, a town in northern Italy). Each adapter 's trace event will use the same trace hook ID as in previous system trace debug driver versions:

- Lanai 's hook ID = 0x737
- Cortina 's hook ID = 0x73C

The 2D graphics device driver component hierarchy for a given configuration and the current settings for the memory trace mode (private or light weight memory trace) and the user trace mode (system trace) can be listed by the **ctctrl** command:

```
# ctctrl -c lanaidd -q -r
---------------------------------------+-------+-------+-------+---------------
                                       | Have  |Mem Trc|Sys Trc| Buffer size
    Component name                     | alias | /level| /level| /Allocated
---------------------------------------+-------+-------+-------+---------------
lanaidd
     .lai0                             |  YES  |  ON/3 |  ON/3 | 2560/YES

# ctctrl -c cortinadd -q -r
---------------------------------------+-------+-------+-------+---------------
                                       | Have  |Mem Trc|Sys Trc| Buffer size
    Component name                     | alias | /level| /level| /Allocated
---------------------------------------+-------+-------+-------+---------------
cortinadd
     .cor0                             |  YES  |  ON/3 |  ON/3 | 2560/YES
```

The **ctctrl** command also allows you to modify the component trace related configuration parameters.

4.2.14 System loader runtime error checking

The AIX system loader was enhanced in AIX V6.1 to begin taking advantage of the AIX RAS component framework for the runtime error checking (RTEC) domain of the AIX enterprise RAS infrastructure. The system loader RTEC adoption provides improved first failure data capture (FFDC) support, which allows defects to be found closer to their root cause in a production environment.

The AIX V6.1 system loader code has been segmented into RTEC-aware components, which are registered individually with the AIX enterprise RAS infrastructure. Tuning can then be done at a component level, such as modifying the error-checking level or dispositions for 64-bit programs without affecting the treatment of 32-bit programs.

The AIX V6.1 system loader manages multiple regions used for loading programs and shared objects. Each region corresponds to a RTEC-aware component. Table 4-2 on page 133 lists the components that are registered with the AIX V6.1 enterprise RAS component framework.

Table 4-2 System loader RAS components

Component	Alias	Description
ldr		Loader base component
ldr.kernext	kernext	Kernel extension region
ldr.lib32	lib32	32-bit libraries
ldr.lib64	lib64	64-bit libraries
ldr.lib32.xxx		32-bit libraries for WPARs or named shared library regions
ldr.lib64.xxx		64-bit libraries for WPARs or named shared library regions
ldr.process32		General 32-bit processes
ldr.process64		General 64-bit processes

The xxx characters in ldr.lib32.xxx and ldr.lib64.xxx are replaced with the region or WPAR name.

4.2.15 NFS and CacheFS runtime error checking

Beginning with AIX V6.1, the Network File System version 4 (NFSv4) and the Cache File System (CacheFS) implementation utilize the AIX RAS component framework for the runtime error checking (RTEC) domain of the AIX enterprise RAS infrastructure.

The NFSv4 extension of AIX V6.1 creates a hierarchical name space that allows runtime error checking to be tuned in a granular manner. The NFSv4 extension defines a generic base component (anchor node) named *nfs* and the initialization of NFSv4 results in a NFS version specific child of the nfs anchor node called *nfs4*. This part of the name space can be considered as essentially static. Client and server nodes are then created as children (or leaf nodes) of the NFSv4 anchor node at runtime. This will result in the NFS component adding the paths *nfs.nfs4.nfs4_client* and *nfs.nfs4.nfs4_server* to the AIX enterprise RAS component name space.

The RTEC NFSv4 component hierarchy of a given AIX configuration and the current settings for error checking level, disposition for low-severity errors, and disposition for medium-severity errors can be listed by the **errctrl** command. The **errctrl** command also allows you to modify the runtime error checking related configuration parameters.

The following **errctrl** command output shows that the default error checking level for all NFSv4 components is normal (level=3), that low-severity errors are ignored (LowSevDis=48), and medium-severity errors are logged (collect service data and continue) (MedSevDisp=64):

```
hhaix6:root:/root # errctrl -c nfs -q -r
---------------------------------------------+-------+-------+-------+--------
                                             | Have  |ErrChk |LowSev |MedSev
    Component name                           | alias | /level| Disp  | Disp
---------------------------------------------+-------+-------+-------+--------
nfs
     nfs4
          .nfs4_client                       |  NO   | ON /3 |  48   |  64
          .nfs4_server                       |  NO   | ON /3 |  48   |  64
```

The CacheFS kernel extension creates a hierarchical name space that allows runtime error checking to be tuned in a granular manner. The CacheFS kernel extension defines a base component named *cachefs* during initialization. This base component can be considered as essentially static. CacheFS creates a child of the anchor node for each CacheFS file system at runtime. The sub-component name consists of the NFSv4 mount point and an appended file system specific unique ID. Special characters of the mount point name are converted to underscores. Also, the JFS2 layer instantiates for each CacheFS directory created in the local file system one *jfs2.filesystem._cachfs_32.metadata* and one *jfs2.filesystem._cachfs_32.user.data* RTEC component.

The RTEC CacheFS component hierarchy of a given AIX configuration and the current settings for error checking level, disposition for low-severity errors, and disposition for medium-severity errors can be listed by the **errctrl** command. The **errctrl** command also allows you to modify the runtime error checking related configuration parameters. The following **errctrl** command output shows that the default error checking level for all CacheFS components is normal (level=3), that low-severity errors are ignored (LowSevDis=48), and medium-severity errors are logged (collect service data and continue) (MedSevDisp=64). The sub-component name _usr_sys_inst_images_ITUAMv6_1_22 refers to the NFSv4 local mount point /usr/sys/inst.images/ITUAMv6.1:

```
hhaix6:root:/root # errctrl -c cachefs -q -r
---------------------------------------------+-------+-------+-------+--------
                                             | Have  |ErrChk |LowSev |MedSev
    Component name                           | alias | /level| Disp  | Disp
---------------------------------------------+-------+-------+-------+--------
cachefs                                      |  NO   | ON /3 |  48   |  64
     ._usr_sys_inst_images_ITUAMv6_1_22      |  NO   | ON /3 |  48   |  64
```

4.2.16 Runtime error checking for watchdog timer

The watchdog timer kernel services are typically utilized to verify that an I/O operation completes in a reasonable time. The watchdog timer services can be used for noncritical times, having a one-second resolution. Because a watchdog timer has less granularity than a timer created with the talloc() kernel service, it exhibits a much lower path-length footprint when starting the timer. However, this highly efficient service is achieved at the expense of structural robustness.

AIX V6.1 enhances the watchdog timer kernel services implementation by adding self-checking functionality to support First Failure Data Capture (FFDC) capabilities and to improve the overall serviceability characteristic.

During the AIX kernel initialization, a specific kernel service is called to set up timer services for the master processor (processor ID 0 in most cases). This kernel service in turn calls the initialization kernel service for the watchdog timer. At the beginning of the later initialization process, AIX V6.1 registers the watchdog timer component with the AIX enterprise RAS component framework under the name *watchdog*. The watchdog timer component is implemented as a child of the *proc* processor component in the RAS component hierarchy.

The RTEC watchdog timer component of a given AIX configuration and the current settings for error checking level, disposition for low-severity errors, and disposition for medium-severity errors can be listed by the **errctrl** command. The **errctrl** command also allows you to modify the runtime error checking related configuration parameters. The following **errctrl** command output shows that the default error checking level for the watchdog timer component is normal (level=3), that low-severity errors are ignored (LowSevDis=48), and medium-severity errors are logged (collect service data and continue) (MedSevDisp=64):

```
hhaix6:root:/usr/include/sys # errctrl -c proc -q -r
---------------------------------------------+-------+-------+-------+--------
                                             | Have  |ErrChk |LowSev |MedSev
    Component name                           | alias | /level| Disp  | Disp
---------------------------------------------+-------+-------+-------+--------
proc
     .disa                                   |  NO   | ON /3 |  48   |  64
     .lock                                   |  NO   | ON /3 |  48   |  64
     .watchdog                               |  NO   | ON /3 |  48   |  64
```

4.2.17 System memory allocator adoption of run-time error checking

In AIX 5L V5.3 TL5, the system memory allocator xmalloc runtime error checking (RTEC) function was integrated into the RAS component hierarchy, and appears as the *alloc.xmdbg* and the *alloc.heap0* components. This allows their runtime error checking properties to be adjusted by the **errctrl** command.

The alloc.xmdbg component provides RTEC capabilities for memory allocation in the kernel heap, pinned heap, and all heaps created by the kernel subsystem through the heap_create subroutine. The alloc.heap0 component applies to the loader specific heap that appears in the kernel segment.

In AIX V6.1, to provide better first failure data capture (FFDC) characteristics, more runtime error checks occur by default in the product code than in previous AIX versions. There is a natural conflict between checking and performance, and that conflict is minimized by sampling whether xmalloc (and xmfree by extension) should employ various checking techniques on a given call. The sampling frequencies can be tuned individually or changed by raising the error checking level. As the checking level goes up, the performance impact is greater.

> **Note:** All default values for sampling frequencies may be subject to change without notice.

High-level controls for xmalloc RTEC

By default, xmalloc RTEC is enabled in AIX V6.1 and the characteristics for this component can be controlled at different levels. At one level, xmalloc RTEC can be disabled (or re-enabled) along with all other AIX runtime error checking. System administrators may use the **smitty ffdc** interface for the **/usr/lib/ras/ffdcctrl** command, or apply the appropriate **errctrl** commands. The **errctrl errcheckoff**, and **errctrl errcheckon** commands affect all of AIX error checking. Error checking characteristics can also be changed for the xmalloc subsystem with component specific tuning parameters. In AIX V6.1, a reboot is never required to change a checking level. All options can be configured at runtime using the **errctrl** command.

The following command is available to turn off error checking for the system memory allocator:

```
# errctrl -c alloc.xmdbg errcheckoff
```

AIX V6.1 offers the additional, optional flag -P to make this setting persistent across reboots.

Use the following command to turn on error checking for xmalloc. The command enables xmalloc RTEC at previously set checking levels or at default levels:

```
# errctrl -c alloc.xmdbg errcheckon
```

Note that the default checking level in AIX V6.1 is ERRCHECK_NORMAL (3), while the checking level in AIX 5L V5.3 was configured to be ERRCHECK_MINIMAL (1).

The alloc.xmdbg and alloc.heap0 components and their potential child components support a variety of tuning parameters that can be changed as a group. This is done with the **errctrl** command using the **errcheckminimal**, **errchecknormal**, **errcheckdetail**, and **errchecklevel=9** sub-commands.

To set alloc.xmdbg RTEC to the minimal error checking level, system administrators need to run the following command:

```
# errctrl -c alloc.xmdbg errcheckminimal
```

When the error-checking level is set to minimal (level 1), the checks and techniques used by xmalloc are applied at fairly low frequencies. These frequencies can be examined with the **kdb xm -Q** command.

This can be done from the command line by piping xm -Q to the **kdb** command:

```
# echo xm -Q | kdb
```

Minimal checking is the default checking level in AIX 5L V5.3. The frequency that appears next to each tuning parameter is proportional to the frequency base. In the following example, the ruin all data technique will be applied five times out of every 1024 (0x400) calls to xmalloc (about 0.5% of the time). 16 byte allocations will be promoted about 10 times out of every 1024 calls to xmalloc (about 1% of the time). The various checks and techniques will be described in more detail later:

```
KDB(1)> xm -Q
XMDBG data structure @ 0000000002521360
Debug State:   Enabled
Frequency Base  00000400
Tunable                         Frequency
Allocation Record               00000033
Ruin All Data                   00000005
Trailer non-fragments           00000005
Trailer in fragments            00000005
Redzone Page                    00000005
VMM Check                       0000000A
Deferred Free Settings
  Fragments                     00000005
```

```
  Non-fragments                 00000005
  Promotions                    00000066

Page Promotion
        Frag size       Frequency
        [00010]         0000000A
        [00020]         0000000A
        [00040]         0000000A

... ommitted lines ...
```

In AIX V6.1, the levels and tuning parameters are slightly different in comparison to AIX 5L V5.3. The **kdb** output has changed, because the frequency base is 65536 in AIX 5L V5.3 but 1024 in AIX V6.1, and because the formatting has been enhanced. These frequencies are always subject to change, but can be examined on a live machine.

To set alloc.xmdbg RTEC to the normal error checking level, system administrators need to run the following command:

```
# errctrl -c alloc.xmdbg errchecknormal
```

When the error-checking level is set to normal (level 3), the checks and techniques are applied at higher frequencies than minimal checking provides. Normal error checking is the default level setting in AIX V6.1. In the following example, a trailer will be added to a fragment about 51 (0x33) times out of every 1024 times a fragment is allocated (about 5%). The deferred free technique will be applied to page promotions about 153 (0x99) times out of every 1024 (0x400) times a fragment is promoted (about 15% of the time). These techniques will be discussed in more detail later. These frequencies are subject to change, but can always be examined on a live machine. In AIX V6.1, the levels and tuning parameters are slightly different in comparison to AIX 5L V5.3:

```
KDB(0)> xm -Q
XMDBG data structure @ 00000000025426F0
Debug State:   Enabled
Frequency Base:    00000400
Tunable                    Frequency
Allocation Record          00000099
Ruin All Data              00000033
Trailer non-fragments      0000000A
Trailer in fragments       00000033
Redzone Page               0000000A
VMM Check                  0000000A
Deferred Free Settings
  Fragments                0000000A
```

```
  Non-fragments              0000000A
  Promotions                 00000099

Page Promotion
       Frag size          Frequency
        [00010]           0000000D
        [00020]           0000000D
        [00040]           0000000D

... ommitted lines ...
```

To set the alloc.xmdbg RTEC to detail error checking level, system administrators need to run the following command:

```
# errctrl -c alloc.xmdbg errcheckdetail
```

When the error-checking level is set to detail (level 7), the checks and techniques are applied at fairly high frequencies. This gives a high checking level with a goal of not impacting system performance too greatly. In the example below, allocation records are kept on every call to xmalloc (0x400 out of 0x400 calls). 0x80 byte fragments are promoted 0x200 out of every 0x400 times the 0x80 byte fragment is allocated (50%):

```
KDB(0)> xm -Q
XMDBG data structure @ 00000000025426F0
Debug State:   Enabled
Frequency Base:    00000400
Tunable                      Frequency
Allocation Record            00000400
Ruin All Data                00000200
Trailer non-fragments        00000066
Trailer in fragments         00000200
Redzone Page                 00000266
VMM Check                    00000266
Deferred Free Settings
  Fragments                  00000066
  Non-fragments              00000066
  Promotions                 00000200

Page Promotion
       Frag size          Frequency
        [00010]           00000200
        [00020]           00000200
        [00040]           00000200
        [00080]           00000200
```

```
... ommitted lines ...
```

These AIX V6.1 levels and tuning parameters are much different in comparison to the previous AIX release. In AIX V5.3, **errcheckdetail** is more severe and is the same as maximal level (9) in AIX V6.1, as shown in the next paragraph. The **kdb** output format has been enhanced for AIX V6.1.

To set the alloc.xmdbg RTEC to the maximum error checking level, system administrators need to run the following command:

```
# errctrl -c alloc.xmdbg errchecklevel=9
```

At this checking level, all tuning parameters are set to the maximum levels. Performance is most affected at this checking level. All the frequencies should match the frequency base, meaning all the checks are always done:

```
KDB(0)> xm -Q
XMDBG data structure @ 00000000025426F0
Debug State:   Enabled
Frequency Base:    00000400
Tunable                          Frequency
Allocation Record                00000400
Ruin All Data                    00000400
Trailer non-fragments            00000400
Trailer in fragments             00000400
Redzone Page                     00000400
VMM Check                        00000400
Deferred Free Settings
  Fragments                      00000400
  Non-fragments                  00000400
  Promotions                     00000400

Page Promotion
       Frag size              Frequency
        [00010]               00000400
        [00020]               00000400
        [00040]               00000400
        [00080]               00000400

... omitted lines ...
```

Low-level xmalloc debug tuning parameters

xmalloc RTEC features are activated for a given allocation based on probabilities. The **errctrl** command that controls the tuning parameters takes the probability of application (frequency) as an argument. In AIX V6.1, the system administrator can set the probability of a check being performed by specifying the frequency of the tuning parameter as a number between 0 and 1024. This is the number of times out of the base frequency (1024) the technique is to be applied by xmalloc. For example, to request 50%, the system administrator specifies a frequency of 512. Frequencies can be input as decimal or hexadecimal numbers, so 50% can be specified as 0x200. As a convenient alternative, the frequency can be expressed as a percentage. To do this, the system administrator specifies a number between 0 and 100 followed by the% sign. In AIX 5L V5.3, the base frequency is 65536, so to request 50%, the user specifies a frequency of 32768. Hexadecimal numbers are not accepted and the percentage frequency is not supported in AIX 5L V5.3.

Note: The base frequency and the default frequencies for any xmalloc debug tuning parameter may be subject to change without notice.

Tuning parameters affected by RTEC level

By default, the value of all the xmalloc related tuning parameters is set based on the error checking level, as described previously. Specific tuning parameters can be changed by using pass-through sub-commands. The following paragraphs detail the pass-through commands and the effects of each tuning parameter.

Keep an allocation record

```
# errctrl -c alloc.xmdbg alloc_record=<frequency>
```

This command sets the frequency of keeping a record for an allocation. Records are also kept if any other debug technique is applied, so the percentage of allocations with a record may be considerably larger than this number would otherwise indicate. The allocation record contains a three-level stack trace-back of the xmalloc and xmfree callers as well as some other debug information about the allocated memory. The presence of a record is a minimum requirement for RTEC.

Ruin storage

```
# errctrl -c alloc.xmdbg ruin_all=<frequency>
```

This options sets the frequency at which xmalloc will return storage that is filled with a ruin pattern. This helps catch errors with un-initialized storage, as a caller with bugs is more likely to crash when using the ruined storage. xmalloc does not perform any explicit checks when this technique is employed. The ruined data will contain 0x66 in every allocated byte on allocation, and 0x77 in every previously allocated byte after being freed.

Check for overwrites in small allocations

```
# errctrl -c alloc.xmdbg small_trailer=<frequency>
```

This is one of three options that affect the frequency of trailers. There are two options that deal with trailers and a third compatibility option. The small_trailer option is specific for allocations that are less than half a page. A trailer is a data pattern that is written immediately after the returned storage. Trailers can consume up to 128 bytes of storage. When storage is freed, xmfree will ensure consistency in the trailer bytes and log an error for any infractions, since inconsistencies represent overwrites.

This option is new in AIX V6.1. In AIX 5L V5.3, all trailers are controlled with a single tuning parameter (alloc_trailer). The error disposition can be made more severe by changing the disposition of medium severity errors as follows:

```
# errctrl -c alloc.xmdbg medsevdisposition=sysdump
```

Overwrites to the trailers and other medium severity errors will cause a system crash if the severity disposition is changed as above.

Check for overwrites in large allocations

```
# errctrl -c alloc.xmdbg large_trailer=<frequency>
```

This option sets the frequency of trailers that are added to allocations that require at least a full page. The page size depends on the heap. This technique catches the same type of errors as a redzone, but a redzone always starts at the next page boundary, and a trailer follows immediately after the bytes that are beyond the requested size. (A redzone page is a page that will cause an invalid page fault if it is referenced. This is a technique used to detect overflow from any area and is often used to protect stacks. xmalloc constructs redzone pages immediately following selected heap memory regions that it allocates.) Trailers are checked at free time for consistency. The error disposition can be affected for these checks just as it is for the small_trailer option. Trailers and redzones can be used together to ensure overruns are detected. Trailers are not used if the requested size is exactly a multiple of the page size. Overwrites can still be detected using the redzone option.

This option is new in AIX V6.1. In AIX 5L V5.3, all trailers are controlled with a single tuning parameter (alloc_trailer).

Check for overwrites in all allocations

```
# errctrl -c alloc.xmdbg alloc_trailer=<frequency>
```

This option is provided for compatibility. It sets the frequency that xmalloc will add a trailer to all allocations. To accomplish this, it overwrites the settings of both the small_trailer and large_trailer options.

Promote fragment allocations to whole pages

```
# errctrl -c alloc.xmdbg promote=<size>,<frequency>
```

This option sets the frequency for which allocations are promoted. When an allocation that is less than half of a 4 KB page is promoted, the returned pointer is as close to the end of the page as possible while satisfying alignment restrictions and an extra redzone page is constructed after the allocated region. No other fragments are allocated from this page. This provides isolation for the returned memory and catches users that overrun buffers. When used in conjunction with the df_promote option, this also helps catch references to freed memory. This option uses substantially more memory than other options. Sizes that are greater than 2 KB are still promoted in the sense that an extra redzone page is constructed for them.

The page size of the heap passed to xmalloc makes no difference. If the heap normally contains 64 KB pages (kernel_heap or pinned_heap on a machine that supports a 64 KB kernel heap page size), the returned memory of a promoted allocation will still be backed by 4 KB pages. These promoted allocations come from a region that has a 4 KB page size, to avoid using an entire 64 KB page as a redzone.

The supported sizes are all powers of two: 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, and 32768. All the commands accept hexadecimal numbers (0x10, 0x20, 0x40,...,0x8000) and decimal numbers as input.

In AIX 5L V5.3, this feature does not provide a redzone page, and always causes the freeing of the fragment to be deferred. See the discussion of deferred free option below. The following command needs to be use in AIX 5L V5.3 to provide a redzone page:

```
errctrl -c alloc.xmdbg doublepage_promote=<size>,<frequency>
```

In AIX V6.1, this option is still available, but the function is identical to the promote option. AIX V 6.1 offers another tuning parameter to control the deferral of promoted fragments in contrast to the deferral of other types of allocations. See the next section for more details.

Change the promotion settings of all sizes at once

```
# errctrl -c alloc.xmdbg promote_all=<frequency>
```

This option duplicates the function of the promote option, but does not take size as an argument. It applies the input frequency to all the promotion sizes with a single command. This option is new in AIX V6.1

Defer the freeing of pages and promoted allocations

```
# errctrl -c alloc.xmdbg df_promote=<frequency>
```

The deferred free technique means that when a memory object is freed, xmalloc will take measures to ensure that the object is not re-allocated immediately. This technique helps catch references to memory that has been freed. This option affects the freeing of promoted fragments. It sets the frequency with which the freeing of promoted fragment is deferred. Page promotion (for example, the promote option) and df_promote are designed to be used together.

This tuning parameter is new in AIX V6.1. The re-allocation of promoted allocations is always deferred in AIX V5.3.

Defer the freeing of pages and small allocations

```
# errctrl -c alloc.xmdbg def_free_frag=<frequency>
```

This option sets the frequency at which non-promoted fragments will be deferred. The difference between this option and the df_promote options must be clarified. A memory page that xmalloc manages contains multiple fragments of the same size or is part of a range of pages. When the def_free_frag option is in use, the freeing of every fragment on a page will be deferred together. This implies the number of pages used by these two techniques is substantially different. The df_promote option constructs one fragment per page (with an additional redzone page), and the def_free_frag option constructs multiple fragments per page with no redzone. This tuning parameter is new in AIX V6.1.

Defer the freeing of pages and large allocations

```
# errctrl -c alloc.xmdbg deferred_free=<frequency>
```

This option also helps catch references to memory that has been freed. It sets the frequency at which xmalloc defers the freeing of larger allocations. Larger allocations are at least one entire 4K page in size. This option should be used with care because it can be expensive from a performance standpoint. When large ranges are freed and deferred, all the pages in the range are disclaimed. Presuming there is no error, all the memory will be faulted and zero filled the next time it is referenced. Read references to freed memory are medium severity errors, while write references always cause a system to crash. If the disposition of medium severity errors is set to cause a system crash, the system will crash on a read reference.

This tuning parameter exists in AIX 5L V5.3, but it affects all allocations.

Redzones for large allocations

```
# errctrl -c alloc.xmdbg redzone=<frequency>
```

This option sets the frequency of redzone page construction. This option is specific for allocations of a page or more. With default error disposition in effect, read references to redzone pages will cause an error log event, and write references will cause a system crash. As in other cases, the user can change the error disposition of medium severity errors to cause a system crash on a bad read reference.

VMM page state checks

```
# errctrl -c alloc.xmdbg vmmcheck=<frequency>
```

This option sets the frequency at which xmfree will check page protection settings, storage key bits, and pin counts for memory being freed back to a heap. Some errors in this area are not fatal. For example, a page that has a higher than expected pin count at free time will waste pinned storage, but there are usually no fatal consequences. When a page is returned that has a lower than expected pin count, or has the wrong page protection settings, or has the wrong hardware storage key associated with it, the system will crash.

Tuning parameters not affected by RTEC level

The following tuning parameters are not affected by the error checking level configuration: memleak_pct, memleak_count, minsize, reset_errlog_count, and deferred_count.

Set memory leak percentage

```
# errctrl -c alloc.xmdbg memleak_pct=<percentage>
```

This option sets the percentage of heap memory that can be consumed before an error is logged. This is specific to the heaps controlled by the component. Heaps that are controlled by other components are not affected. For example alloc.heap0 is a separate component that controls the heap used by the loader, and it uses a different percentage than the kernel_heap, which is controlled by alloc.xmdbg. Component level heaps created by the heap_create kernel service can be registered separately and can be given different percentages.

For example, # **errctrl -c alloc.xmdbg memleak_pct=50** will cause an error to be logged if 50% of a system heap is consumed. This command requires the user to make a judgment about how much storage should be consumed before a leak should be suspected. Users who do not have that information should not use the command. The current values that reflect the percentage can be viewed with the **xm -Q** command. The output appears near the bottom:

```
KDB(0)> xm -Q
XMDBG data structure @ 0000000002523050

... omitted lines ...

Ratio of memory to declare a memory leak: 0x400(1024)/0x400(1024)
Outstanding memory allocations to declare a memory leak: -1
Deferred page reclamation count (-1 == when necessary): 16384
Minimum allocation size to force a record for: 1048576
```

Note that the default percentage is 100% (1024/1024). Memory leak errors are classified as a low severity errors and the default disposition is to ignore them. The error disposition for low severity errors can be modified to log an error or to cause a system crash.

Set memory leak count

```
# errctrl -c alloc.xmdbg memleak_count=<num>
```

This option sets an outstanding allocation limit for all the fragment sizes. This is meant as an aid in catching memory leaks that are very slow growing. If the total number of outstanding allocations of any fragment size grows beyond this limit, an error is logged. For example, an error occurs if the limit is set to 20,000, and 20,001 allocations are outstanding for any of the fragment sizes. This error is classified as a low severity error and the default disposition for the error is to ignore it. The error disposition for low severity errors can be modified to log an error or to cause a system crash. The default value of this setting is -1, meaning no check is made. This limit must be set to a positive value by the operator to cause the check to be made.

The **xm -Q** command shows the current setting of this value near the bottom of the output:

```
KDB(0)> xm -Q
XMDBG data structure @ 0000000002523050

... omitted lines ...

Ratio of memory to declare a memory leak: 0x400(1024)/0x400(1024)
Outstanding memory allocations to declare a memory leak: -1
```

```
Deferred page reclamation count (-1 == when necessary): 16384
Minimum allocation size to force a record for: 1048576
```

In AIX 5L V5.3, this option counts the total number of outstanding allocations. In AIX V6.1, a separate count of allocations of each different size has been implemented and AIX V6.1 xmalloc RTEC reports if any of them is growing beyond the provided limit. This enhancement avoids bookkeeping on each allocation and consequently improves performance.

Set large allocation record keeping

```
# errctrl -c alloc.xmdbg minsize=<num>
```

This sets the size of an allocation that we will always record. Very large allocations are frequently never freed, so this setting allows the operator to record all outstanding allocations that are greater than or equal to minsize bytes. The default value of this tuning parameter is 0x1000000 bytes. The **xm -Q** command shows the current setting near the bottom of the output:

```
KDB(0)> xm -Q
XMDBG data structure @ 00000000002523050

... omitted lines ...

Ratio of memory to declare a memory leak: 0x400(1024)/0x400(1024)
Outstanding memory allocations to declare a memory leak: -1
Deferred page reclamation count (-1 == when necessary): 16384
Minimum allocation size to force a record for: 1048576
```

Reset error log handling

```
# errctrl -c alloc.xmdbg reset_errlog_count
```

To avoid the error log from being flooded, each subcomponent of the alloc component will only record up to two hundred errors in the error log before reaching a threshold. This threshold can be reset with this option. If the two hundred log limit is reached and the count is not reset, error logging by the component will not resume until after a partition reboot.

In the previous AIX release, a separate count is kept for many different errors, and only one error of each type is logged.

Set the deferral count

```
# errctrl -c alloc.xmdbg deferred_count=<num>
```

The deferral count is the total number of pages that are deferred before xmalloc recycles deferred storage back to a heap. It is obvious that the freeing of storage cannot be deferred indefinitely, but it might not be obvious that the consequence of deferring too long is that heaps can become fragmented, which could result in allocation failures for large requests. xmalloc supports setting this option to -1, which causes xmalloc to defer re-allocation as long as possible. This means the heap is exhausted before memory is recycled. In AIX V6.1, the default value is 0x4000 deferrals. (In AIX 5L V5.3, the default is 0x100 deferrals.) In general, this value should only be changed with component owner guidance. The **xm -Q** command shows the current setting of this tuning parameter near the bottom of the output:

```
KDB(0)> xm -Q
XMDBG data structure @ 0000000002523050

... omitted lines ...

Ratio of memory to declare a memory leak: 0x400(1024)/0x400(1024)
Outstanding memory allocations to declare a memory leak: -1
Deferred page reclamation count (-1 == when necessary): 16384
Minimum allocation size to force a record for: 1048576
```

The **errctrl** command can be used to display the alloc portion of the RAS component hierarchy and each sub-component RTEC attributes. The **errctrl** command also allows you to modify the runtime error checking related configuration parameters. The following **errctrl** command output shows that the default error checking level for all system memory allocator components is normal (level=3), and that low-severity errors are ignored (LowSevDis=48). For the alloc.heap0 component, medium-severity errors are logged (collect service data and continue) (MedSevDisp=64). In case of the alloc.xmdbg component, a medium-severity error initiates a live dump (MedSevDisp=80):

```
hhaix6:root:/root # errctrl -c alloc -q -r
---------------------------------------------+-------+-------+-------+--------
                                             | Have  |ErrChk |LowSev |MedSev
    Component name                           | alias | /level| Disp  | Disp
---------------------------------------------+-------+-------+-------+--------
alloc
     .heap0                                  |  NO   | ON /3 |  48   |  64
     .xmdbg                                  |  NO   | ON /3 |  64   |  80
```

The kernel debugger can be used from the command line to examine the values of all the frequency settings as follows:

```
# echo xm -Q | kdb
```

4.3 Dump facilities

With AIX, the *traditional* dump, also called the *legacy* dump, is taken when a crash occurs before a system logical partition or full system partition is reinitialized. A dump is a picture of partition memory and the processor state. It is initiated by a user request or by AIX when a severe error is detected and the operating system must be halted.

Note: A user-initiated dump is different from a dump initiated by an unexpected system halt because the user can designate which dump device to use. When the system halts unexpectedly, a system dump is initiated only to the primary dump device.

As many systems have a large amount of memory, the time to dump has increased significantly and has a significant impact on the system outage time. Several technologies have been introduced recently within AIX to address this issue:

- *Minidump* facility, starting with AIX 5L V5.3 TL03

  The minidump is a small compressed dump that is stored to NVRAM when the system crashes or a dump is initiated, and then written to the error log on reboot. It can be used to see some of the system's state and do some debugging when a full dump is not available. It can also be used to get a quick snapshot of a crash without having to transfer the entire dump from the crashed system.

- *Parallel dump* facility, starting with AIX 5L V5.3 TL05

  A new optimized compressed dump format is introduced in AIX 5L V5.3 TL05. The dump file extension for this new format is still .BZ. Parallel dumps are produced automatically when supported by the AIX release. In this new compressed dump file, the blocks are compressed and unordered; this unordering feature allows multiple processors to dump in parallel sub-areas of the system. Thus, when a system dump happens on a multiprocessor system, the time to produce the dump image is now I/O bound limited and so greatly reduced.

This new file format for parallel dump is no more readable when using the usual **uncompress** and **zcat** commands; the new **dmpuncompress** command must be used. In order to increase dump reliability, a new -S checking option, that is used with the -L option for the statistical information about the most recent dump, is also added to the **sysdumpdev** command. The -S option scans a specific dump device and sees if it contains a valid compressed dump.

- *Component dump* facility, starting with AIX V6

  The enterprise Reliability Availability Serviceability strategy is to maintain the *continuous availability* of System p servers through extended key error detection and recovery capabilities implementing mainframe-like features for the hardware, AIX operating system, and also for external third-party software. In order to provide a granular approach to RAS, enterprise RAS defines a component framework where AIX and third-party software can register components that enable their specific RAS features, such as trace, dump, and error checking features.

  The Component Trace facility (CT), like the Runtime Error checking (RTE) facility, has been implemented for the AIX Operating system components with AIX 5L V5.3 TL05. For additional informations on these facilities, see *AIX 5L Differences Guide Version 5.3 Addendum*, SG24-7414.

  The Component Dump facility (CD) for the AIX operating system components is now introduced with AIX Version 6.

- *Live dump* facility, starting with AIX V6

  Live dumps are small dumps that do not require a system restart. The Live Dump facility uses the Component Dump implementation to dump only AIX components, registered as a live dump enabled component, that are a live dump aware component. Software or system administrators can initiate live dumps while the system is running; planned downtime is no longer necessary to dump a system. Moreover, because selective dump aware components can be chosen, the live dump facility reduces significantly the time to dump and the size requirement for dump files.

- *Firmware-assisted dump* facility, starting with AIX V6

  The firmware-assisted dump means that an AIX dump is taken while the partition is restarting. This increases the reliability of a partition system dump by minimizing the work done by the failing operating system and lets it be done by the new restarting instance. The firmware is involved to preserve the memory area across the reboot.

4.3.1 The dumpctrl command

The **dumpctrl** command is a new, integrated interface to manage the various dump formats.

With AIX Version 6, the implementation of the AIX dump components provides an enhanced dump granularity and allows you to dump these components without requiring a reboot. Thus, this new dump capability, based on these components, is called a *live dump*. Before AIX Version 6, the only supported type of dump was the *system dump*, which requires a reboot afterwards.

As shown in Figure 4-2, to manage the attributes of these two different types of dumps, AIX provides a unified user-interface through the **dumpctrl** command to manage both:

- The *traditional* dump, also called the *system dump*, which requires a reboot,
- The *live dump*, based on the new dump components, which is implemented with the component infrastructure that allows you to make a dump while the server is running.

> **Important:** Only the root user can use the **dumcptrl** command.

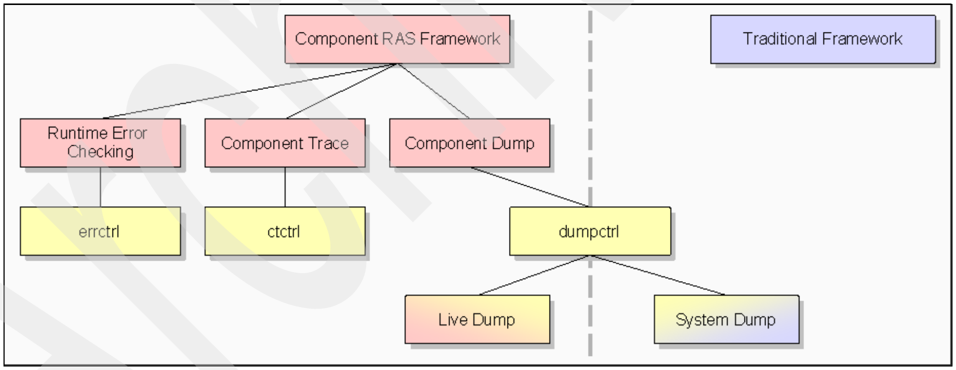


Figure 4-2 Two dump frameworks, a unified user-interface: dumpctrl

Regarding the SMIT panels, each type of dump keeps its own SMIT menu, as shown by the SMIT Problem Determination panel shown in Figure 4-3.

- To call directly the SMIT system dump panel, use **smitty dump**.
- To call directly the SMIT live dump panel, use **smitty ldmp**.

```
                              Problem Determination

Move cursor to desired item and press Enter.

  Electronic Service Agent
  Error Log
  Trace
  System Dump
  Alog
  Change/Show/Reset Core File Copying Directory
  Hardware Diagnostics
  Verify Software Installation and Requisites
  Advanced First Failure Data Capture Features
  Kernel Recovery
  Component / Live Dump
  ProbeVue
  Storage Protection Keys





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

Figure 4-3 Problem Determination SMIT panel

The following sections describe the main capabilities of the **dumpctrl** command:

- To show dump components
- To control live dump attributes
- To control system dump attributes with the description of a new system dump type, based on POWER6 firmware, named firmware-assisted dump.

4.3.2 Component dump facility

With AIX Version 6, AIX dump components are available. They have been registered through the RAS Component framework. The following example shows how to register a dump component:

```
/*
 * This sample creates a component, makes it dump-aware, and handles
both live
 * and system dump.
 */
```

```
...lines missing for clarity
#include <sys/ras.h>
#include <sys/livedump.h>
#include <sys/kerrnodefs.h>
#include <sys/eyec.h>
#include <sys/raschk.h>
...lines missing for clarity
/* Component name and handle */
const char Compname[] = "sample_comp";
ras_block_t Rascb=NULL;
...lines missing for clarity
{
   kerrno_t rv = 0;
   int rc;

      /* Unloading */
      if (Rascb) ras_unregister(Rascb);
...lines missing for clarity
   /* Register the component as dump aware */
   rv = ras_register(&Rascb,
      (char*)Compname,
      (ras_block_t)0,
      RAS_TYPE_OTHER,
      "sample component",
      RASF_DUMP_AWARE,
      sample_callback,
      NULL);
   if (rv) return(KERROR2ERRNO(rv));
...lines missing for clarity
   /* Make the component system and live dump aware. */
   rv = ras_control(Rascb, RASCD_SET_SDMP_ON, 0, 0);
   if (rv) return(KERROR2ERRNO(rv));
   rv = ras_control(Rascb, RASCD_SET_LDMP_ON, 0, 0);
   if (rv) return(KERROR2ERRNO(rv));
...lines missing for clarity
   rv = ras_customize(Rascb);
   if (rv) return(KERROR2ERRNO(rv));
...lines missing for clarity
```

Note: The flag RASF_DUMP_AWARE indicates what type of RAS systems this component is aware of. With RASF_DUMP_AWARE, this component is a dump aware component.

- The **RASF_SET_SDMP_ON** command makes this component system dump aware.
- The **RASF_SET_LDMP_ON** command makes this component live dump aware.

The new **dumpctrl** command modifies or displays the dump attributes of system components.

The following example shows the dump properties of the jfs2 component:

```
# dumpctrl -qc -c jfs2
-----------------------------------------------+------+-----------+------------
                                               | Have | Live Dump | System Dump
      Component Name                           |Alias |  /level   |  /level
-----------------------------------------------+------+-----------+------------
jfs2                                            |   NO |    ON/3    |   OFF/3
```

Note: The **dumpctrl -qc** command lists all of the dump component hierarchy.

Since the **dumpctrl** command is a unified interface for both live dump and system dump, it displays concurrently both the two dump aware capabilities of the component:

- *Component* type for Live Dump Level

  Refers to a component specified with the RAS infrastructure (one created with the ras_register() kernel service call).

- *Legacy component* type for System Dump Level

  Refers to a dump component specified with either the dmp_add() or the dmp_ctl() kernel services, which refers to the traditional AIX system dump.

For example, the lvm component is supported by both frameworks. This means that two dump components for lvm are implemented for each dump framework:

```
# dumpctrl -qc
-----------------------------------------------+------+-----------+------------
                                               | Have | Live Dump | System Dump
      Component Name                           |Alias |  /level   |  /level
-----------------------------------------------+------+-----------+------------
...lines missing for clarity
lvm                                            |   NO |   ON/3    |   ON/3
     .rootvg                                   |   NO |   ON/3    |   ON/3
```

```
          .metadata                            |  NO |   ON/3   |   ON/3
               .lvs                            |  NO |   ON/3   |   ON/3
                    .fslv00                    |  NO |   ON/3   |   ON/3
                    .fslv01                    |  NO |   ON/3   |   ON/3
                    .fslv02                    |  NO |   ON/3   |   ON/3
                    .fslv03                    |  NO |   ON/3   |   ON/3
                    .fslv04                    |  NO |   ON/3   |   ON/3
                    .fslv05                    |  NO |   ON/3   |   ON/3
...lines missing for clarity
```

The **dumpctrl** command is able to list live dumps with the specified components:

```
# dumpctrl -h
...lines missing for clarity
-s    : List live dumps in the dump repository.
...lines missing for clarity
Selector: either "-c all" or one or more of
-c list : comma- or space-separated list of component names,
-l list : comma- or space-separated list of component aliases,
-t list : comma- or space-separated list of type or type_subtype names
-C name : failing component name (only valid with -s)
-L name : failing component alias (only valid with -s
-T name : failing component type_subtype name (only valid with -s)
```

An output of the -s option when no live dump exists is shown in the following output:

```
# dumpctrl -s
The live dump repository located at:
/var/adm/ras/livedump
contains no live dumps that match the specified parameters (if any).
```

SMIT panels (Figure 4-4 and Figure 4-5 on page 157) are also available to modify the dump component attributes under the main menu **smitty ldmp**.

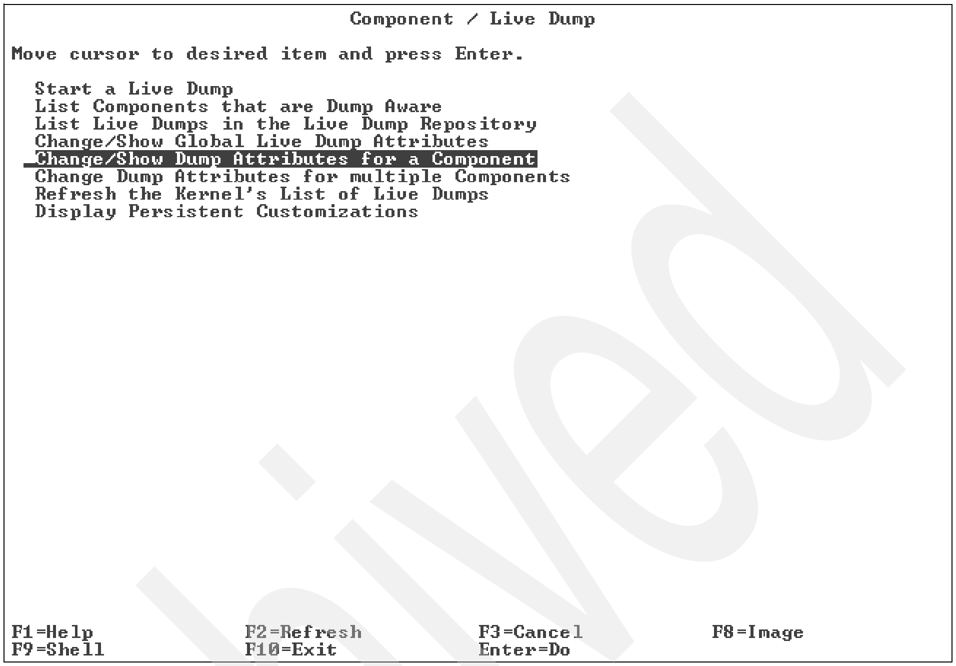
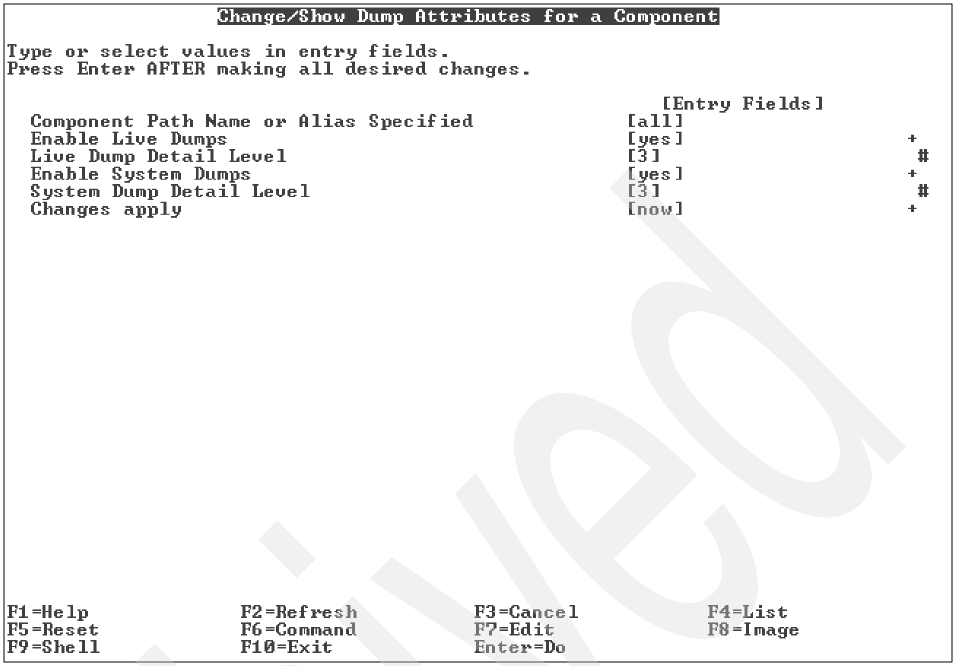


Figure 4-4 SMIT Panel to request change/show the dump component attributes


```
                  Change/Show Dump Attributes for a Component

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

                                                          [Entry Fields]
  Component Path Name or Alias Specified              [all]
  Enable Live Dumps                                   [yes]              +
  Live Dump Detail Level                              [3]                 #
  Enable System Dumps                                 [yes]              +
  System Dump Detail Level                            [3]                 #
  Changes apply                                       [now]              +

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

Figure 4-5 SMIT Panel to change/display Dump attribute for a component

4.3.3 Live dump facility

The live dump facility uses the Component Dump framework to dump only AIX components registered as live dump aware components.

Software or system administrators can initiate live dumps while the system is running: planned downtime is no longer necessary to dump a system.

- Software programs can use live dumps as part of recovery actions.
- A system administrator can initiate live dumps when a subsystem does not respond or behaves erroneously.

The live dump is intended to provide dump capability to the kernel and extensions when the system is still functional.

> **Important:** Live dump should not be used if the system is not entirely functional. If no tasks can be dispatched or the system cannot perform I/O, then the system dump should be used instead. Live dump should not be used as the dump choice when a complete system failure is determined.

Live dump file space

Because selective live dump aware components can be chosen, the live dump facility significantly reduces the time required to dump and the size requirement for the dump files.

By default, live dumps are written to the /var/adm/ras/livedump directory. The live dump directory can be changed by using the **dumpctrl** command.

> **Note:** Unlike system dumps that are written to a dedicated dump device, live dumps are written to the file system. A best practice is to maintain the live dump directory on rootvg, and ensure enough space.

The contents of the livedump repository can be shown using the **dumpctrl -s** command or by selecting "List Live Dumps in the Live Dump Repository" from the Component/Live Dump SMIT menu. (Figure 4-6).

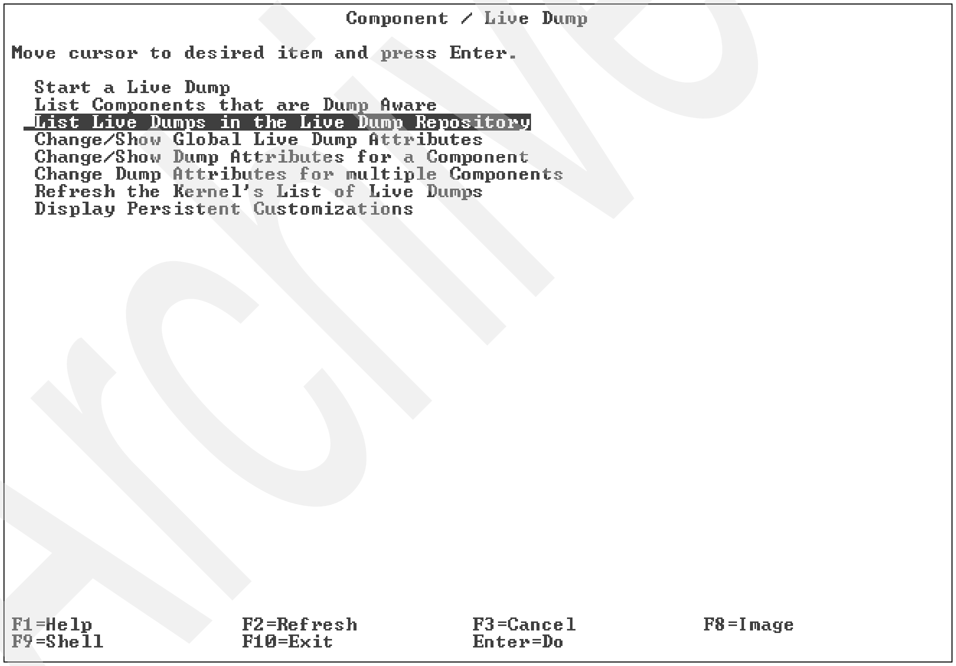


Figure 4-6 SMIT Live dump panel: smitty ldmp

Live dump file size

To control the size of a live dump, each dump component is required to limit the size of its dump data. This is controlled by the dump detail level, a value between 0 to 9, that can be changed by using the **dumpctrl** command.

There are three main live dump levels: ldmpminimal, ldmpnormal, and ldmpdetail, corresponding to levels 1, 3, and 7. This limits the size of the data dump for each dump component. Therefore, the live dump file size depends on the number of selected dump components.

Table 4-3 shows the recommended and upper limit values for component dump data given the dump detail level.

Table 4-3 Dump detail level and component dump data size limit

Level	Suggested maximum	Enforced maximum
Less than ldmpnormal (0, 1, 2)	1 MB	2 MB
Less than ldmpdetail (3, 4, 5, 6)	2 MB	4 MB
Less than 9 (7, 8)	4 MB	8 MB
9	No limit	No limit

If the system is unable to write the current dump due to an I/O error, an error is logged. If the dump is designated as a one-pass dump, it is kept in memory until it can be saved using the **`dumpctrl -k`** command. This command is run automatically every five minutes.

Serialized live dump

There are two ways to take a live dump:

- A serialized live dump:

  All processors are stopped while dumping.

- An unserialized live dump:

  The processors are operating.

> **Important:** In AIX Version 6.1, live dumps are only serialized live dumps.

A serialized live dump causes a system to be frozen or suspended when data is being dumped. The freeze is done by stopping all processors, except the one running the dump. Such a freeze should not exceed one tenth of a second.

This value can be modified by the **`dumpctrl`** command. We recommend using the default value. If the freeze period exceeds five seconds, the system is unfrozen, and only dump data gathered so far is written.

When the system is frozen, the data is copied into a pre-allocated pinned memory. This dedicated pinned memory is called the *live dump heap*.

The data is written to the file system only after the system is unfrozen.

Live dump heap size

The default live dump heap size is the minimum of 64 MB and 1/64th the size of physical memory. It will not be less than 4 MB.

The maximum heap size is also limited to 1/16th the size of real memory.

Table 4-4 provides live dump heap size limits for several real memory sizes.

Table 4-4 Live dump heap size limits

Size of real memory	Default heap size	Min. heap size	Max. heap size
128 MB	4 MB	4 MB	8 MB
256 MB	4 MB	4 MB	16 MB
1 GB	16 MB	4 MB	64 MB
4 GB	64 MB	4 MB	256 MB
16 GB	64 MB	4 MB	1 GB

The heap size can be changed dynamically using the **`dumpctrl`** command or by way of dynamic reconfiguration, that is, adding or removing real memory.

Managing the live dump heap content

Duplicate live dumps that occur rapidly are eliminated to prevent system overload and to save file system space. Eliminating duplicate dumps requires periodic, once every 5 minutes by default, scans of the live dump repository. This is done by calling **`/usr/sbin/dumpctrl -k`** using an entry in the root user's crontab. This period can only be changed by editing the crontab.

To eliminate duplicate dumps, the **`dumpctrl-k`** command uses the following policies that can be changed by the **`dumpctrl`** command:

Pre-capture policy Pre-capture elimination is designed to prevent duplicate live dumps. It uses an age limit. When checking for duplicates, only dumps not older than a day (86400 seconds) will be considered.

Post-capture policy Post-capture elimination is used to remove low priority live dumps when a higher priority dump must be written, and the file system free space is low.

A live dump has a priority of either *info* or *critical*, for informational or critical dumps. The default is critical. If, while writing a critical dump, the system runs out of space, post-capture elimination removes live dumps with info priority, starting with the oldest one, until the critical dump can be written.

All policy Pre-capture elimination and post-capture elimination are both in effect.

None policy No live dump elimination is performed.

There is a free space percentage associated with the live dump repository. When the free space falls below this percentage, the system logs an error message to the error log. As shown in Figure 4-7, the free space is 22% while the desired limit is at 25%, the default value. The system administrator should increase the file system size or delete the live dumps no longer desired. The contents of the live dump directory can be displayed with the **dumpctrl -s** command.

```
---------------------------------------------------------------------------
LABEL:          DMPCHK_LDMPFSFULL
IDENTIFIER:     4BE53A52

Date/Time:       Tue Oct 16 15:00:01 GMT+02:00 2007
Sequence Number: 145
Machine Id:      00C1F1704C00
Node Id:         lpar01
Class:           O
Type:            PEND
WPAR:            Global
Resource Name:   dumpcheck

Description
Livedump filesystem almost full

        Recommended Actions
        Expand filesystem or delete dumps that are not needed

Detail Data
percent free
          22
desired percent free
          25
FILE SYSTEM MOUNT POINT
@
@
@
```

Figure 4-7 The freespc parameter and error log

Live dump attributes

With the **dumpctrl** command, all the described live dump attributes can be set with the form:

```
dumpctrl attribute1=value1 attribute2=value2
```

To display live dump attributes, use the -ql option of the **dumpctrl** command:

```
dumpctrl -ql
```

The following example shows how to display and modify live dump attributes controlling the live dump directory and the live dump detail level. Note that the live dump directory is also known as the live dump repository:

```
# dumpctrl -ql
Live Dump Enabled:                      yes
Live Dump Directory:                    /var/adm/ras/livedump
Live Dump Free Space Threshold:         25%
Live Dump Global Level:                 3
Live Dump Heap Size:                    0 MB (0 indicates default heap size)
Live Dump Duplicate Suppression Type:   all
Live Dump Max System Freeze Interval:100ms
# dumpctrl ldmpdetail dir=/tmp
# dumpctrl -ql
Live Dump Enabled:                      yes
Live Dump Directory:                    /tmp
Live Dump Free Space Threshold:         25%
Live Dump Global Level:                 7
```

The live dump attributes can also be modified using the SMIT panel shown in Figure 4-8 under the main menu **smitty ldmp**.

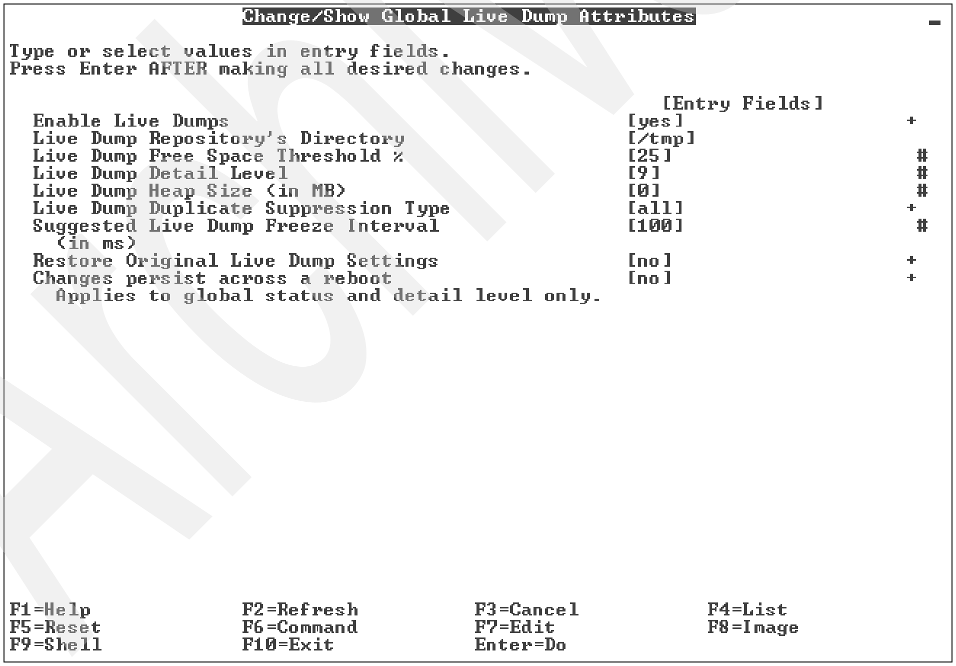


Figure 4-8 SMIT panel to change live dump attributes

Table 4-5 on page 163 provides all the live dump options that can be set by the **dumpctrl** command.

Table 4-5 Live dump attributes and defaults

Attribute	Specification	Default value
dir	Specifies a live dump directory name.	/var/adm/ras/livedump
freespc	Specifies a live dump free space threshold using a decimal value from 0 to 99.	25 (means 25%)
ldmpenable	Specifies whether a live dump is enabled. The possible values are yes and no; the ldmpon attribute can be used instead of ldmpenable=yes, and the ldmpoff attribute instead of ldmpenable=no.	yes
ldmplevel	Specifies the live dump level using a decimal value from 0 to 9; the ldmpminimal, ldmpnormal, or ldmpdetail attributes can be used instead of ldmplevel=1, 3, 7.	3 (normal)
heapsz	Specifies the live dump heap size using a decimal value in megabytes. A value of 0 indicates that the formula for the heap size mentioned previously is to be used.	0
duptype	Specifies the duplicate dump suppression type. The following are the possible values: all, pre, post, and none.	all
maxfreeze	Specifies the maximum recommended system freeze interval using a decimal number in milliseconds.	100 ms

The persistence of an attribute refers to how attributes are applied. They may be applied immediately, to new components only, and remain in effect across a reboot. Table 4-6 provides the persistence of live dump attributes.

Table 4-6 Live dump attributes and persistence

Attribute	Specification	Persistence
ldmpenable	Live dump enabled	The **bosboot** command is required using the -P flag.
dir	Live dump directory	Takes effect immediately and upon system reboot.
freespc	Live dump free space threshold	Takes effect immediately and upon system reboot.

Attribute	Specification	Persistence
ldmplevel	Live dump level	The **bosboot** command is required using the -P flag.
heapsz	Live dump heap size	Takes effect immediately and upon system reboot.
duptype	Duplicate dump suppression type	Takes effect immediately and upon system reboot.
maxfreeze	Maximum recommended system freeze interval	Takes effect immediately and upon system reboot.

Some of the error log and dump commands are delivered in the bos.sysmgt.serv_aid package.

Live dump commands included in bos.sysmgt.serv_aid include the **livedumpstart** command.

A live dump may also be initiated from the AIX kernel or from a kernel extension.

For additional information, see "Live Dump Facility" in *Kernel Extensions and Device Support Programming Concepts*, found at:

http://publib.boulder.ibm.com/infocenter/pseries/v5r3/topic/com.ibm.aix.kernelext/doc/kernextc/kernextc.pdf

4.3.4 System dump facility

A system generates a system dump when a severe error occurs. System dumps can also be user-initiated by system administrators. A system dump creates a picture of the system's memory contents. System administrators and programmers can generate a dump and analyze its contents when debugging new kernel extensions. Note that the live dump is also a good tool for debugging new code.

The system dump facility is based on the existing dump framework, but has evolved in AIX Version 6 to use the dump granularity provided by the Dump Components.

Some of the Dump Components can be system-aware (for more details, see 4.3.2, "Component dump facility" on page 152), allowing granular control of the amount of data that is dumped in a system dump. Components that are system-dump aware can be excluded from a system dump to reduce the dump size. To see or modify which dump components are system dump aware or selected for a system, the **dumpctrl** command or SMIT panels can be used (see Figure 4-4 on page 156 and Figure 4-5 on page 157).

The system dump is intended to provide dump capability to the kernel and extensions when a severe error occurs, and the kernel has to halt the system.

When a system dump occurs, the partition or server is stopped with an 888 number flashing in the operator panel display, indicating the system has generated a dump and saved it to a dump device. This is only for the traditional system dump. The firmware-assisted system dump, new in Version 6.1, is saved while the operating system is re-booting.

System dump attributes

With the **dumpctrl** command, all the described system dump attributes can be set with the form:

```
dumpctrl attribute1=value1 attribute2=value2
```

To display system dump attributes, use the -qs option of the **dumpctrl** command:

```
dumpctrl -qs
```

The following example shows how to display and modify system dump attributes concerning the copy directory and the level of detail:

```
# dumpctrl -qs
Dump Legacy Components:                yes
System Dump Global Level:              3
System Dump Copy Directory:            /var/adm/ras
Display Boot Time Menu on Dump Failure: yes
Allow Dump on Keyboard Sequence:       no
Primary Dump Device:                   /dev/hd6
Secondary Dump Device:                 /dev/sysdumpnull
# dumpctrl sdmpdetail
# dumpctrl -qs
System Dump Global Level: 7
```

The system dump attributes can be also modified with the SMIT panel shown in Figure 4-10 on page 169 under the main menu **smitty dump**.

Table 4-7 provides all the system dump options that can be set by the **dumpctrl** command.

Table 4-7 System dump attributes and defaults

Attribute	Specification	Default value
sdmpenable	Specifies whether the system dump is enabled. The possible values are yes and no: sdmpon can be used instead of sdmpenable=yes and sdmpoff instead of sdmpenable=no.	yes
legacyenable	Specifies whether the legacy dump components are enabled. The possible values are yes and no: legacyon can be used instead of legacyenable=yes and legacyoff instead of legacyenable=no.	yes
sdmplevel	Specifies the system dump level using a decimal value from 0 to 9. You can specify the sdmpminimal, sdmpnormal, or sdmpdetail attribute instead of sdmplevel=1, 3, 7.	3 (normal)
copydir	Specifies a copy directory path name.	/var/adm/ras
forcecopy	Specifies whether the forcecopy attribute is enabled. The possible values are yes and no.	yes
keyseq	Specifies whether the key sequences at operator panel always cause a dump. The possible values are yes and no.	no
primary	Specifies the primary dump device path name.	/dev/hd6 or /dev/lg_dumplv
secondary	Specifies the secondary dump device path name.	/dev/sysdumpnull

The persistence of an attribute refers to how attributes are applied. They may be applied immediately, to new components only, and remain in effect across a reboot. Table 4-8 on page 167 provides the persistence of system dump attributes.

Table 4-8 System dump attributes and persistence

Attribute	Specification	Persistence
sdmpenable	System dump enabled.	The **bosboot** command with the -P flag is required.
legacyenable	Dump legacy components.	Takes effect immediately and upon system reboot. No **bosboot** command with the -P flag is required.
sdmplevel	System dump level.	The **bosboot** command using the -P flag is required.
copydir	A copy directory path name.	Takes effect immediately and upon system reboot.
forcecopy	Brings up the boot time menu if it cannot make a copy.	Takes effect immediately and upon system reboot.
keyseq	Key sequences always cause a dump.	Takes effect immediately and upon system reboot.
primary	The primary dump device.	Takes effect immediately and upon system reboot. No bosboot command with the -P flag is required.
secondary	The secondary dump device.	Takes effect immediately and upon system reboot. No bosboot command with the -P flag is required.

Some of the error log and dump commands are delivered in the bos.sysmgt.serv_aid package. System dump commands included in the bos.sysmgt.serv_aid include the **sysdumpstart** command.

Firmware-assisted dump

With the introduction of the POWER6 processor based systems, system dumps can be assisted by firmware. Firmware-assisted system dumps are different from traditional system dumps that are generated before a system partition is reinitialized because they take place when the partition is restarting.

In order to improve fault tolerance and performance, disk writing operations are done as much as possible during the AIX Boot phase in parallel with the AIX initialization.

Figure 4-9 provides all dump capabilities and shows that the firmware-assisted dump is a new type of system dump compared to the traditional one of parallel dump since AIX V5.3 TL05 or the classic one for previous AIX versions.

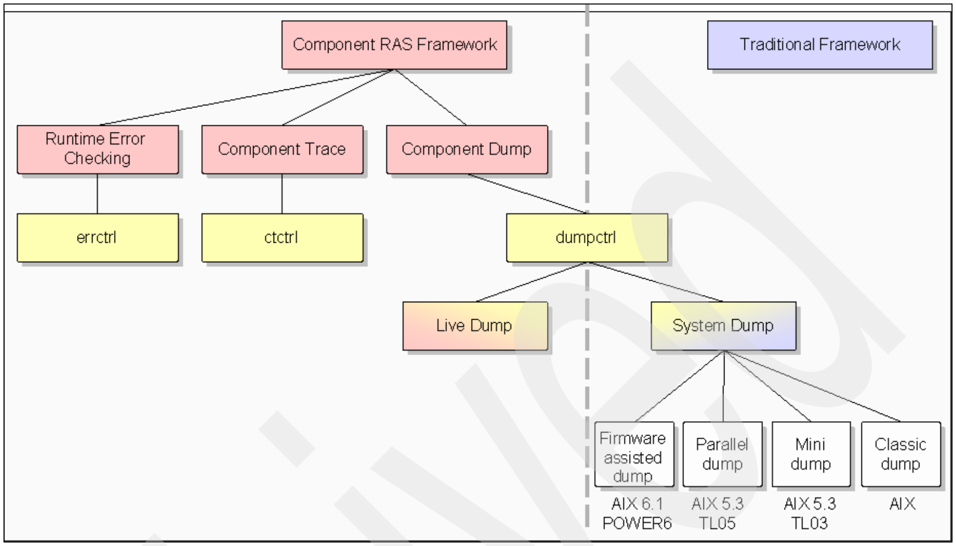


Figure 4-9 Overview of all dump capabilities

To select the type of system dump, a new entry, Change the Type of Dump, as shown in Figure 4-10 on page 169, is added to the SMIT main panel for dump: **`smitty dump`**.

The SMIT panel shown in Figure 4-11 on page 169 allows the administrator to choose between a traditional or a firmware-assisted dump. This choice can also be done on the command line with the new **`sysdumpdev`** option -t:

- To select a traditional system dump, run:

  ```
  sysdumpdev -t 'traditional'
  ```

- To select a fw-assisted system dump, run:

  ```
  sysdumpdev -t 'fw-assisted'
  ```

Important: AIX Version 6.1 generates a traditional system dump by default.

```
                                   System Dump

Move cursor to desired item and press Enter.

  Show Current Dump Devices
  Show Information About the Previous System Dump
  Show Estimated Dump Size
  Change the Type of Dump
  Change the Full Memory Dump Mode
  Change the Primary Dump Device
  Change the Secondary Dump Device
  Change the Directory to which Dump is Copied on Boot
  Start a Dump to the Primary Dump Device
  Start a Traditional System Dump to the Secondary Dump Device
  Copy a System Dump from a Dump Device to a File
  Always ALLOW System Dump
  Check Dump Resources Utility
  Change/Show Global System Dump Properties
  Change/Show Dump Attributes for a Component
  Change Dump Attributes for multiple Components






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

Figure 4-10   SMIT panel: type of system dump

With the menu selection shown in Figure 4-10, the following panel appears:

```
                             Change the Type of Dump

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

                                                         [Entry Fields]
  Type of Dump                                         fw-assisted            +








  +--------------------------------------------------------------------------+
  |                               Type of Dump                               |
  |                                                                          |
  | Move cursor to desired item and press Enter.                             |
  |                                                                          |
  |   traditional                                                            |
  |   fw-assisted                                                            |
  |                                                                          |
  | F1=Help                   F2=Refresh                F3=Cancel            |
F1| F8=Image                  F10=Exit                  Enter=Do             |
F5| /=Find                    n=Find Next                                    |
F9+--------------------------------------------------------------------------+
```

Figure 4-11   SMIT panel: traditional or firmware-assisted dump

Firmware-assisted system dumps can be one of these types:

Selective memory dump Selective memory dumps contain selected kernel information. Note that the traditional system dump is also a selective memory dump.

Full memory dump The whole partition memory is dumped without any interaction with the AIX instance that is failing.

To select the memory dump mode of the firmware-assisted dump, a new entry, Change the Full Memory Dump Mode, as shown in Figure 4-12, is added to the SMIT main panel for dump: **`smitty dump`**.

```
                                  System Dump

Move cursor to desired item and press Enter.

  Show Current Dump Devices
  Show Information About the Previous System Dump
  Show Estimated Dump Size
  Change the Type of Dump
  Change the Full Memory Dump Mode
  Change the Primary Dump Device
  Change the Secondary Dump Device
  Change the Directory to which Dump is Copied on Boot
  Start a Dump to the Primary Dump Device
  Start a Traditional System Dump to the Secondary Dump Device
  Copy a System Dump from a Dump Device to a File
  Always ALLOW System Dump
  Check Dump Resources Utility
  Change/Show Global System Dump Properties
  Change/Show Dump Attributes for a Component
  Change Dump Attributes for multiple Components






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

Figure 4-12 SMIT panel: Change the Full Memory Mode

The SMIT panel shown in Figure 4-13 on page 171 allows the administrator to choose the desired mode for the full memory dump. This choice can also be done on the command line with the new **`sysdumpdev -f`** option:

- To choose the selective memory mode, the full memory dump must be disallowed by running:

  `sysdumpdev -f ‘disallow’`

- To specify that the full memory dump is performed only if the operating system cannot properly handle the dump request, run:

  `sysdumpdev -f ‘allow’`

- To enforce the full memory system dump, it is always performed by running:

  `sysdumpdev -f ‘require’`

The disallow option is the default.

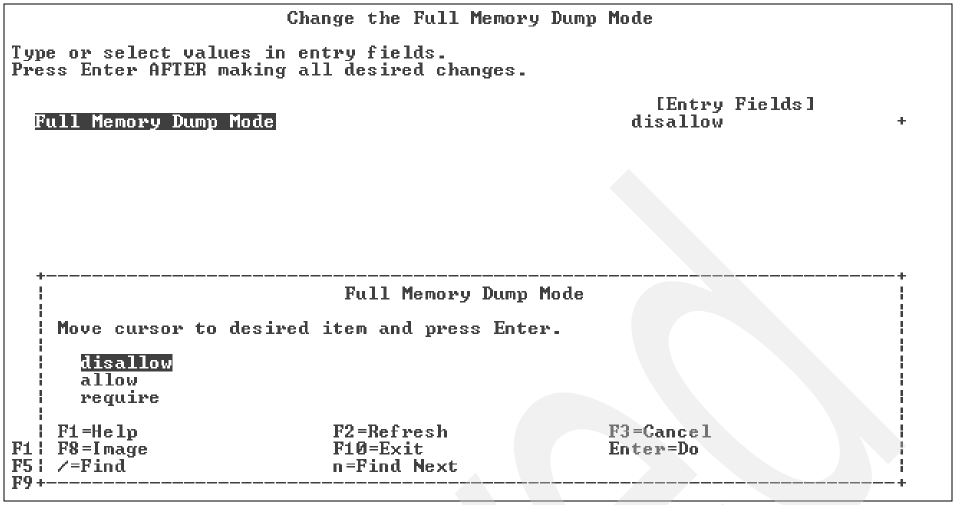


Figure 4-13 SMIT panel: Types of memory dump mode

A firmware-assisted system dump takes place under the following conditions:

- The firmware assisted dump is supported only on POWER6-based servers and later.
- The memory size at system startup is equal to or greater than 4 GB.
- The system has not been configured to do a traditional system dump.

If the firmware-assisted dump cannot be supported in spite of platform support or if the configuration of this dump facility fails, AIX forces a traditional system dump and logs in the errlog a meaningful message indicating the failure reason.

Note: As dump data is written at the next restart of the system, the AIX dump tables that are used to refer the data cannot be preserved.

The following are the main steps of a firmware-assisted system dump:

1. When all conditions for a firmware-assisted dump are validated (at system initialization), AIX reserves a dedicated memory scratch area.
2. This predefined scratch area is not released unless the system administrator explicitly configures a legacy dump configuration.
3. The predefined scratch area size is relative to the memory size and ensures AIX will be able to reboot while the firmware-assisted dump is in progress.

4. System administrators must be aware that this dedicated scratch area is not adjusted when a memory DR operation modifies the memory size. A verification can be run with the **sysdumpdev** command by system administrators in order to be notified if the firmware-assisted system dump is still supported.
5. AIX determines the memory blocks that contain dump data and notifies the dedicated hypervisor to start a firmware-assisted dump with this information.
6. The hypervisor logically powers the partition off, but preserves partition memory contents.
7. The hypervisor copies just enough memory to the predefined scratch area so that the boot process can start without overwriting any dump data.
8. The AIX boot loader reads this dedicated area and copies it onto disk using dedicated open firmware methods. The hypervisor has no authority and is unable by design to write onto disk for security reasons.
9. AIX starts to boot and in parallel copies preserved memory blocks. The preserved memory blocks are blocks that contain dump data not already copied by the AIX boot loader. As with the traditional dump, a firmware-assisted dump uses only the first copy of rootvg as the dump device; it does not support disk mirroring.
10. The dump is complete when all dump data is copied onto disk. The preserved memory then returns to AIX usage.
11. AIX waits until all the preserved memory is returned to its partition usage in order to launch any user applications.

4.4 Performing a live dump

Dump components, explained in 4.3.2, “Component dump facility” on page 152, must be known before performing a live dump. All the live dump attributes must be set with the **dumpctrl** command, as described in 4.3.3, “Live dump facility” on page 157. Then, a live dump can be performed by both of the following methods:

- Using the new SMIT sub-panel “Start a Live Dump” of the menu **smitty ldmp**. In the SMIT panel shown in Figure 4-14 on page 173, the component to be dumped is vmm.frs. The symptom string is mandatory and is any-description.
- With the new **livedumpstart** command.

  Only two arguments are required to run a live dump: the component to be dumped and the symptom string, which is mandatory but can take any value.

With the **livedumpstart** command, the notation +component dumps the data from that component and its ancestors while the notation component+ dumps the data from that component and its descendents.

The following example (and Figure 4-14) shows how to use the **livedumpstart** command to do a live dump on the VMM component (AIX Virtual Memory Manager) and all its descendents:

```
# livedumpstart -c vmm+ symptom="string is mandatory and is what you
want"
0453-142 The dump is in file
/var/adm/ras/livedump/nocomp.200710222353.00.DZ.
#
```

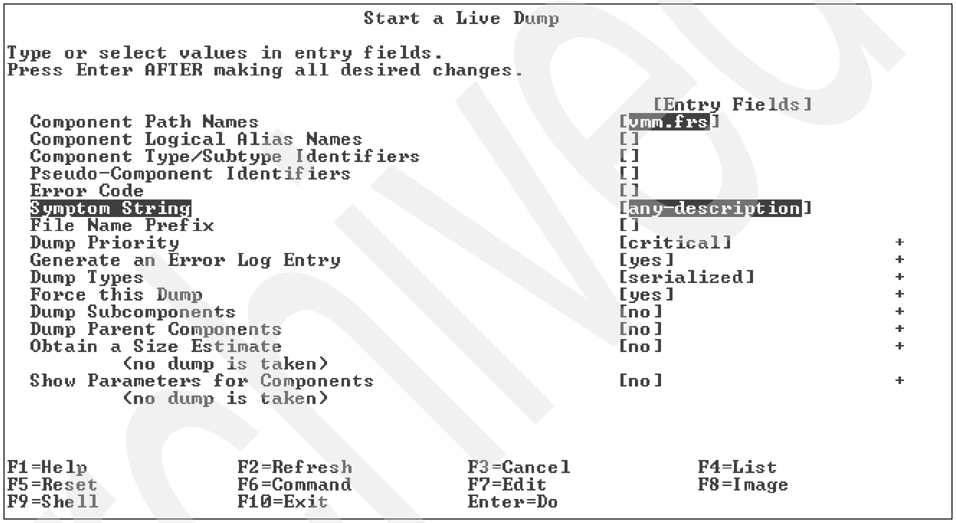


Figure 4-14 SMIT panel: Starting a live dump

As mentioned in 4.3.3, “Live dump facility” on page 157, a live dump may also be initiated from the AIX kernel or from a kernel extension. For additional information, see “Live Dump Facility” in *Kernel Extensions and Device Support Programming Concepts*, found at:

http://publib.boulder.ibm.com/infocenter/pseries/v5r3/topic/com.ibm.aix.kernelext/doc/kernextc/kernextc.pdf

4.5 Kernel error recovery

Starting with AIX V6.1, the AIX kernel has been enhanced with the ability to recover from unexpected errors. Kernel components and extensions can provide failure recovery routines to gather serviceability data, diagnose, repair, and recover from errors. In previous AIX versions, kernel errors always resulted in an unexpected system halt.

The kernel error recovery is a continuous reliability and availability feature to improve system stability. In AIX V6.1, the kernel components, such as the watchdog services, have failure recovery routines implemented.

Kernel error recovery support is not enabled by default. Kernel extensions created in AIX 5L V5.3 are still supported with both enabled and disabled kernel error recovery. Kernel extensions implementing the new failure recovery routines cannot be run on pre-AIX V6.1 versions.

4.5.1 Recovery concepts

Kernel components and kernel extensions enabled to support kernel error recovery will register their failure recovery routines to a recovery manager at runtime. These routines will typically perform the following actions:

- Collect serviceability data.
- Verify and correct data structures.
- Free or otherwise handle resources held or modified by the failing component at the time of the error (such as locks).
- Determine the error action.

The recovery manager is responsible for controlling the recovery process. The recovery manager is a collection of kernel functions to manage and run the failure recovery routines (Figure 4-15 on page 175).

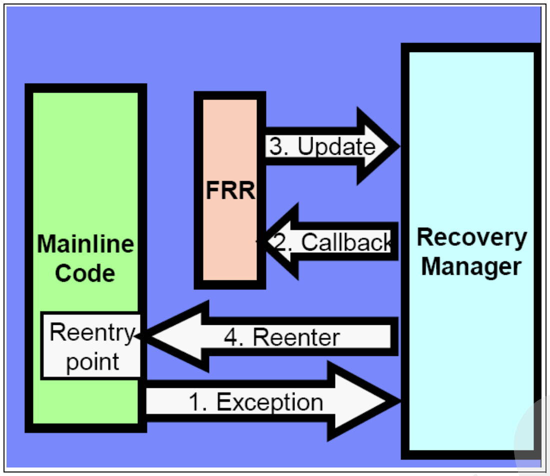


Figure 4-15 Kernel recovery process

If a failure occurs in the kernel component or extension, the exception is routed to the recovery manager. The recovery manager will then call the corresponding failure recovery routine. After completing the routine, the recovery manager will then pass control back to the failing component at the re-entry point.

Most failure recovery routines initiate a live dump before any repair has been attempted. Each kernel recovery will be logged into the AIX error log and, if applicable, with a reference to the live dump. Live dump should be enabled on your system in order to provide IBM service support with serviceability data. See 4.3.3, "Live dump facility" on page 157 for more information about live dump.

Kernel recovery may cause the system to be temporarily unavailable. All failure routines during an kernel recovery have to complete within a total time of ten seconds. The time limit is chosen to allow kernel recovery to occur within the default HACMP heartbeat timeout to prevent unwanted takeovers. If these limits are exceeded, an error will be logged in the AIX error log.

> **Note:** In some cases, complete error recovery is not possible and error isolation is executed instead. Some functions might be lost after a kernel recovery, but the operating system remains in a stable state. If necessary, restart your system to restore the lost functions.

4.5.2 Kernel error recovery management

The failure recovery routines are part of the source code within the kernel components and extensions. No management of the FRR is needed or possible. This section describes the changes to user interfaces related to kernel recovery.

AIX error log entries

AIX will log every kernel error recovery occurrence into the AIX error log. Three new error log entries are introduced with kernel error recovery (Table 4-9).

Table 4-9 Kernel error recovery error log entries

Identifier	Label	Description
00412073	RECOVERY	This error is logged each time a kernel error recovery occurs. The failing function, the FRR name, and, if applicable, the live dump file name are logged. The live dump is stored in the /var/adm/ras/livedump directory.
B709A434	RECOVERY_NOTIFY	This error log type is issued at most once a minute. With every RECOVERY_NOTIFY error log entry, a message is written to the console that a error recovery occurred.
B9657B5B	RECOVERY_TIME	This error is logged if a kernel error recovery process exceeds either the two seconds or the ten seconds timeout.

You can use the error log entries and the corresponding live dumps to provide IBM service with more information in case of a problem.

The following shows a sample RECOVERY error log entry:

```
---------------------------------------------------------------------------
LABEL:          RECOVERY
IDENTIFIER:     00412073

Date/Time:       Thu Oct 25 18:39:07 2007
Sequence Number: 521
Machine Id:      00C1F1704C00
Node Id:         lpar02
Class:           O
Type:            INFO
WPAR:            Global
Resource Name:   RMGR

Description
Kernel Recovery Action
```

```
Probable Causes
Kernel Error Detected

        Recommended Actions
        Contact IBM Service

Detail Data
Live Dump Base Name
RECOV_20071025233906_0000
Function Name
watchdog
FRR Name
watchdog_frr
Symptom String
273
EEEE00009627A058
0000000000000000
watchdog+514
sys_timer+154
clock+2E0
Recovery Log Data
0001 0000 0000 0000 F100 0415 4003 36B0 0000 0111 0000 0000 0000 0000 0016 C4F8
8000 0000 0002 9032 EEEE 0000 9627 A058 0000 0000 0000 0000 0000 0000 0000 0000
...
```

SMIT panel

A new SMIT panel is available to deactivate, activate, and show the current state of kernel error recovery. It can be accessed by selecting **Problem Determination** → **Kernel Recovery**.

The raso command tunables

With the **raso** command, you can manage reliability, availability, and serviceability parameters. The new tunables shown in Table 4-10 are introduced to change the behavior of the recovery manager. All recovery tunables are restricted and should not be changed unless requested by the IBM service support. See 6.2, "Restricted tunables" on page 249 for additional information.

Table 4-10 The raso tunables for kernel error recovery

Tunable	Description
recovery_framework	With the recovery_framework tunable, you can enable or disable the kernel error recovery. A system reboot is required for the change to take effect. The default state is disabled.

Tunable	Description
recovery_action	The recovery_action tunable allows you to temporarily disable the kernel error recovery without a system reboot. If an kernel error occurs, the system will be halted without any recovery attempts. This option only has an effect if the recovery_framework parameter is enabled. Setting the recovery_action parameter to the halt system value does not provide any performance improvement.
recovery_debugger	The recovery_debugger tunable parameter allows the kdb (kernel debugger) to be invoked when recovery actions occur. This tunable is intended for debugging only by IBM Support.
recovery_average_threshold	The recovery_average_threshold tunable manages the threshold on the maximum average recovery actions. The system stops if the average number of kernel recovery actions per minute exceeds this value. The default value is 5.

The KDB kernel debugger

The kernel debugger **kdb** command displays, with the subcommand stat, the started and currently running number of failure recovery routines:

```
# kdb
(0)> stat
SYSTEM_CONFIGURATION:
CHRP_SMP_PCI POWER_PC POWER_6 machine with 2 available CPU(s)  (64-bit
registers)

SYSTEM STATUS:
sysname... AIX
nodename.. lpar02
release... 1
version... 6
build date Sep 17 2007
build time 21:00:47
label..... 0738A_610
machine... 00C1F1704C00
nid....... C1F1704C
age of system: 3 day, 21 hr., 35 min., 20 sec.
xmalloc debug: enabled
FRRs active... 0
FRRs started.. 0
```

4.6 Concurrent update

AIX V6.1 introduces the ability to update certain kernel components and kernel extensions in place, without needing a system reboot. IBM service support can deliver interim fixes as concurrent updates. At the time of writing, interim fixes are the only supported fix type for concurrent updates. You can manage the new concurrent fixes with the **emgr** command.

Applying fixes without needing a reboot provides you with a method to fix critical problems without service interruption. As with traditional interim fixes, you can choose if a concurrent update should be made persistent across system reboots or applied only to the currently running system, which is the default behavior.

In addition, concurrent updates can be removed from the system without needing a reboot if the fix is applied to state only and no commit operation has been issued.

Performing concurrent updates on an operating system is a complex task and places stringent demands on the operating system. Interim fixes to kernel components and kernel extensions have to meet certain technical prerequisites in order to be provided as concurrent updates. Therefore, not all kernel fixes will be available as concurrent updates.

4.6.1 Concurrent update method

This section discusses the used methods to update kernel components and kernel extensions and provides you a technical introduction on how it works. Refer to 4.6.2, "The emgr command concurrent update operations" on page 181 for an explanation of how to perform concurrent updates with the **emgr** command.

The **emgr** command is used to initiate concurrent updates. It will perform the prerequisite checks on the interim fix and then execute a new system call named kpatch(). This system call controls the patching of the kernel. Several checksums are used to verify the integrity of the new fixed object files as well as the target kernel components. This procedure makes sure that only supported kernel component versions are fixed.

AIX V6.1 keeps a separate memory area where it stores all the new fixed object files. The kpatch system call will load all new object files into this memory area. At this time, the updates are not activated. In order to be able activate an interim fix in a running AIX, the system has to be paused for a short time period. The only executed task has to be the kpatch system call. All CPUs except the one running kpatch() will freeze during the update.

The kpatch system call will replace the first instruction of each function to patch with an redirection statement to the new fixed code. Before replacing takes place, the first instruction is saved in order to be able to recover functionality for a potential in place interim fix removal. After all redirection statements have been set, the CPU freezes are released and the AIX kernel uses the new fixed code for any execution.

Figure 4-16 shows a high level view of the task performed for a concurrent update:

1. The **emgr** command calls the kpatch() routine.
2. kpatch() loads the fixed objects to the patch area.
3. kpatch() initiates the CPU freezes.
4. kpatch() saves the first instruction of a function and then replaces it with a redirection to the new fixed code resident in the patch area.
5. kpatch() initiates an unfreeze of all CPUs.
6. kpatch() reports the status of the concurrent update back to the **emgr** command.

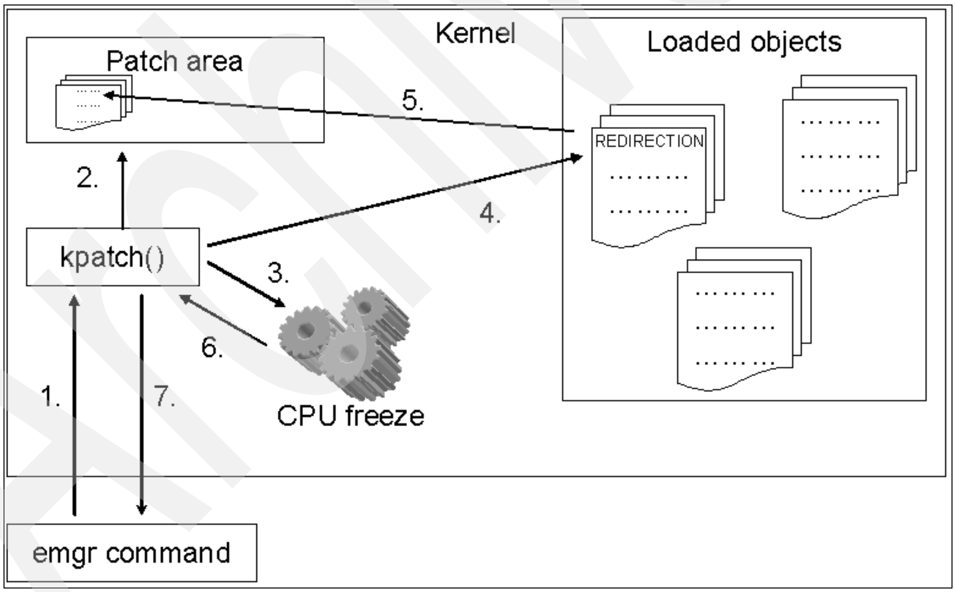


Figure 4-16 Concurrent in memory update high level overview

4.6.2 The emgr command concurrent update operations

The **emgr** command is used to apply, remove, verify, and list concurrent kernel updates. We recommend using the preview mode with the -p flag before performing the actual apply or remove operation.

The following examples show operations on a kernel extension named sample_kext and an interim fix labeled CU_Demo, since at the time of writing no interim fixes are available from the service support.

Applying concurrent updates

To apply an concurrent update interim fix, use the new -i flag with the **emgr** command. The output will look similar to standard interim fixes, except that the file type in the file section will indicate that it is a concurrent update:

```
# emgr -i CU_Demo.071015.epkg.Z
...
File Number:      1
   LOCATION:      /usr/lib/drivers/sample_kext
   FILE TYPE:     Concurrent Update
   INSTALLER:     installp (new)
   SIZE:          8
   ACL:           root:system:755
   CKSUM:         17531
   PACKAGE:       None or Unknown
   MOUNT INST:    no
...
```

In order to make the update persistent across reboots, add the -C flag:

```
# emgr -Ci CU_Demo.071015.epkg.Z
```

The commit operation can be also issued separately after the application of a fix:

```
# emgr -C -L CU_Demo
```

Note that committed updates cannot be removed without a reboot.

Removing concurrent updates

Use the -r flag to remove a concurrent update. Note that you have to use the -L flag and specify the fix label as an argument:

```
# emgr -r -L CU_Demo
```

In some situations the updated objects cannot be unloaded. You can use the -R flag in such a situation and the fix will be removed from the database, but the system will need a reboot in order to unload the concurrent code:

```
# emgr -R -L CU_Demo
```

List concurrent updates

Use the -l flag to list information about the installed concurrent updates:

```
# emgr -l -L CU_Demo
ID  STATE LABEL      INSTALL TIME      ABSTRACT
=== ===== ========== ================= ==============================
1    P    CU_Demo    10/24/07 19:11:14  Demonstration CU ifix

STATE codes:
 S = STABLE
 M = MOUNTED
 U = UNMOUNTED
 Q = REBOOT REQUIRED
 B = BROKEN
 I = INSTALLING
 R = REMOVING
 T = TESTED
 P = PATCHED
 N = NOT PATCHED
 SP = STABLE + PATCHED
 SN = STABLE + NOT PATCHED
 QP = BOOT IMAGE MODIFIED + PATCHED
 QN = BOOT IMAGE MODIFIED + NOT PATCHED
 RQ = REMOVING + REBOOT REQUIRED
```

Note the new states introduced with the concurrent updates at the end of the state code list (see Table 4-11).

Table 4-11 New interim fix states displayed with the emgr command

State	Description
P	The concurrent update is applied.
N	The concurrent update is installed but not activated. This state is displayed after a reboot if the fix has not been made persistent (stable). To recover from this state, you have to remove the fix and apply it again.
SP	The concurrent update is applied and made persistent (stable).
SN	The concurrent update has been made persistent but is currently not applied.

State	Description
QP	The concurrent update is applied and the boot image has been modified.
QN	The concurrent update is not applied but the boot image is updated.
RQ	The concurrent update is marked to be removed, but a reboot is still required to remove it from the memory.

Verify concurrent updates

You can use the verify option on concurrent updates in the same manner as on standard interim fixes:

```
# emgr -c -L CU_Demo
```

Note that verification is only intended for persistent concurrent updates. Verification on applied concurrent updates will always fail.

4.7 Core dump enhancements

A core file is created in the current working directory when various errors occur. Errors such as memory-address violations, illegal instructions, bus errors, and user-generated quit signals, commonly cause this core dump.

Previous AIX releases only dump core files if either the real user ID is root or the effective user ID (EUID) and effective group (EGID) match the real user ID (RUID) and the real group ID (RGID). All core dumps are created with an access mode of 0666 in octal notation. (0666 grants read and write access rights to the owner, the owners group, and to others. No execution is permitted.)

AIX V6.1 changes the user ID (UID) / group ID (GID) authorization requirements for core dump file creation to provide the dump capability for SUID and SGID processes. Figure 4-17shows the UID / GID authorization dependencies that govern the core dump process.

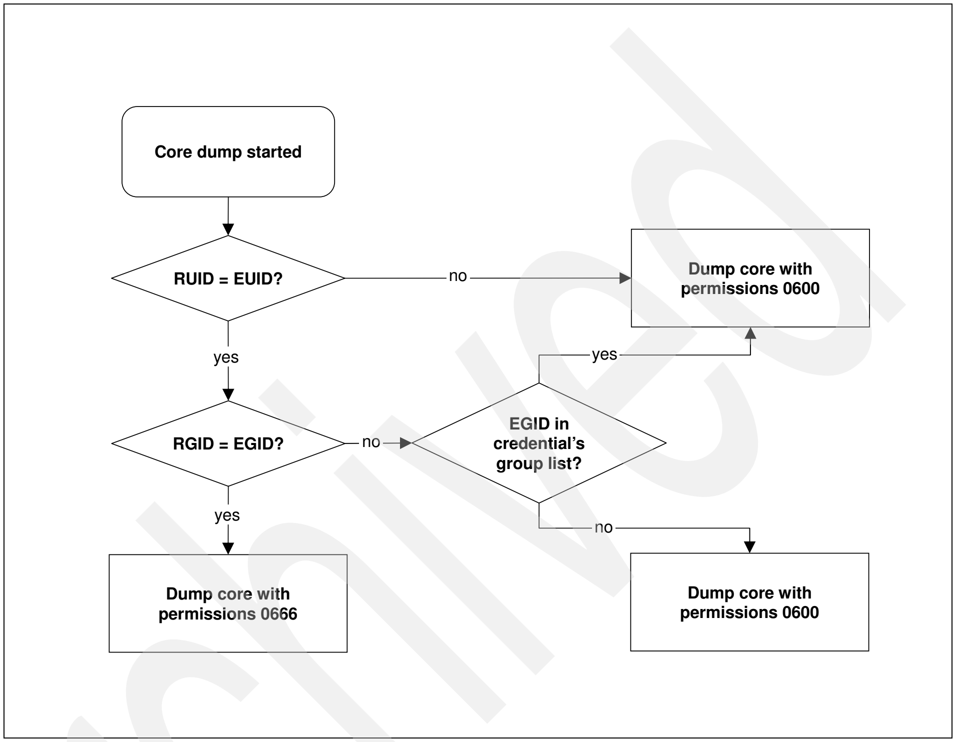


Figure 4-17 Core dump UID / GID dependencies

The enhanced AIX V6.1 core dump authorization framework is designed to balance serviceability with security concerns. The section "core File Format", in Chapter 2. "File formats", in *AIX Version 6.1 Files Reference*, SC23-5249 explains the details of the core dump process as follows:

- All dumped cores are in the context of the running process. They are dumped with an owner and a group matching the effective user ID (UID) and group ID (GID) of the process. If this UID/GID pair does not have permission to write to the target directory that is determined according to the standard core path procedures, no core file is dumped.
- If the real user ID (RUID) is root, the core file can always be dumped, but with a mode of 0600.

- If the effective user ID (EUID) matches the real user ID (RUID), and the effective group ID (EGID) matches any group in the credential's group list, the core file is dumped with permissions of 0600.
- If the EUID matches the RUID, but the EGID does not match any group in the credential's group list, the core file cannot be dumped. The effective user cannot see data that the user does not have access to.
- If the EUID does not match the RUID, the core file can be dumped only if you have set a core directory using the **syscorepath** command. This avoids dumping the core file into either the current working directory or a user-specific core directory in such a way that you cannot remove the core file. The core is dumped with a mode of 0600. If you have not used the **syscorepath** command to set a core directory, no core is dumped.

4.8 Trace hook range expansion

Over the past few years of AIX development history, the trace ability received considerable enhancements and matured to become a very important and strategic key feature not only for performance evaluation but for First Failure Data Capture (FFDC) functionality. Traditional tracing works mainly by placing hooks containing relevant data into a buffer when a system component deems that an important event should be recorded. Each component defines their traceable events. In order to define their events, each component is given a range of trace hook IDs. Each trace hook allocates 12 bits for the trace hook ID, allowing for 4096 unique trace hook IDs.

In anticipation of future demands, AIX V6.1 expands the trace hook ID range from 12 bits to 16 bits.

The implemented trace hook range expansion will allow for significantly more trace hook IDs to be used. The trace hook ID range will include the old range, 0x000 to 0xFFF, and the new range, 0x1000 to 0xFFFF. 16-bit IDs in which the least significant nibble is 0 (such as 0x2100, 0xABC0, or 0xFFF0) will be equivalent to their 12-bit IDs to avoid collision. For example, hook ID 0x218 will be represented as 0x2180 in a 64-bit kernel. Thus, to avoid collision, 0x2180 and 0x218 will be regarded as the same ID.

Another restriction applies to hooks below 0x1000. These hooks must have zero as their least significant nibble. For example, 0x0101 and 0x00A1 are not allowed since hook ID 0x1010 and 0x0A10 map to these hook IDs respectively.

The new trace hook scheme allows for a total of 65536 - 7680 = 57856 unique hook IDs for 64-bit applications.

The expanded hook range will only be accessible to 64-bit applications since 32-bit applications do not have room in the hook word for the extra four bits necessary. Thus, 32-bit applications will be restricted to using only the existing 12-bit range that has not already been allocated. Currently, there are around 2000 un-allocated 12-bit hook IDs that provide an ample resource for new trace hook ID assignment. Some of these, however, will be potentially used for the remaining 32-bit kernel development in AIX 5L.

In support of the expanded trace hook range in AIX V6.1, the command lines of the following four commands were updated: **trace**, **trcupdate**, **trcevgrp**, and **trcrpt**:

1. The **trace -j** and the **trace -k** command options accept four-hex-digit hook IDs. The -j and -k options work as on previous AIX releases with regards to two- and three-hex-digit hook IDs. For example, -j 12, prior to AIX V6.1, traced hooks 120-12F, and will trace hooks 1200, 1210, 1220,... 12F0, but not any intervening hooks, such as 1201 under AIX V6.1. The same applies to specifying three-hex-digit hooks. For example, -k 130 ignores only hook 1300, and does not apply to a hook range in AIX V6.1. The recommended way to group trace hooks is with the trace groups (see the **trcevgrp** command).
2. The **trcupdate -x** command option is aware of up to four hex digits per hook ID in the hook ID list.
3. The **trcevgrp -h** command option is aware of up to four hex digits per hook ID in the hook ID list.
4. The **trcrpt -d** and **trcrpt -k** command options accept four-hex-digit hook IDs. Beginning with AIX V6.1 and in similar way to the previously described **trace** command, specifying a two-hex-digit hook ID in the *hh* form results in *hh00*, *hh10*,...,*hhF0*. Specifying a three-hex-digit hook ID in the *hhh* form results in *hhh0*. Specifying a four-hex-digit hook ID in the *hhhh* form results in *hhhh*. Four-hex-digit hook IDs can always be displayed. However, if a four-hex-digit hook ID has a trailing digit of zero, the zero is removed to display only three hex digits. This occurs because four-hex-digit hook IDs in the form *hhh0* are equivalent to three-hex-digit hook IDs in the form *hhh*. The **trcrpt -D** and **trcrpt -K** command options are aware of up to four hex digits per hook ID in the event group list.

In addition to the enhanced command-line interface, the **trcgui** graphical user interface and the trace related SMIT panels are aware of four-hex-digit trace hook IDs too.

4.9 LVM configuration and trace logs

AIX releases prior to AIX V6.1 are equipped with internal Logical Volume Manager (LVM) logging services that are aimed at assisting IBM support specialists in error detection and problem analysis. These services collect information that is related to configuration and management operations on enhanced concurrent and non-concurrent volume groups. But the increase of complexity over the recent past has raised additional requirements for LVM configuration and tracing logs to facilitate continuous surveillance and long-term data recording.

AIX V6.1 provides enhancements to LVM configuration and tracing logs in three areas:

- Introduction of the new lvmcfg logging service to keep a long-term record of changes to volume groups over time.
- Enhanced lvmt trace log functionality for continuous and detailed LVM configuration change recording.
- Enhanced trace log characteristics for the Group Services Concurrent Logical Volume Manager (gsclvmd) daemon.

All implemented additions and changes utilize the AIX alog facility and focus on the LVMs First Failure Data Capture (FFDC) capabilities to improve the ability to quickly determine recreate scenarios and to minimize problem resolution time.

> **Note:** All configuration and trace log facilities described in this section are not intended for customer use but to collect important information for IBM Support specialists to assist in problem recreation, analysis, and resolution. To that extent, this section provides only background information to advanced system administrators regarding the ongoing effort to enhance the continuous availability features of AIX.

4.9.1 LVM configuration log

Recent field studies have identified the need for the LVM to keep a long-term log of changes to volume groups over time. To meet this requirement, AIX V6.1 provides a new log, called the *lvmcfg log*, which keeps track of what LVM commands were run, what arguments were passed to those commands, and what exit value those commands returned. This logging service will always be enabled on any AIX V6.1 system. To record the information in the log, each high level AIX V6.1 LVM command (excluding the commands that provide only information listings) was enhanced to call two new functions, one at the beginning of execution and the other before it exits or returns from the main code segment. If the high level command is implemented as script, the wrapper commands for the relevant functions are called. The beginning function call will open both the lvmcfg log file (lvmcfg.log) and the lvmt log file (lvmt.log).

The lvmt log is described in more detail in 4.9.2, “LVM detailed trace configuration log” on page 189.

The function call will then add a start entry to both logs with the name of the command being run and the arguments that were passed to that command. The ending function call adds an end entry to both logs with the name of the command and the exit code that the command is exiting with. It also closes each open log and performs any other necessary cleanup.

In order to view the lvmcfg log, the **alog** command can be used. The following example shows the **alog** command output for the lvmcfg log on an AIX V6.1 system:

```
# alog -t lvmcfg -o | pg
Starting Log
[S 135196 155670 09/27/07-15:46:32:215 extendlv.sh 794] extendlv hd2 12
[E 135196 0:860 extendlv.sh 33] extendlv: exited with rc=0
[S 126986 192630 09/27/07-17:03:07:756 extendlv.sh 794] extendlv hd6 3
[E 126986 0:673 extendlv.sh 33] extendlv: exited with rc=0
[S 82038 110648 09/27/07-12:03:53:362 chlv.sh 527] chlv -L primary_bootlv hd5
[E 82038 0:404 chlv.sh 23] chlv: exited with rc=0
[S 82042 110648 09/27/07-12:03:53:782 syncvg.sh 539] /usr/sbin/syncvg -v rootvg
[E 82042 0:325 syncvg.sh 19] /usr/sbin/syncvg: exited with rc=0
[S 180370 151658 09/27/07-12:04:01:718 cfgvg.c 106] cfgvg
[E 180370 0:001 cfgvg.c 212] cfgvg: exited with rc=0
[S 327750 393228 09/27/07-16:16:14:043 extendlv.sh 794] extendlv hd2 1
[E 327750 0:815 extendlv.sh 33] extendlv: exited with rc=0
[S 110692 82020 10/01/07-10:53:32:121 chlv.sh 527] chlv -L primary_bootlv hd5
[E 110692 0:364 chlv.sh 23] chlv: exited with rc=0
[S 110696 82020 10/01/07-10:53:32:502 syncvg.sh 539] /usr/sbin/syncvg -v rootvg
[E 110696 0:417 syncvg.sh 19] /usr/sbin/syncvg: exited with rc=0
[S 204906 200802 10/01/07-10:54:00:231 cfgvg.c 106] cfgvg
[E 204906 0:001 cfgvg.c 212] cfgvg: exited with rc=0
[S 336098 241678 10/01/07-14:27:50:344 mklv.sh 617] mklv -t jfs2 rootvg 1
[E 336098 0:809 mklv.sh 72] mklv: exited with rc=0
[S 336102 241678 10/01/07-14:27:51:329 chlv.sh 527] chlv -L /wpars/mywpar1 fslv00
[E 336102 0:375 chlv.sh 23] chlv: exited with rc=0

... omitted lines ...
```

Each entry into both the lvmcfg log and the lvmt log is comprised of a preamble and a message. The preamble for start entries contain the following information:

S Marks entry as a start entry

pid Process ID

ppid Parent process ID

time stamp Date and time the entry was recorded
Format: MM/dd/yy-hh:mm:SS:sss
(MM=month, dd=day, yy=year, hh=hour, mm=minute, SS=sec, sss=millisec)

filename Name of command (executable, shell script)

line number Line number in reference to code executed

The start entry preamble has the following format:
[S pid ppid date time stamp filename line number]

The preamble for end entries contain the following information:

E Marks entry as an end entry

pid Process ID

time stamp Date and time since start entry was recorded
Format: MM/dd/yy-hh:mm:SS:sss
(MM=month, dd=day, yy=year, hh=hour, mm=minute, SS=sec, sss=millisec)

filename Name of command (executable, shell script)

line number Line number in reference to code executed

The end entry preamble has the following format:
[E pid time stamp filename line number]

The lvmcfg log file, lvmcfg.log, adheres to the `alog` file format and is stored in the /var/adm/ras directory. The default size of the lvmcfg log is defined to be 50 KB. As required by the `alog` facility, the minimum log size for the log is 4 KB, but no implementation specific restrictions on the maximum log size exist. The log entries are wrapped within the log file.

4.9.2 LVM detailed trace configuration log

The same considerations that initiated the addition of the new lvmcfg logging service also guide the enhancements to the existing LVM trace logging facility called *lvmt*. In previous AIX releases, this facility writes to files in the /tmp directory to store the trace data. This tracing service is disabled by default and must be turned on using an environment variable before any trace data is recorded. The trace files are simply appended to and can consume all of the space in /tmp if tracing is left on for an extended period of time. This trace facility is an “all or nothing” type of service. When turned on, there is no way to control the amount of information traced.

In addition to the new lvmcfg log, the AIX V6.1 LVM subsystem utilizes an enhanced lvmt logging service to improve the continuous availability signature of the operating system. The lvmt logging service provides a trace facility for the LVM commands similar to what the light weight memory trace provides to the AIX kernel. The enhanced lvmt logging service will always be enabled on any AIX V6.1 system. As outlined in 4.9.1, “LVM configuration log” on page 187, all start and end entries that are recorded in the lvmcfg log are also written to the lvmt log. But if required, AIX V6.1 high level LVM commands write additional entries to the lvmt log with a call to the (undocumented) lvmt() function. Each call to the lvmt() function will include a verbosity level. The lvmt() function will add the entry to the lvmt log if the verbosity level included is less than or equal to the verbosity set for the system.

The verbosity level has a value of 0-9. The verbosity level that the running command uses will be determined when the log is first open. The verbosity level to use will be determined in a sequence of steps. First, if the environment variable LVMT_VERBOSE is set to a numeric value between 0 and 9, that value will be used as the verbosity. Second, if the verbosity is not set by the LVMT_VERBOSE environment variable, then the file /etc/lvmtlog.cfg will be read for a line starting with LVMT_VERBOSE= followed by a number between 0 and 9. If that file exists and contains the line, then that value will be used for verbosity. Third, if the verbosity is not found in the environment variable or the file, then it will default to a verbosity of 3. Any logs entries with a verbosity level at or below the set verbosity level will be entered into the lvmt log. Setting the verbosity level to 0 will turn off logging. Macros for four levels of verbosity are predefined for use by the LVM:

LVMT_ERROR This level has a verbosity of 1 and is used to add error conditions found in the code.

LVMT_WARN This level has a verbosity of 2 and is used to add warnings into the lvmt log.

LVMT_INFO This level has a verbosity of 3 and can be assigned to basic information about the execution of the code.

LVMT_DETAIL This level has a verbosity of 7 and is used to add detailed information about the code execution and program flow.

In order to view the lvmt log, the **alog** command can be used. The following example shows the **alog** command output for the lvmt log on an AIX V6.1 system:

```
# alog -t lvmt -o | pg
Starting Log
[S 82038 110648 09/27/07-12:03:53:362 chlv.sh 527] chlv -L primary_bootlv hd5
[E 82038 0:404 chlv.sh 23] chlv: exited with rc=0
[S 82042 110648 09/27/07-12:03:53:782 syncvg.sh 539] /usr/sbin/syncvg -v rootvg
[E 82042 0:325 syncvg.sh 19] /usr/sbin/syncvg: exited with rc=0
```

```
[S 180370 151658 09/27/07-12:04:01:718 cfgvg.c 106] cfgvg
[E 180370 0:001 cfgvg.c 212] cfgvg: exited with rc=0
[S 245898 1 09/27/07-12:05:18:224 utilities.c] Command started without lvmcfg_start call
[1 245898 0:000 utilities.c 1330] lvm_getvgdef: FAIL: no matching vgids,
returning=LVM_OFFLINE
[1 245898 0:000 queryutl.c 1339] lvm_lsvg: FAIL: lvm_getvgdef failed, rc=-100
[S 200822 1 09/27/07-12:05:18:743 utilities.c] Command started without lvmcfg_start call
[1 200822 0:000 utilities.c 1330] lvm_getvgdef: FAIL: no matching vgids,
returning=LVM_OFFLINE
[1 200822 0:000 queryutl.c 1339] lvm_lsvg: FAIL: lvm_getvgdef failed, rc=-100
[S 327750 393228 09/27/07-16:16:14:043 extendlv.sh 794] extendlv hd2 1
[E 327750 0:815 extendlv.sh 33] extendlv: exited with rc=0
[S 110692 82020 10/01/07-10:53:32:121 chlv.sh 527] chlv -L primary_bootlv hd5
[E 110692 0:364 chlv.sh 23] chlv: exited with rc=0
[S 110696 82020 10/01/07-10:53:32:502 syncvg.sh 539] /usr/sbin/syncvg -v rootvg
[E 110696 0:417 syncvg.sh 19] /usr/sbin/syncvg: exited with rc=0
[S 204906 200802 10/01/07-10:54:00:231 cfgvg.c 106] cfgvg
[E 204906 0:001 cfgvg.c 212] cfgvg: exited with rc=0
[S 336098 241678 10/01/07-14:27:50:344 mklv.sh 617] mklv -t jfs2 rootvg 1
[E 336098 0:809 mklv.sh 72] mklv: exited with rc=0
[S 336102 241678 10/01/07-14:27:51:329 chlv.sh 527] chlv -L /wpars/mywpar1 fslv00
[E 336102 0:375 chlv.sh 23] chlv: exited with rc=0

... omitted lines ...
```

As mentioned previously, each entry into the lvmt log will contain a preamble and a message. The preamble for the start and end entries contain the same information as the related log entries in the lvmcfg log. (Refer to 4.9.1, “LVM configuration log” on page 187 for a detailed description of start and end entry preambles.)

The preamble for lvmt log entries that are added through the lvmt() function call (and as such are not start or end entries) contain the following information:

Verbosity level Level of verbosity

pid Process ID

time stamp Date and time since start entry was recorded
Format: MM/dd/yy-hh:mm:SS:sss
(MM=month, dd=day, yy=year, hh=hour, mm=minute, SS=sec, sss=millisec)

filename Name of command (executable, shell script)

line number Line number in reference to code executed

The lvmt entry preamble has the following format:

[verbosity pid time stamp filename line number]

The lvmt log file, lvmt.log, adheres to the **`alog`** file format and is stored in the /tmp directory. The default size of the lvmcfg log is defined to be 200 KB. As required by the **`alog`** facility, the minimum log size for the log is 4 KB, but no implementation specific restrictions on the maximum log size exists. The log entries are wrapped within the log file.

4.9.3 The gsclvmd daemon log

The Group Services Concurrent Logical Volume Manager daemon (gsclvmd) is needed in HACMP environments to manage enhanced concurrent volume groups that are accessed by HACMP cluster nodes. Because of the complexity of the Group Services Concurrent Logical Volume Manager part of the LVM, the related gsclvmd daemon traditionally has its own logging facility, called *gsclvmd trace*. In previous AIX releases, gsclvmd trace writes to several plain text files, namely the parent gsclvmd process logs to /tmp/gsclvmd.log and the child processes log to /tmp/ch.log.<vgid>. (<vgid> is a place holder for the relevant volume group ID.) The log files are not bound, so there is the potential of filling up the /tmp directory if tracing is left on un-monitored for extended periods of time. Because of that possibility, gsclvmd trace used to be disabled by default and cannot be turned on until a volume group is varied on.

AIX V6.1 enhances the gsclvmd trace to use the **`alog`** facility and to write to only one *lvmgs log* file (lvmgs.log). The enhanced gsclvmd trace service will always be enabled on any AIX V6.1 system to support the AIX First Failure Data Capture framework.

The lvmgs log is written by using the same function as in previous AIX releases. However, a new argument has been added to this function to account for 10 distinct verbosity levels. The lvmgs log has the same verbosity levels as the lvmt log, but the predefined macros that are used by the LVM are named slightly different:

NOTIFY_ERROR	Verbosity level 1
NOTIFY_WARN	Verbosity level 2
NOTIFY_INFO	Verbosity level 3
NOTIFY_DETAIL	Verbosity level 7

The verbosity that will be used can be set in a similar way as the lvmt verbosity. First, the environment variable LVMGS_VERBOSE will be checked to see if it has a value between 0 and 9. If it does, that will be used for the gsclvmd verbosity. Next, the /etc/lvmtlog.cfg file will be checked for a line starting with LVMGS_VERBOSE= followed by a number between 0 and 9. If either of those values are not set, a default verbosity of 3 will be used. Setting the verbosity to 0 will effectively turn off logging. It should be noted that in customer environments gsclvmd is started through HACMP. Therefore, the gsclvmd process will not inherit the environment variables from the calling shell. In this case, the LVMGS_VERBOSE variable will not be seen by the gsclvmd process.

The gsclvmd trace log can be viewed using the **alog** command, just like with the lvmcfg and lvmt logs. The following example shows the **alog** command output for the gsclvmd log on an AIX V6.1 system:

```
# alog -t lvmgs -o
Starting Log
[3 122978 1 11/02/07-16:48:56:959 gsclvmd.c 259] main: new parent process
[3 122978 1 11/02/07-16:49:42:275 gsclvmd.c 660] do_daemon_loop: got a new message from
src
[3 122978 515 11/02/07-16:49:42:275 gsclvmd.c 1074] handle_src_req: request from
tellclvmd, action=21[STARTVG]
[3 122978 515 11/02/07-16:49:42:276 gsclvmd.c 1177] src_startvg:
newvgid=0027b16d00004c000000011602572d55
[3 221426 1 11/02/07-16:49:42:281 gsclvmd.c 252] main: new child process
[3 221426 1 11/02/07-16:49:42:282 gsclvmd.c 1391] child_startup:
vgid=0027b16d00004c000000011602572d55,
socket=/dev/ch.sock.0027b16d00004c000000011602572d55
[3 221426 1 11/02/07-16:49:42:282 gschild.c 284] clvm_init_ha_gs: (ENTER)
[3 221426 1 11/02/07-16:49:42:282 gschild.c 285] clvm_init_ha_gs:
vgid=0027b16d00004c000000011602572d55, varyon=0
[3 221426 1 11/02/07-16:49:42:282 gsutility.c 636] get_vg_major: Major num=29
[3 221426 258 11/02/07-16:49:42:308 gschild.c 1086] read_config_request: (ENTER)
[3 221426 515 11/02/07-16:49:42:309 gschild.c 1264] read_vgsa_request: (ENTER)
[3 221426 772 11/02/07-16:49:42:309 gschild.c 772] join_vg_groups: (ENTER): grp_index=0
[3 221426 1029 11/02/07-16:49:42:309 gschild.c 977] wait_for_requests: (ENTER)
[3 221426 772 11/02/07-16:49:42:309 gschild.c 878] join_vg_groups: ha_gs_join returned
successfully for vgda group, tid=0
[3 221426 772 11/02/07-16:49:42:309 gschild.c 945] join_vg_groups: (LEAVE): grp=0,
ncollides=0

... omitted lines ...
```

The preamble for gsclvmd log entries contains the following information:

Verbosity level	Level of verbosity of entry
pid	Process ID
tid	Thread ID
date	Date the entry was recorded
time stamp	Date and time since start entry was recorded Format: MM/dd/yy-hh:mm:SS:sss (MM=month, dd=day, yy=year, hh=hour, mm=minute, SS=sec, sss=millisec)
filename	Name of command (executable, shell script)
line number	Line number in reference to code executed

The lvmgs log entry preamble has the following format:

[Verbosity pid tid time stamp filename line number]

The lvmgs log file, lvmgs.log, adheres to the **alog** file format and is stored in the /tmp directory. The default size of the lvmgs log is 200 KB. As required by the **alog** facility, the minimum log size for the log is 4 KB, but no implementation specific restrictions on the maximum log size exist. The log entries are wrapped within the log file.

4.10 Group Services Concurrent LVM enhancements

The Group Services Concurrent Logical Volume Manager daemon, *gsclvmd*, is needed in HACMP environments to manage enhanced concurrent volume groups. In such HACMP configurations, the LVM device driver of each cluster node logs errors for missing disks in an enhanced concurrent volume group and also logs quorum losses.

Beginning with AIX V6.1, additional error conditions that are specific to the gsclvmd daemon but not visible to the LVM device driver layer are captured by the AIX error log facility. If the error situation indicates that the LVM device driver state on the local node cannot stay in synchronization with the device driver state on the remote cluster nodes, the relevant volume groups will be varied off.

The following error conditions are handled by either logging an error, varying-off the volume group, or both:

- Expulsion of remote nodes
- Expulsion from the Volume Group Status Area (VGSA) / Volume Group Descriptor Area (VGDA) group
- Voting after the time limit expired for a configuration change or status change
- Failed configuration change
- Loss of connectivity with group services
- Termination of the gsclvmd child daemon process
- Start gsclvmd when group services is not running
- Start gsclvmd without the proper environment variables

Seven new error log labels were implemented to support the enhanced error logging services for the gsclvmd daemon, as shown in Table 4-12 on page 195.

Table 4-12 gsclvmd error labels

Error label	Description	Probable cause
LVM_GS_RLEAVE	Remote node concurrent volume group failure detected	Remote node concurrent volume group forced offline
LVM_GS_LLEAVE	Local node concurrent volume group failure detected	Concurrent volume group forced offline
LVM_GS_CFGTIME	Vote time limit expired	Excessive load on the local node
LVM_GS_CFGFAIL	Concurrent volume group configuration change failed	Lost communication with remote nodes or attempted invalid volume group configuration change
LVM_GS_CONNECTIVITY	Group services detected a failure	Unable to establish communication with cluster daemons or concurrent volume group forced offline
LVM_GS_CHILDGONE	Concurrent LVM daemon forced volume group offline	Unrecoverable event detected by concurrent LVM daemon, lost communication with remote nodes, and lost quorum
LVM_GS_NOENV	Unable to start gsclvmd	Unable to establish communication with cluster daemons

The following list describes which of the labels in Table 4-12 will be used for each listed error condition, and whether or not the volume group needs to be forced offline:

Expulsion of remote node

In this case, the volume group will remain online on the local node. The remote node that has been removed from the group is not capable of accessing data in the volume group or of writing any data to the volume group. Configuration changes and I/O operations can continue on the local node and any operational remote nodes. The local node and any remote nodes that remain in the group will each log LVM_GS_RLEAVE to indicate that a remote node has left the group. Meanwhile, the remote node that left the group will log LVM_GS_LLEAVE if the situation permits it. If the remote node actually crashed or failed in such a way that the gsclvmd never got a chance to run, then there will be error report entries describing the system outage rather than the volume groups outage.

Expulsion from the VGSA/VGDA group

In this case, the volume group will be forced offline on the local node. Because communication with the remote nodes is no longer possible, remote nodes could change the partition mapping for the volume group without the local nodes' knowledge. This means all reads and writes must be stopped on the local node, since there is no guarantee that a partition the local node is reading from or writing to has not been moved to a different location or, even worse, replaced by a partition from a different location. The local node will log LVM_GS_LLEAVE and each remote node will log LVM_GS_RLEAVE.

Voting after time limit expired on configuration change or status change

In this case, the volume group will not be forced offline on the local node or any remote nodes. The local node will log LVM_GS_CFGTIME as an informational message. The remote nodes will not log anything, since they are not visible to the attempted vote that was not counted (the remote nodes will see the default vote instead of the vote that was attempted).

Failed configuration change

The gsclvmd daemon error handling will not force the volume group offline on the local node or any remote nodes. All nodes will log LVM_GS_CFGFAIL as an informational message. The caller distributing configuration change commands to remote nodes is responsible to do whatever back out is necessary to ensure a consistent state after a failure occurred. Only the caller, but not the gsclvmd child process, knows what steps need to be taken before the volume group is forced offline.

Loss of connectivity with group services

In this case, the volume group will be forced offline on all nodes and all nodes will log LVM_GS_CONNECTIVITY. Without running group services, partitions cannot be marked stale, since there is no access to vote and obtain the concurrent VGSA lock to perform that operation.

Termination of the gsclvmd child daemon process
This condition will be considered a loss of quorum and the relevant volume group will be forced offline. All nodes will log LVM_GS_CHILDGONE.

Start gsclvmd when group services is not running
In this case, no volume groups will be forced offline. The local node will log LVM_GS_NOENV.

Start gsclvmd without the proper environment variables set
In this case, no volume groups will be forced offline. The local node will log LVM_GS_NOENV.

4.11 Paging space verification

The root cause analysis of problems that are related to data corruption can be very difficult, because the symptoms exhibited are likely to be totally unrelated to the code segment that induced the data corruption.

AIX V6.1 provides the new paging space verification feature to improve the first failure data capture (FFDC) capability in respect to paging space data corruption problems. Paging space verification ensures that the data read in from paging space matches the data that was written out. When a page is paged out, a checksum will be computed on the data in the page and saved in a pinned array associated with the paging device. If and when it is paged back in, a new checksum will be computed on the data that is read in from the paging space and compared to the value in the array. If they do not match, the kernel will log an error and halt if the error occurred in system memory, or send an exception to the application if it occurred in user memory.

If the paging space verification feature is enabled, the checksums are stored in dedicated 256 MB segments, one per paging device. Each segment contains an array of checksums, with one checksum for each 4 KB disk block on the corresponding device. The space for this array is allocated and pinned at swapon time. The handle for a device’s segment along with other checksum data will be stored in a pgdev_chksum_data structure:

```
struct pgdev_chksum_data
{
char        pcd_chksum_size; /* bits in chksum, 0 == disabled*/
char        pcd_pad[7];      /* pad */
vmhandle_t  pcd_vmh;         /* handle of chksum segment */
long        pcd_nblocks;     /* # of alloc’d chksums */
};
```

A pinned array of these structures with a length equal to the maximum number of paging devices will be defined in the kernel. The memory of the array will be initialized to zero at boot time. The fact that the checksums for a paging device must all fit in a single 256 MB segment with one checksum per paging space block puts an upper limit on the maximum supportable paging space size.

Table 4-13 Maximum paging space size

Checksum size	**Checksums in 256 MB segment**	**Maximum paging space size**
8-bit	2^{28}	2^{40} bytes (1 TB)
16-bit	2^{27}	2^{39} bytes (512 GB)
32-bit	2^{26}	2^{38} bytes (256 GB)

All of the listed sizes are larger than the 64 GB per device maximum paging space size limit in AIX V6.1. Checksums of larger than 32 bits are unnecessary, since the maximum checksum value for a single 4 KB block is $2^{12} * 2^8 = 2^{20}$, and therefore easily fits within 32 bits.

The /etc/swapspaces file format supports two new optional fields per stanza to store attribute values related to the paging space verification feature:

auto The value of this attribute indicates whether the device should be swapped on automatically at boot. Only two values are allowed: yes or no.

checksum_size The value of this attribute determines the size in bits of the checksums for the device. Four values can be specified: 0, 8, 16, 32.

If the auto field is not present in a stanza, it will default to yes; if the checksum_size field is not present, it will default to 0. If no stanza is present for a paging device, it will default to an auto field value of no and a checksum_size of 0. This maintains compatibility with existing /etc/swapspaces files. The following listing shows the content of the /etc/swapspaces file on a system that has the additional paging device (paging00) configured to use paging space verification with a checksum size of 16-bit:

```
# cat /etc/swapspaces
* /etc/swapspaces
*
* This file lists all the paging spaces that are automatically put into
* service on each system restart (the 'swapon -a' command executed from
* /etc/rc swaps on every device listed here).
*
* WARNING: Only paging space devices should be listed here.
*
```

```
* This file is modified by the chps, mkps and rmps commands and referenced
* by the lsps and swapon commands.

hd6:
        dev = /dev/hd6
        auto = yes
        checksum_size = 0


paging00:
        dev = /dev/paging00
        auto = yes
        checksum_size = 16
```

The **swapon –a** command will swapon any device with a stanza in /etc/swapspaces that either has no auto field or an auto field with yes as an assigned attribute.

Beginning with AIX V6.1the **mkps** command supports the new option -c. The -c option specifies the size of the checksum to use for the paging space verification in bits. Valid options are 0 (checksum disabled), 8, 16, and 32. If -c is not specified, it will default of 0. The **mkps** command always writes a stanza to the /etc/swapspaces file for a newly created paging device, setting its auto field according to the -a option (yes / no) and its checksum_size field according to the new -c option. The usage message of the **mkps** command is:

```
# mkps -?
mkps: Not a recognized flag: ?
0517-050 Usage: mkps [-a] [-n] [-t lv] [-c ChksumSize] -s NumLPs Vgname Pvname
        Makes a paging space using logical volume.
mkps [-a] [-n] -t nfs hostname pathname
        Makes a paging space using an NFS server
```

The **chps** command has also been enhanced in AIX V6.1 to support a -c checksum size option. The new option allows you to change the checksum size for existing paging devices. Note that the command will fail on swapped on paging devices, in order to prevent the inherent risks of changing the checksum size while pages are on disk and paging I/O is in progress. If the system administrator wants to change the checksum size for a device only in the /etc/swapspaces file, so that it will be effective the next time the device is swapped on, they can use the -c option in combination with the -f option. This option is also new to AIX and will have no effect if the -c option is not used at the same time or if the paging space is not swapped on.

The usage message of the **chps** command is:

```
# chps -?
chps: Not a recognized flag: ?
0517-030 Usage: chps [-s NewLPs | -d DecrLPs] [-a {y|n}] [-c ChksumSize] [-f]
Psname
        Changes attributes of a paging space.
```

In AIX V6.1, the **lsps** command will have the checksum size added to its output, displaying whatever value is in the device's /etc/swapspaces checksum_size field, or 0 if there is either no /etc/swapspaces stanza or no checksum_size field for the device:

```
# lsps -a
Page Space      Physical Volume   Volume Group Size %Used Active  Auto  Type Chksum
paging00        hdisk0            rootvg        512MB     1   yes   yes    lv    16
hd6             hdisk0            rootvg        512MB     3   yes   yes    lv     0
```

5

System management

In this chapter, the following system management enhancements are discussed:

5.1 Web-based System Manager enhancements

In this section, major Web-based System Manager changes are discussed.

5.1.1 The mknfsproxy and rmnfsproxy interfaces

This section describes the Web-based System Manager dialogs that will need to take the new **mknfsproxy** and **rmnfsproxy** commands into account. Those changes were introduced after AIX 5L V5.3 TL6.

Cache File Systems plug-in

Two new dialogs are introduced in the Cache File Systems plug-in. They are only visible if the bos.nfs.cachefs package is installed. These dialogs are accessible in the Cache File Systems sub-plug-in, from the File systems menu. The name of the new dialogs are Create Proxy Server and Remove Proxy Server (See Figure 5-1 on page 203).

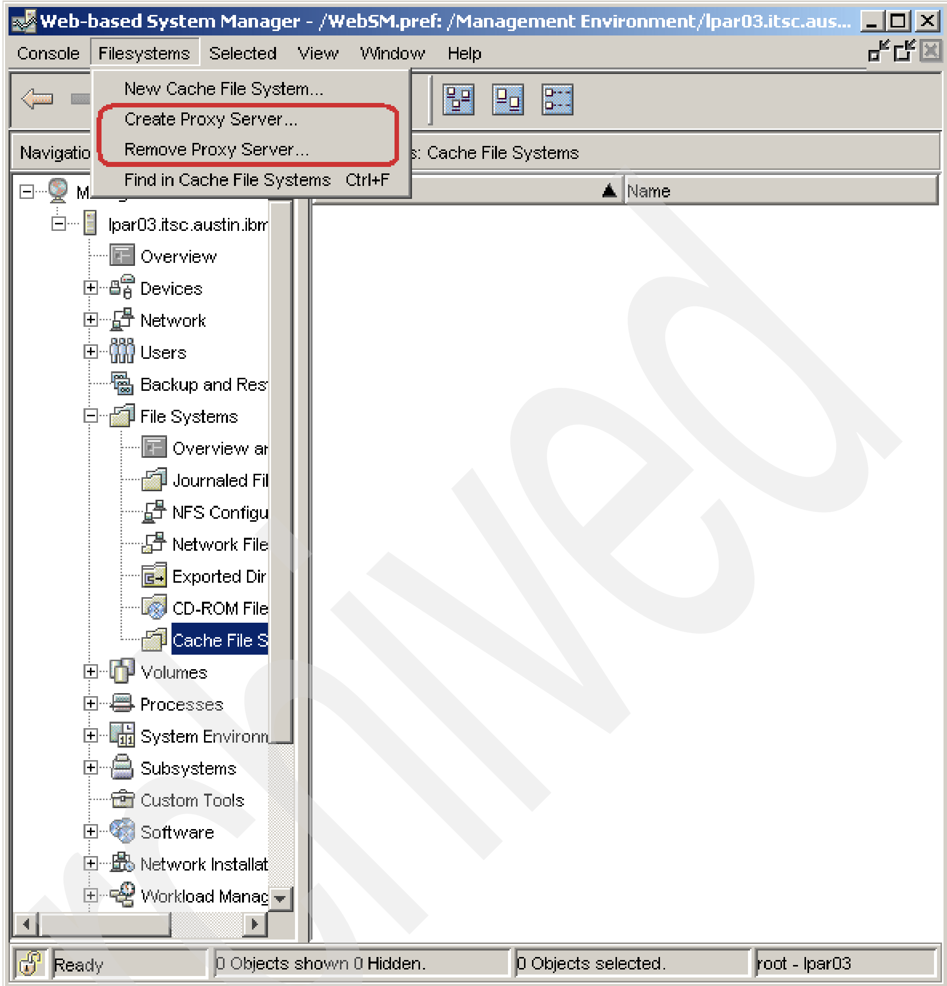


Figure 5-1 Proxy Server menus

These menu item launch the Create Proxy Server and Remove Proxy Server dialogs.

Create Proxy Server dialog

The Create Proxy Server dialog includes the fields shown in Table 5-1.

Table 5-1 Create Proxy Server dialog

Name	Description
Path name of mount point[a)]	This is the directory where the cache file system will be mounted.
Path name of remote directory[a)]	This is the directory on the remote host that the cache file system will access.
Host where remote directory resides[a)]	This is the remote host that the cache file system will access.
Mount options	These are the NFS mount options that can be optionally applied to the NFS client mount.
Cache directory[a)]	This is the local JFS2 file system where the cache file system will store the cached data and state.
Cache directory options	These are the cache file system configuration options, using the form param=n.
Export options[a)]	Specifies the NFS server export options for the created cache file system instance. If this is supplied, the created cache file system instance will also be NFS exported using the supplied options. If this option is not supplied, the created cache file system instance will be exported with the same NFS version specified by the mount options.
Whole file locking[b)]	When this check box is checked, it causes the cache file system instance to acquire a single lock from its associated NFS back end that covers the entire file when any byte range locks are requested. When the count of byte range locks drops to 0 (zero), the lock on the back-end NFS server is released.

a) This is a mandatory parameter.
b) This is a check box. The default is unchecked.

The OK button is only enabled when all the required fields are filled in. The Cancel button dismisses the dialog (see Figure 5-2).

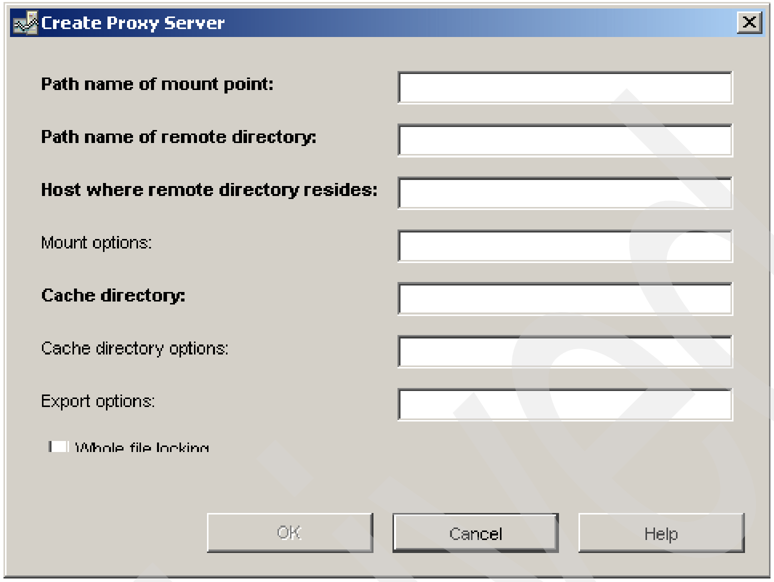


Figure 5-2 Create Proxy Server dialog

When the OK button is pressed, the dialog is dismissed and the following command is launched in a dialog box:

```
# /usr/sbin/mknfsproxy -L -c <cache directory> -d <mount point> [-o
<cache directory options>] -m [<mount options<] <remote host>:<remote
directory> [-e <export options>]
```

Remove Proxy Server dialog

The Remove Proxy Server dialog includes the fields in Table 5-2.

Table 5-2 Remove Proxy Server dialog

Name	Description
Path name of mount point[a]	Specifies where the proxy-enabled cache file system instance to be removed was mounted.
a) This is a mandatory parameter.	

The OK button is only enabled when all the required fields are filled in. The Cancel button dismisses the dialog (see Figure 5-3).

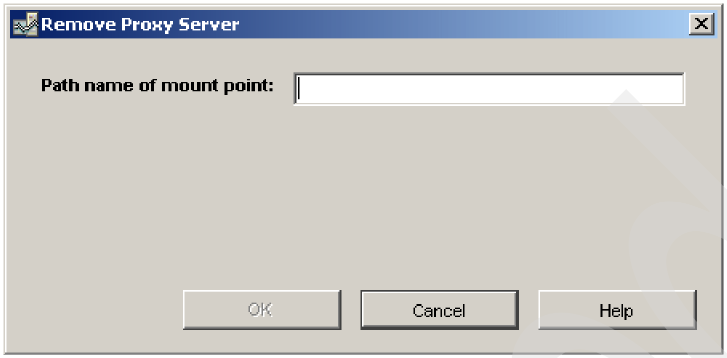


Figure 5-3 Remove Proxy Server dialog

When the OK button is pressed, the dialog is dismissed and the following command is launched in a dialog box:

```
# /usr/sbin/rmnfsproxy <mount point>
```

5.1.2 Modified Web-based System Manager menus

Some of Web-based Systems Manager menus are changed because of performance tool changes. Web-based System Manager does not display restricted parameters by default for tunables. So, to display these parameters, an administrator has to set the Show restricted Parameters Menu (see Figure 5-4).

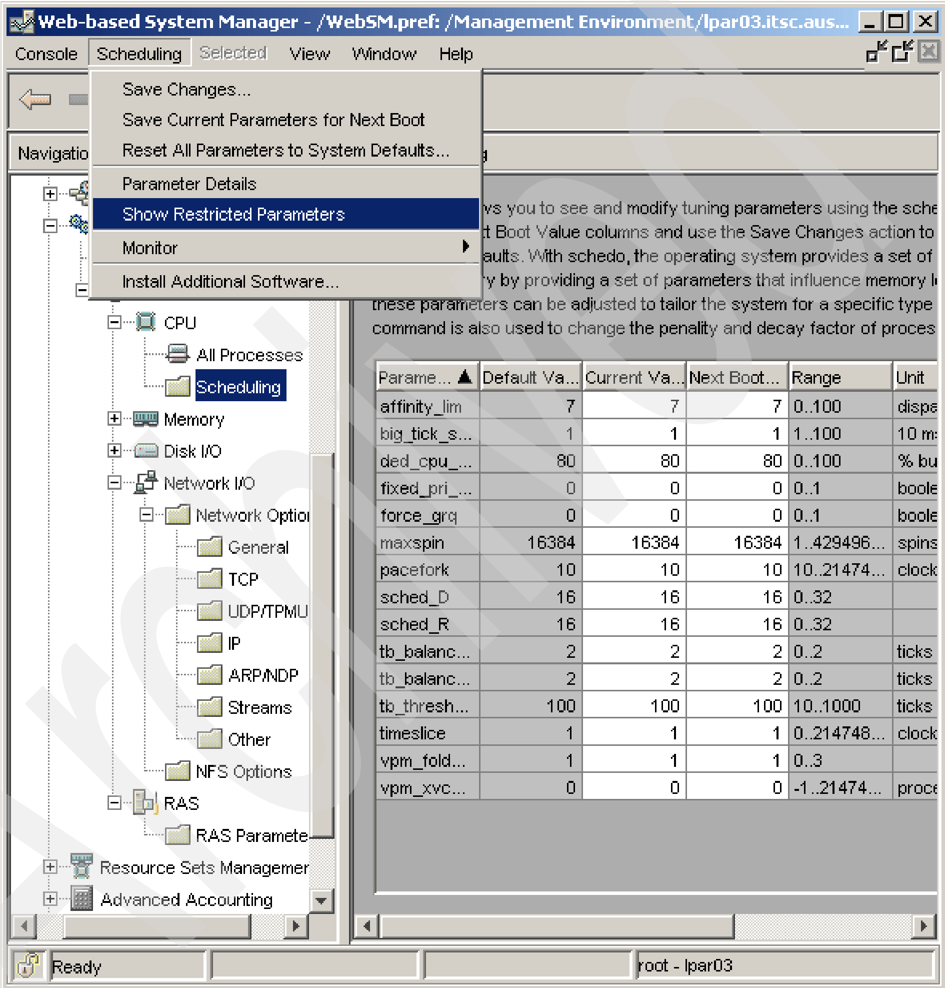


Figure 5-4 Example of Show Restricted Parameters

To access Show restricted Parameters, an administrator selects **Performance** → **System Tuning tasks**, and then accesses each resource and sub task. Then the administrator can show the menu from the top menu. The menus can be shown when the tasks shown in Table 5-3 are selected.

Table 5-3 List of resource names and task names and menus

Resources	Selection flow from Resources	Name of top menu
CPU	**CPU → Scheduling**	Scheduling
Memory	**Memory → Scheduling**	Scheduling
	Memory → Virtual Memory	Memory
Disk I/O	**Disk I/O → I/O Parameters**	I/O
Network I/O	**Network I/O → Network Options → General**	General
	Network I/O → Network Options → TCP	TCP
	Network I/O → Network Options → UDP/TPMU	UDP/TPMU
	Network I/O → Network Options → IP	IP
	Network I/O → Network Options → ARP/NDP	ARP/NDP
	Network I/O → Network Options → Streams	Streams
	Network I/O → Network Options → Other	Other
	Network I/O → NFS Options	NFS
RAS	RAS Parameters	RAS

For the options to be displayed, see 6.2, “Restricted tunables” on page 249.

5.2 AIX Print spooler redesign

Today, the AIX printing environment has evolved to be an enterprise ready spooling system capable of handling thousands of print queues and print jobs. But there have been no significant changes after the initial spooler design of AIX was introduced. So, the addition and deletion of print queues currently can take a significant amount of time on systems that have thousands or tens of thousands of queues. To improve the performance of the overall printing subsystem, AIX Print spooler design is changed in AIX V6.1.

5.2.1 Spooler command changes

The file /etc/qconfig holds the names and attributes of all the queues and devices on the system. There is some redundancy built into the way this file is read and handled by different programs.

Upon startup, The qdaemon daemon reads /etc/qconfig and generates a file named /etc/qconfig.bin containing complete queue/device information and returns a pointer to a list of queues to qdaemon. This list of queues is kept up to date throughout the life of qdaemon.

Other utilities, such as **mkque**, **mkquedev**, **lsque**, **lsquedev**, **rmque**, and **rmquedev**, which get called when a print queue is added, listed, or removed, respectively, also need to know what is in /etc/qconfig.

This change removes redundancy by exploiting the information that is already in /etc/qconfig.bin. Since that information is the most up-to-date state of what is in /etc/qconfig, it is simple enough to read /etc/qconfig.bin from **mkque**, **mkquedev**, **lsque**, **lsquedev**, **rmque**, and **rmquedev**.

5.3 Increase default size of argument area

The default argument area size used is increased in AIX V6.1. Previous to AIX V6.1, AIX has a relatively small default argument space (24 KB), which causes applications to fail when they are passed a large set of command-line arguments. Generally, when applications are passed with wild card characters, such as an asterisk, the argument size is indefinite and the current default of 24 KB is not sufficient to handle such requirements. At times, it can result in application core dump or memory failures. From AIX V6.1 onwards, the argument area is a configured parameter, but the default has never changed before. Hence, there is a need to increase the default argument size currently supported in AIX. The configurable range for the argument space is 24 KB to 4 MB.

Prior to AIX V6.1, the #defined values ARG_MAX and NCARGS, whose current value is 24576 (24 KB), are used:

- ARG_MAX value in limits.h file (It reflects a static size.)
- Return value from sysconf(_SC_ARG_MAX) (runtime value)

The argument size requirement generally depends on the amount of available memory. The actual value supported can be obtained using sysconf(), as shown above.

These NCARGS parameters are stored in the ODM stanza PdAt. The default value for ncargs is six blocks. As each block size is 4 KB, the default argument area is 6 * 4 KB, or 24 KB. If the size of the arguments exceeds the size of the argument area, the command does not execute and exits with an error message of `arg list too long`.

The existing ODM stanza for NCARGS is:

```
PdAt:
   uniquetype = "sys/node/chrp"
   attribute = "ncargs"
   deflt = "6"
   values = "6-1024,1"
   width = ""
   type = "R"
   generic = "DU"
   rep = "nr"
   nls_index = 71
```

To change the default value to 1 MB, the default field is updated to 256. The new ODM stanza is:

```
PdAt:
   uniquetype = "sys/node/chrp"
   attribute = "ncargs"
   deflt = "256"
   values = "6-1024,1"
   width = ""
   type = "R"
   generic = "DU"
   rep = "nr"
   nls_index = 71
```

Apart from this, the ARG_MAX value in the file limits.h is increased from 24576 (6 * 4 KB) to 1048576 (256 * 4 KB).

Note: Do no use ARG_MAX if your application needs to be aware of the runtime maximum argument size; use sysconf(_SRC_MAX_ARGS) instead. Refer to the /usr/sys/limits.h for more information.

You can check the current setting with the **lsattr** command:

```
(AIX V5.3)
# lsattr -R -l sys0 -a ncargs
6...1024(+1)
$ lsattr -El sys0 -a ncargs
```

```
ncargs 6 ARG/ENV list size in 4K byte blocks True

(AIX V6.1)
# lsattr -R -l sys0 -a ncargs
256...1024(+1)

$ lsattr -El sys0 -a ncargs
ncargs 256 ARG/ENV list size in 4K byte blocks True
```

The following steps show how things are done on AIX V6.1. The following code is a sample for checking runtime ncargs:

```
$ cat sysconf.c
#include <stdio.h>
#include <unistd.h>

void main(){

long num_args=0;

num_args=sysconf ( _SC_ARG_MAX );

printf("Number of Argument is %d\n", num_args);

}
```

The following code compiles and runs this sample program:

```
$ /usr/vac/bin/cc -o sysconf sysconf.c
$ ./sysconf
Number of Argument is 1048576
```

Tip: If you change ncargs from 256 to 512, the runtime values are dynamically changed as follows:

```
$ su
root's Password:
# chdev -l sys0 -a ncargs=512
sys0 changed
# exit
$ ./sysconf
Number of Argument is 2097152
```

5.4 Limit threads per process

AIX V6.1 provides a mechanism to limit the number of threads per process and the number of processes per user. Previous versions of AIX do not offer any direct mechanism for controlling these limits. Although the existing implementation is a traditional one, it has several limitations. The foremost is that a runaway or errant process can consume system resources by creating an excessive number of threads or processes, thereby reducing available system resources for other processes and users. At the user level, the existing implementation is also restrictive in that it does not allow users the fine-grained control over their own resource consumption and control that is in demand in certain critical markets. In this section, we discuss implementations, configurations, and considerations for this function.

5.4.1 Background

This feature originated in the High Performance Computing (HPC) sector, where a number of clients have desired this functionality. These clients often encounter the scenario where some of their researchers create programs that consume an high percentage of the available system resources. In the extreme case, these programs can greatly reduce system performance and thereby prevent other users and processes from making any progress. A way of handling this situation is required that provides greater user and process isolation.This feature provides users and system administrators the ability to set limits on the number of threads a process can create and on the number of processes that a user can create. Upon trying to create more threads or processes than allowed, the creation simply fails. The programmer is now required to properly handle thread and process creation failure within a program to limit using excessive resources.

5.4.2 Implemented mechanisms

The following mechanisms are introduced:

- Limiting the number of threads per process and the number of processes per user.
- Configuring the limits on the number of threads and processes both statically and dynamically.

5.4.3 Implemented functions

The following functions are also provided:

- Provides a mechanism for setting, getting, monitoring, and managing the limits imposed on the number of threads per process and the number of processes per user by extending the existing resource limit framework.
- Supports limiting the number of threads per process in both M:N and 1:1 thread modes. In M:N mode, the user space pthread library enforces the thread limit. In 1:1 mode, the kernel enforces the thread limit.
- Updates the kernel debugger output as necessary to display the new resource limits.
- To support availability efforts, when kernel thread creation fails or process creation fails as a result of hitting the limits, a trace entry is created.

5.4.4 Implemented changes

To support this function to limit values, system calls and defined values are changed.

Defined values

The following values are changed:

- RLIM_NLIMITS. In order to limit the number of threads per process, the existing functionality of the resource limit infrastructure is extended. Specifically, the macro defining the number of limits in sys/resource.h is increased from 8 to 10:

```
#define RLIM_NLIMITS                 10
```

- To support backwards compatibility, a new value is defined to keep track of how many rlimits there were prior to this modification. This is used for various functions, such as getprocs, which rely on the size of the requested structure to determine how to pack the data for the response to the caller. The new value is defined in sys/proc_compat.h as follows:

```
#define RLIM_NLIMITS_53             8
```

- To introduce the actual new limits, the new values are defined in sys/resource.h as follows:

```
#define RLIMIT_THREADS               8
#define RLIMIT_NPROC                 9
```

System calls

The following system calls are changed:

- getrlimit, getrlimit64, setrlimit, and setrlimit64

  There are no direct impact or modifications to the error codes returned from these functions. As with other resource limits, the limit on the number of threads per process is also enforced for kernel processes. Thus, the getrlimit() and setrlimit() services are supported for use by kernel processes as well as user processes. You can discover more about RLIMIT_THREADS and RLIMIT_NPROC by looking for information about the parameters for the getrlimit subroutine in the AIX InfoCenter.

- pthread_create

  The previous and current behavior of pthread_create is to return EAGAIN if WLM is enabled and the limit on the number of threads for a class has been exceeded. On AIX V6.1, pthread_create will also return EAGAIN if an attempt to create a thread fails as a result of exceeding the thread limit.

5.4.5 How to configure these limits

All configuration of these limits follow exactly the same manner as for all other existing resource limits. These resource limits can be set both statically and dynamically as follows:

- Static Configuration

  As with the other user attributes, RLIMIT_THREADS and RLIMIT_NPROC receive default initialization from the contents of /etc/security/limits. If these limits are not defined in this file, the existing AIX default behavior of having no limits on the number of threads per process and processes per user are applied. This is done by simply initializing with the unlimited value.

  > **Note:** As a special case, consider a migration installation, such as upgrading from AIX 5L V5.3 to AIX V6.1. In this case, there is no entry in the limits file for the limit on the number of threads per process or processes per user. If an entry is not found in the limits file, the limit defaults back to unlimited. Thus, in this special case with a migration installation, all users are given unlimited as their limits on the number of threads per process and processes per user. Since this is the existing behavior in AIX 5L V5.3, there is no change from the expected behavior for these upgrading users. Thus, no special configuration is required of an user if they want the existing behavior to persist through the upgrade.

- Dynamic Configuration

  In order to support dynamic configuration, the **ulimit** command and the built-in shell commands are changed, as described in the following section.

User space commands and shell modifications

The **/usr/bin/ulimit** command and the built-in shell **ulimit** command are modified in order to allow setting and getting the new resource limit. As a result, the option -r and -u is newly added to **ulimit** command. For static configuration, the threads and threads_hard options are also introduced to the **chuser** and **mkuser** commands.

Static configuration method

The administrator of a system can change the limit using the **chuser** command statically as follows:

```
# chuser threads=20 threads_hard=30 nobu
```

User nobu can execute 20 threads per process as a soft limit, and 30 threads per process as a hard limit.

Dynamic configuration methods

A user may dynamically change their limit using the **ulimit** command.

Changing the number of threads per process

The following example changes the number of threads per process:

```
$ ulimit -a
time(seconds)        unlimited
file(blocks)         2097151
data(kbytes)         131072
stack(kbytes)        32768
memory(kbytes)       32768
coredump(blocks)     2097151
nofiles(descriptors) 2000
threads(per process) unlimited
processes(per user)  unlimited
$ ulimit -r 20 <- Changing number of threads per process
$ ulimit -a
time(seconds)        unlimited
file(blocks)         2097151
data(kbytes)         131072
stack(kbytes)        32768
memory(kbytes)       32768
coredump(blocks)     2097151
```

```
nofiles(descriptors) 2000
threads(per process) 20
processes(per user)  unlimited
```

Changing number of process per user

The following example changes the number of processes per user:

```
$ ulimit -a
time(seconds)        unlimited
file(blocks)         2097151
data(kbytes)         131072
stack(kbytes)        32768
memory(kbytes)       32768
coredump(blocks)     2097151
nofiles(descriptors) 2000
threads(per process) unlimited
processes(per user)  unlimited
$ ulimit -u 20 <- Changing number of process per user
$ ulimit -a
time(seconds)        unlimited
file(blocks)         2097151
data(kbytes)         131072
stack(kbytes)        32768
memory(kbytes)       32768
coredump(blocks)     2097151
nofiles(descriptors) 2000
threads(per process) 20
processes(per user)  20
$
```

If you want to change the hard limit, specify the -H option.

Considerations

In this section, we discuss considerations for when a user changes limits.

- Settable values

  The range of allowable values for both new limits is [1,unlimited]. Specifically, a limit of 0 is unsupported since a user must be able to create at least one process and each process must have at least one thread. The value of unlimited is the existing default behavior in AIX, that is, no limitations are imposed on the number of threads per process or the number of processes per user.

- Reducing current values

  Attempting to dynamically reduce the limits below the current number of running threads or processes is supported. All future attempts to create additional threads while the thread limit is exceeded fails. Similarly, all future process creations while the process limit is exceeded also fails. The rationale for allowing the limits to be lowered below the number of currently running threads or processes is as follows:

  – First, it allows a user to set the desired limit and thereby prevent all future thread or process creations. A well-behaved application could potentially query its limits, and take efforts to reduce its thread or process count in order to be more resource conscious. If this were not allowed, the current number of threads in a greedy process or the number of processes a user has running would be an unwanted, artificial lower boundary on the limit.

  – Secondly, this implementation is the most consistent with other resource limit behavior. For example, a user can lower their file size resource limit below the size of one of their existing, larger files that have already been created. Future attempts to create a file larger than this limit then correctly fail.

5.5 Threading pthread default 1:1

This section introduces changing the default behavior of the pthreads library. After AIX V4.3.1, the contention scope is m:n (AIXTHREAD_MNRATIO) or process scope (AIXTHREAD_SCOPE=P) by default. But to run middleware (for example, Web Sphere MQ, Tivoli® Storage Manager, and so on) and user applications (especially Java™ applications) appropriately, the default behavior is often changed.

AIX V6.1 changes this behavior to 1:1 or system scope (AIXTHREAD_SCOPE=S) by default. If AIXTHREAD_SCOPE is set as system scope (S), AIXTHRED_MNRATIO is disabled and it works as 1:1. Table 5-4 shows the default values.

Table 5-4 AIX Thread environment valuables

Environment valuables	AIX 5L V5.3 and before (Default)	AIX V6.1 (Default)
AIXTHREAD_SCOPE	P	S
AIXTHREAD_MNRATIO	8:1	Disabled (act as 1:1)

5.6 RFC 2790 SNMP host resource groups

Simple Network Management Protocol with Distributed Program Interface Version 2 (SNMP-DPI-2) is an application layer protocol that gives administrators the ability to control and monitor managed devices in a network.

The AIX implementation of SNMP-DPI-2 consists of three major components:

- SNMP network management station
- SNMP agent
- SNMP sub-agent

An administrator interacts with a managed object through a network management station; the station communicates by using SNMP requests through UDP ports 161 and 162. The managed object has a centralized agent that communicates with the management station and translates SNMP requests into DPI® operations for distributed sub-agents using dpiPortForTCP, which is a port specified by the DPI API framework. RFC 1592 details the SNMP-DPI-2 interface.

Each sub-agent fulfills DPI operations using back-end hosts controlling Management Information Bases (MIBs). A particular sub-agent, host resources (hr), is specified in RFC 1514 (obsoleted by RFC 2790). It includes hosts for various information groups, such as Systems, Device, File Storage, Running Software, Running Software Performance, and Installed Software. Host resource information is stored on MIBs as variables and tables in a sub-tree structure and conforms to the Structure of Management Information (SMI) specification in RFC 1155. Most of the information required by hosts is stored in AIX Object Data Manager (ODM) classes.

AIX V6.1 implements two additional SNMP-DPI-2 hosts for the Running Software (hrSWRun), and Running Software Performance (hrSWRunPerf) information groups in compliance with RFC 2790.

The structure of the host resource sub-agent and its hosts is a sub-tree. The following listing illustrates the sub-agent structure down by one nesting level, with absolute SMI object identifiers in parentheses:

```
- host (1.3.6.1.2.1.25)
   - hrSystem       (1.3.6.1.2.1.25.1)
   - hrStorage      (1.3.6.1.2.1.25.2)
   - hrDevice       (1.3.6.1.2.1.25.3)
   - hrSWRun        (1.3.6.1.2.1.25.4)
   - hrSWRunPerf    (1.3.6.1.2.1.25.5)
   - hrSWInstalled  (1.3.6.1.2.1.25.6)
```

All six MIB variables listed below the host resource sub-agent identify an individual sub-tree which is managed by the AIX hostmibd daemon. You can use the **snmpinfo -m dump** command to explore the full structure of each of the six host resource groups.

5.6.1 The Running Software information group

The Running Software group host MIB stores information for software that is running or loaded into memory. This includes the operating system, device drivers, and applications. Running Software information is stored in host MIB variables. The MIB table, a special MIB variable, stores an entry for each piece of software running on the managed object. Each entry contains management information for the software, such as the name, runtime parameters, type, and status. All MIB objects have unique Object Identifiers (OIDs) that are assigned based on nesting levels on the SNMP-DPI-2 host resource sub-trees. OIDs for host MIBs are specified in RFC 2790. The OID for the Running Software group host is {host 4}. The conceptual name of the Running Software group host is hrSWRun.

The MIB sub-tree is organized such that host MIB variables are on the host MIB root level, tables are on the root level, the entry is a sub-tree of tables, and entry fields are sub-trees of entries. The following listing illustrates the sub-tree for hrSWRun:

```
- host (1.3.6.1.2.1.25)
   - hrSystem       (1.3.6.1.2.1.25.1)
   - hrStorage      (1.3.6.1.2.1.25.2)
   - hrDevice       (1.3.6.1.2.1.25.3)
   - hrSWRun        (1.3.6.1.2.1.25.4)
      - hrSWOSIndex        (1.3.6.1.2.1.25.4.1)
      - hrSWRunTable       (1.3.6.1.2.1.25.4.2)
         - hrSWRunEntry          (1.3.6.1.2.1.25.4.2.1)
            - hrSWRunIndex             (1.3.6.1.2.1.25.4.2.1.1)
            - hrSWRunName              (1.3.6.1.2.1.25.4.2.1.2)
            - hrSWRunID                (1.3.6.1.2.1.25.4.2.1.3)
            - hrSWRunPath              (1.3.6.1.2.1.25.4.2.1.4)
            - hrSWRunParameters        (1.3.6.1.2.1.25.4.2.1.5)
            - hrSWRunType              (1.3.6.1.2.1.25.4.2.1.6)
            - hrSWRunStatus            (1.3.6.1.2.1.25.4.2.1.7)
   - hrSWRunPerf    (1.3.6.1.2.1.25.5)
   - hrSWInstalled  (1.3.6.1.2.1.25.6)
```

5.6.2 The Running Software Performance information group

Process performance information is managed by the Running Software Performance group host. The Running Software Performance host uses a MIB table to store statistics, such as CPU usage and allocated memory for each piece of software in memory. The OID for the Running Software Performance group host is {host 5}. The conceptual name is hrSWRunPerf. This host is closely coupled with the hrSWRun host.

The hrSWRunPerf sub-tree is arranged similarly to the hrSWRun sub-tree, with the table on the root level, entries as a sub-tree of table, and entry fields as sub-trees of entries. The following listing depicts the structure:

```
- host (1.3.6.1.2.1.25)
   - hrSystem       (1.3.6.1.2.1.25.1)
   - hrStorage      (1.3.6.1.2.1.25.2)
   - hrDevice       (1.3.6.1.2.1.25.3)
   - hrSWRun        (1.3.6.1.2.1.25.4)
   - hrSWRunPerf    (1.3.6.1.2.1.25.5)
      - hrSWRunPerfTable   (1.3.6.1.2.1.25.5.1)
         - hrSWRunPerfEntry       (1.3.6.1.2.1.25.5.1.1)
            - hrSWRunPerfCPU             (1.3.6.1.2.1.25.5.1.1.1)
            - hrSWRunPerfMem             (1.3.6.1.2.1.25.5.1.1.2)
   - hrSWInstalled  (1.3.6.1.2.1.25.6)
```

5.7 IBM Systems Director Console for AIX

The IBM Systems Director Console for AIX is a new management tool (**pconsole**) for AIX V6.1 that:

- Enables converged consoles on AIX
- Enables AIX management in the converged console
- Works with a Workload Partition environment.

This management tool is based on the following components:

- Light Weight Infrastructure 7.1

  The Light Weight Infrastructure (LWI) has a small footprint, is simple to configure, and secures the infrastructure for hosting Web applications, Web services, and other application related components. The LWI is based on the Open Services Gateway Initiative (OSGi) architecture and is derived from WebSphere® Everyplace® Deployment 6.0. The LWI is comprised of the base OSGi/Eclipse service platform plus additional custom components and bundles that support Web applications, Web services, and the building of components.

- ISC Standard Edition 7.1

  The primary goal of the Integrated Solutions Console (ISC) is to provide a single platform that can host all the Web-based administrative console functions built by IBM server, software, and storage products in a manner that allows customers to manage solutions rather than specific IBM products.

5.7.1 Packaging and requirements

The IBM Systems Director Console for AIX is automatically installed after the AIX V6.1 installation is completed. The following filesets are installed:

```
sysmgt.pconsole.rte
sysmgt.pconsole.apps.wdcem
sysmgt.pconsole.apps.websm
sysmgt.pconsole.apps.wrbac
sysmgt.pconsole.apps.wsmit
lwi.runtime
```

It requires 512 MB (default) of heap memory. You can customize the heap size. See 5.7.9, “Configuration and management” on page 240 for more details.

When the network configuration is finished, you can access it using a Web browser and entering your user name and password for the configured target system:

```
URL
http://<hostname>:5335/ibm/console
https://<hostname>:5336/ibm.console
```

Supported browsers are Internet Explorer Version 7 and Mozilla Firefox.

Figure 5-5 shows this tool.

Figure 5-5 IBM Systems Director Console for AIX Welcome page

5.7.2 The layout of the IBM Systems Director Console

IBM Systems Director Console for AIX consists of the following elements:

- ► Console toolbar
- ► Navigation area
- ► Work area

These components are discussed in the following sections.

Console toolbar across top

The console toolbar (Figure 5-6) provides the following functions:

- ► User name (for example, “Welcome root”)
- ► Help
  - – Infocenter window
    - • ISC Help
    - • IBM Systems Director Console for AIX Administrators Guide
- ► Logout

Figure 5-6 Console toolbar

Navigation area

The navigation area (Figure 5-7 on page 224) is a guide to tasks. In this area, the following task categories can be expanded or collapsed (for example, OS Management):

- ► Task categories
  - – Welcome
  - – My Startup Pages
  - – OS Management (AIX settings)
  - – Health
  - – Settings (Console settings)

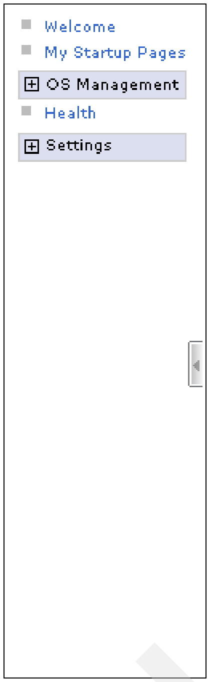


Figure 5-7 Navigation area

Work area

In the work area (Figure 5-8), the administrator can open several pages, and change from page A to B by using the page bar:

- Page bar
  - Multiple pages/tabs
  - Action Selection List (Close, Open, Refresh, and add to Startup Pages)

Figure 5-8 Page bar

- Portlets

  Portlets are shown in Figure 5-9 on page 225. Administrators can operate any tasks on the portlets.

Figure 5-9 Portlets

5.7.3 My Startup Pages (customization)

You can customize startup pages by:

- Defining which applications to start upon login.
- Managing them through the **My Startup Pages** application.

Each user can create individual customizations.

You can choose these functions from the OS Management menu. If you want to add functions to your start pages, use the following instructions:

1. Open the page that you want to add to My Startup Pages.
2. Select **Action Selection List** → **Add to My Startup pages**.

5.7.4 Health Summary plug-in

The Health Summary plug-in adds new vsm information to IBM Systems Director for AIX. It provides multiple portlets as follows:

Summary The summary portlet provides system configuration, network configuration, and paging space configuration.

Metrics The metrics portlet displays physical memory and virtual memory paging space CPU (both total and each of the CPUs) utilization. It provides values and a chart of those values.

Top Processes You can see processes that use the highest CPU utilization. It shows the process ID, parent process ID, CPU utilization, CPU time, and user ID.

File systems Shows a list of mounted file systems. You can see the mount point, size, and utilization.

5.7.5 OS management

If you manage a system, you can use the tasks provided in Table 5-5 on page 227.

Table 5-5 OS management tasks

Tasks	Description
Security and Users	Security and Users provides you with tools that you can use to perform common tasks related to user authentication and access. Use the task links on the right to quickly jump to those tasks.
Role Based Access Control	Role Based Access Control (RBAC) provides the ability to create and manage user roles that restrict access to files and other resources based on the user's active roles. This application supports the creation and management of roles, authorizations, privileged commands, and privileged devices. RBAC provides an enhancement over previous role management.
Manage the Cryptography Standard	You can determine the standards here.
System Environments	System Environments provides you with tools that you can use to perform common tasks related to system characteristics. Use the task links on the right to quickly jump to those tasks.
Print Spooling	Print Spooling provides you with tools that you can use to perform common tasks related to the spooler subsystem, which is the queuing system for printing. Use the task links on the right to quickly jump to those tasks.
Communications Applications and Services	Communications Applications and Services provides you with tools that you can use to perform common tasks related to communication applications. Use the task links on the right to quickly jump to those tasks.
System Storage™ Management	System Storage Management provides you with tools that you can use to perform common tasks related to physical and logical volumes, file systems, directories, files, and backups. Use the task links on the right to quickly jump to those tasks.

Tasks	Description
Processes and Subsystems	Processes and Subsystem provides you with tools that you can use to perform common tasks related to managing the system processes, subsystems, and subservers. Use the task links on the right to quickly jump to those tasks.
Problem Determination	Problem Determination provides you with tools that you can use to perform common tasks related to problem identification and resolution. Use the task links on the right to quickly jump to those tasks.
Performance and Resource Scheduling	Performance and Resource Scheduling provides you with tools that you can use to perform common tasks related to the performance, job scheduling, and workload. Use the task links on the right to quickly jump to those tasks.
Devices	Devices provides you with tools that you can use to perform common tasks related to physical devices. Use the task links on the right to quickly jump to those tasks.
Advanced Accounting	Advanced Accounting provides you with tools that you can use to perform common tasks related to the collection and recording of job related information. Use the task links on the right to quickly jump to those tasks.
Software Installation and Maintenance	Software Installation and Maintenance provides you with tools that you can use to perform common tasks related to installing new software, or managing previously installed software. Use the task links on the right to quickly jump to those tasks.
Software License Management	Software Installation and Maintenance provides you with tools that you can use to perform common tasks related to installing new software, or managing previously installed software. Use the task links on the right to quickly jump to those tasks.

Tasks	Description
Workload Partition Administration	Workload Partition Administration provides you with tools that you can use to perform common tasks related to the workload partitions (WPAR). Use the task links on the right to quickly jump to those tasks.
Cluster Systems Management	Cluster Systems Management (CSM) provides you with tools that you can use to perform common tasks related to setting up and maintaining a cluster of nodes that run the AIX or Linux operating system. Use the task links on the right to quickly jump to those tasks.
Distributed Command Execution Manager	Distributed Command Execution Manager (DCEM) provides you with tools to create and execute commands across a group of machines on a network.
System Management Interface Tool (SMIT)	A SMIT menu is provided.
Web-based System Manager	A Web-based System Manager Menu is provided.

Most of the tasks are the same as the tasks that SMIT provides. In this section, we focus on the following newly introduced tasks;

- Distributed Command Execution Manager (DCEM)
- System Management Interface Tool (SMIT)
- Web-based System Manager

These tasks are discussed in the following sections.

Distributed Command Execution Manager

Distributed Command Execution Manager (DCEM) provides an interface to the distributed shell (**dsh**). The distributed shell is a command targeting a cluster of remote systems. DCEM can save command specifications. A Perl script is created by the command specification for reuse. DCEM supports **dsh**, CSM, and NIM hosts and groups. It also supports **rsh** and **ssh** authentication.

The following figures are an example of executing HelloWorld with **rsh**. Table 5-6 provides the various incarnations:

1. Select **Execution**. Input the information shown in Table 5-6 into the fields shown in Figure 5-10 on page 231.

Table 5-6 Distributed Command Execution Manager HelloWorld example

Input Items	Description	Required?	Example
Name	Job Name	Optional	HelloWorld
Description (optional)	Description for Job	Optional	
Path (Default $PATH)	PATH		$PATH
Default User (Default Login user)	User to execute command		root
Command	Command to be executed	Mandatory	/usr/bin/echo helloworld

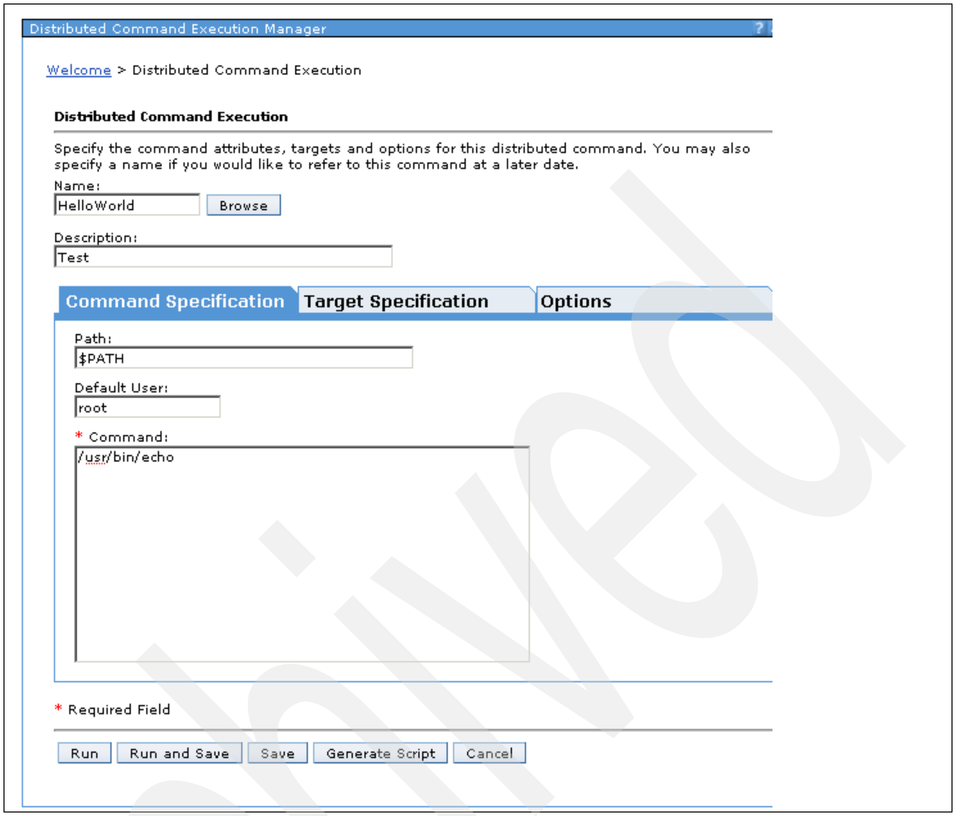


Figure 5-10 Distributed Command Execution Manager menu

2. Select the **Target Specification** tab and input the information shown in Table 5-7 into the fields shown Figure 5-11 on page 233.

Table 5-7 Target Specification input

Input item	Subcategory	Description	Examples
DSH Targets	DSH Hosts	Target system IP address or host name	n.n.n.n
	DSH Groups		-
CSM Targets	CSM Hosts	Target system IP address or host name	-
	CSM Groups		-
NIM Targets	NIM Hosts	Target system IP address or host name	-
	NIM Groups		-

You can select one of items.

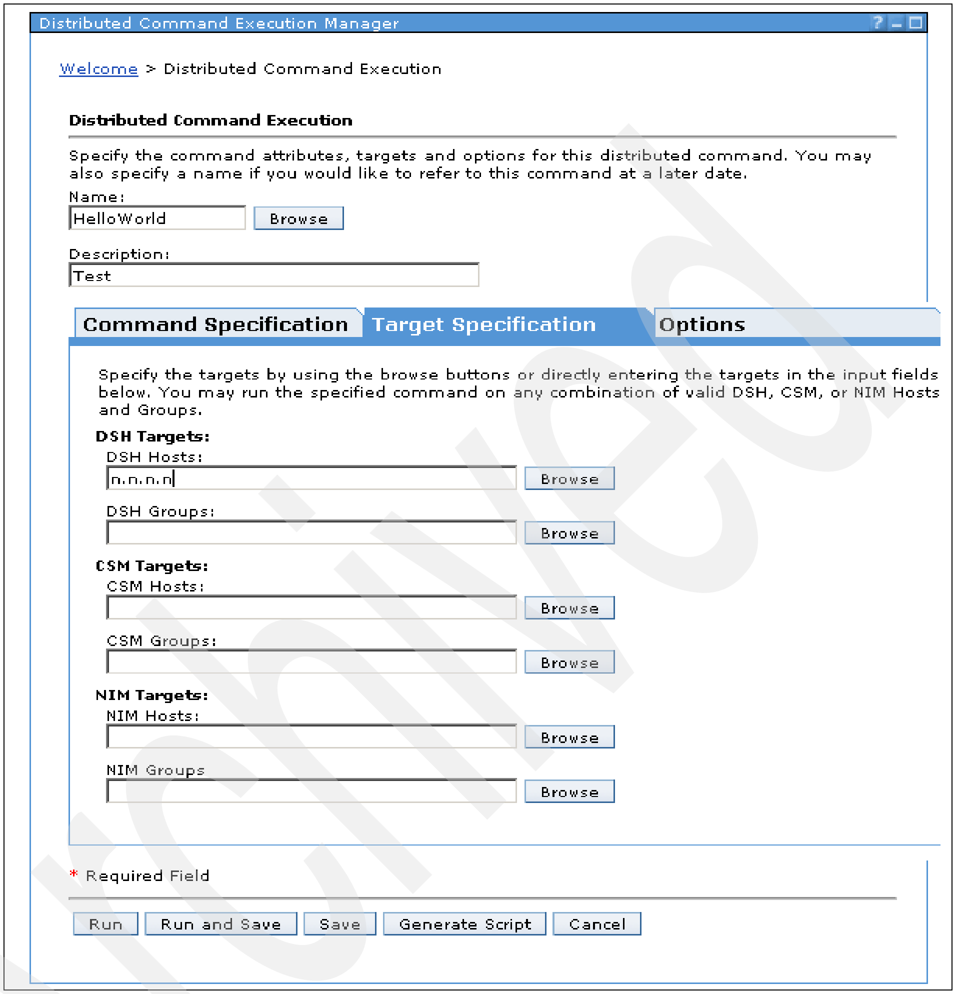


Figure 5-11 Target Specification tab

3. Select the **Option** tab and fill in the fields as shown in Figure 5-12.

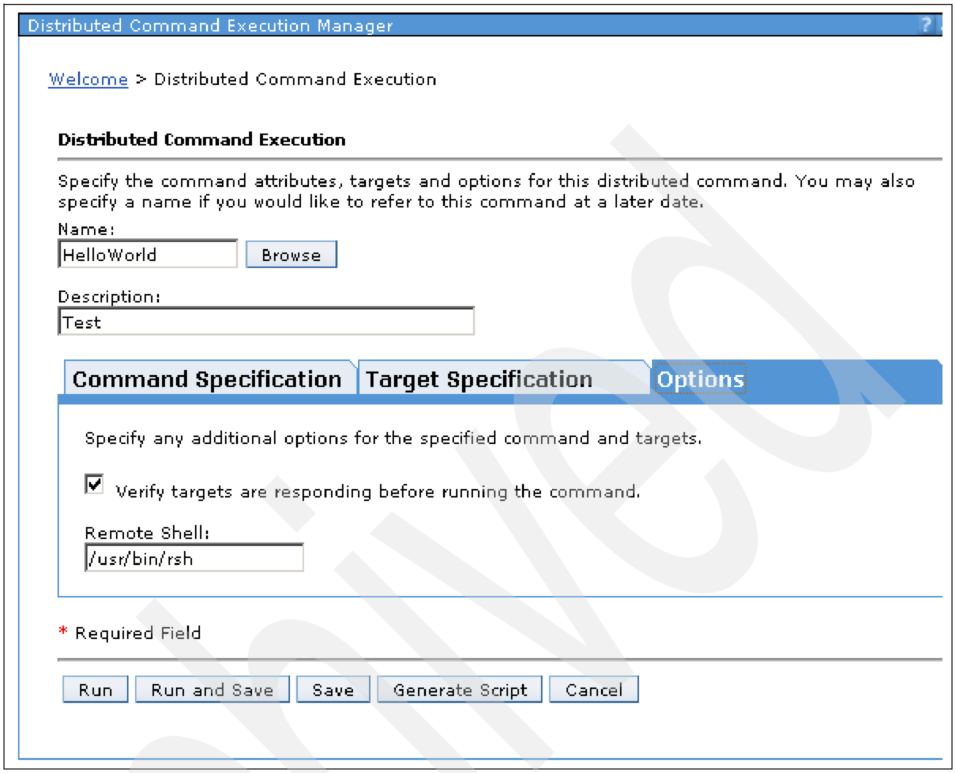


Figure 5-12 Option tab

4. When the input is correctly entered, you can execute by clicking the **Run** button.

SMIT

IBM Systems Director Console for AIX provides a Web interface for SMIT stanzas. The interface is dynamically generated. Classic SMIT supports all stanzas defined in the ODM.

Through the manager, SMIT appears as in Figure 5-13 on page 235.

Figure 5-13 System Management Interface Tools menu

Web-based System Manager

IBM Systems Director Console for AIX only provides an interface to execute Web-based System Manager (**wsm**). To use Web-based System Manager, you have to configure the server and client for Web-based System Manager as follows;

- Install and configure HTTPServer on the server system.
- Install and configure Web-based System Manager on the server system.
- Download and install the Java Webstart client on the client system.

After the configuration is finished, you can execute **wsm**, which works independently of IBM Systems Director Console for AIX.

5.7.6 Managing Workload Partitions

By default, the **pconsole** command is not installed with a Workload Partition system. So, you have to execute the following command to enable it:

```
/opt/pconsole/bin/wparConsole.sh -e
```

> **Note:** Some system management tasks will fail within a Workload Partition because these tasks may affect global resources, such as physical devices.

5.7.7 Settings

The Settings Task Guides provide the following tasks:

- Manage Global Refresh
- Credential Store
- Console Logging and Tracing
- Console User Authority
- My Active Roles

These tasks are discussed in the following sections.

Manage Global Refresh

Manage Global Refresh is a function to specify the interval time for refreshing portlets. For example, portlets from the Health Task Guide are set as follows:

- HealthSummary Portlet Entity
- HealthMetrics Portlet Entity
- HealthMetricDetail Portlet Entity
- HealthTopProcesses Portlet Entity
- HealthFileSystem Portlet Entity

Credential Store

If you need to change SSL keys, you can change them with this task. For more information, see “HTTPS (SSL)” on page 237.

Console Logging and Tracing

See 5.7.9, “Configuration and management” on page 240.

Console User Authority

See “Roles” on page 238.

My Active Roles

If you use a user that is assigned to New Role, you have to use this Task to check if your role is active or not. If your role is not active, change it to active in this task.

- Users can be viewed and activated with the My Active Roles application in the Settings category.
- A maximum of eight AIX roles can be active for a user at one time.

For more information, see “Roles” on page 238.

5.7.8 AIX security

IBM Systems Director Console for AIX implements the following security functions, discussed in the following sections:

- HTTPS (SSL)
- Authentication (login)
- Authorization (roles)

HTTPS (SSL)

IBM Systems Director Console for AIX does not support plain socket connections. HTTPS is enabled out-of-the-box. Its characteristics are:

- Default certificate and keystore password
- Same for all LWI installations
- Browser warnings
  - Signer not recognized
  - Domain name mismatch

If you want to change the SSL settings, consult the product documentation. The following settings provide a summary:

- Use iKeyman to manage certificates:
  a. Run **`/usr/java5/jre/bin/ikeyman`**.
  b. Delete the default certificate.
  c. Create a new certificate request or self-signed certificate.
  d. Change the keystore password.
  e. The keystore is stored in pconsole/lwi/security/keystore/ibmjsse2.jks.

- Update the console properties:
  a. Run **/pconsole/lwi/conf/webcontainer.properties.**
  b. Stop the console runtime.
  c. Copy webcontainer.properties to sslconfig.
  d. Edit sslconfig.
  e. Rename or remove webcontainer.properties.
  f. Start the console runtime:
     - The new webcontainer.properties is created with obfuscated passwords.
     - The sslconfig file is removed.

You may need to check the following files:

- /pconsole/lwi/conf/sslconfig
- /pconsole/lwi/conf/webcontainer.properties

Authentication (login)

If an administrator is required to log in to the system using IBM Systems Director Console for AIX, they are required to have an AIX login account on the server (just as it is for an AIX system user). To log in to the system, enter your AIX user name and password. IBM Systems Director Console for AIX provides a single console session per user per server. If you encountered a login problem, please check the following items:

- ☐ No user account on the server?

  Have the administrator create an account.
- ☐ Password expired or not set (new user account)?

  Log in using a local terminal or telnet and set the password.
- ☐ Already logged into the console?

  Look for the warning message that gives you the option to terminate the previous session.

After some of the above items are resolved, retry the login.

Roles

There are two roles for IBM Systems Director Console for AIX: console roles and administrator roles.

Console roles

An administrator can assign users to console roles by selecting **Settings** → **Console User Authority**. The Console Administrator (root by default) can assign users to roles with the User Authority app under Settings. The console roles are defined by the console applications and not integrated with the AIX roles. Those roles are saved in file private to ISC.

Setting up console authorizations

If you always plan to log in as another administrator (except root), the console administrator role (root by default) can be assigned to non-root users with the User Authority application.

To assign an administrator role, Enhanced RBAC must be enabled:

> **Note:** The Welcome Page that appears when the console administrator logs in guides you through the console setup.

1. Role Based Access Control (RBAC)

   Define a role in AIX for each unique administrative job.

2. Security and Users

   Add roles to administrative user ID properties in AIX.

3. Console Authorization

   Define user IDs in the console and grant permission to navigation categories.

4. Activate Role

The console compares the authorizations the user has with the authorizations that the application identifies as required and displays a warning message if the user is missing authorizations.

The console executes commands with the users UID and his active authorizations.

Examples

The following is a role assignment example:

1. Create new role:
   a. AIX Roles: *NobuRole.*
   b. AIX Authorizations in NobuRole: aix.device.config.printer, aixdevice.stat.printer, aix.security.role, aix.security.user, aix.security.group, and aix.security.passwd.

2. Non-root user *nobu* authorized for console tasks. AIX user name: *nobu.*
3. Console Roles: aixUsers and aixPrinters.
4. AIX roles may not be active by default. You must select **Setting Task Group** → **My Active Roles**, and check **yes** to activate your role.

5.7.9 Configuration and management

The following is information for system configuration and management:

- Plug-ins use the /pconsole/apps/eclipse/plug-ins directories.
- The configuration files are in /pconsole/lwi directories.
- `pconsole` is defined in SRC to deal with signals.
- The `pconsole` heap size is defined in /pconsole/lwi/conf/pconsole.javaopt.
- The `pconsole` logs are kept in the /var/log/pconsole/logs directory.

  The log files are written by XML. The logs rotate using the file names error-log-*n*.xml and trace.log-*n*.xml.
- wSMIT.log

  If you use classic SMIT, the log file is located in $HOME/wsmit.log. The content of the log file is the same as $HOME/smit.log.
- DCEM log

  The log files are located in $HOME/dcem/logs/decm.log.

5.8 VMM dynamic variable page size

Pages are fixed-length data blocks held in virtual memory. The page size defines the unit size of the memory portions allocatable by the operating system or an application. The supported page sizes is both dependent on the hardware architecture as well as on the operating system. The IBM System p servers and AIX V6.1 support the page sizes shown in Table 5-8.

Table 5-8 AIX and POWER page size support

Page size	Required processor architecture	Description
4 KB	ALL	The standard page size for all AIX 5L and older versions running on POWER™ architecture.
64 KB	POWER5+™ or later	This page size was introduced with AIX 5L V5.3 TL 5300-04 multiple page size support.

Page size	Required processor architecture	Description
16 MB	POWER4™ or later	Also called large pages. It is intended only for high performance computing (HPC). Use the **vmo** command to enable large page sizes. It was introduced with AIX 5L V5.1 ML 5100-02.
16 GB	POWER5+ or later	Also called huge pages. It is intended only for high performance computing (HPC). Use the Hardware Management Console (HMC) to enable huge page sizes. This page size was introduced with AIX 5L V5.3 TL 5300-04 multiple page size support.

AIX V6.1 and the POWER6 architecture introduce dynamic variable page size support (VPSS):

- VMM can dynamically use a larger page size based on the application memory usage. This will improve memory access performance.
- The use of larger page sizes is transparent to applications.
- VPSS is activated by default if the underlying hardware supports it.
- With the default settings, AIX will use larger page sizes only if it does not result in an increase in memory usage for a process.
- You can use **vmo** tunables to influence VPPS behavior.

> **Important:** If you are using POWER6 System p 570 servers, make sure your system firmware is at level EM320 or later to support variable page sizes.

5.8.1 Variable page size concept

Using larger page sizes increases memory access performance, since fewer address translations in the hardware have to be done and the caching mechanisms can be used more efficiently. On the other hand, memory regions may be wasted if a larger page size is allocated and then populated with data less than the page size.

AIX 5L V5.3 TL 5300-04 added support for the medium size 64 KB pages. In order to use the medium page sizes, applications need to be recompiled or explicitly set loader shell environment variables. Starting with AIX V6.1 on POWER6-based processors, the Virtual Memory Manager (VMM) can dynamically promote pages to a larger page size. This page promotion is completely transparent to the application and will be done without the need for user intervention.

The variable page size support is based on the processor's ability to have mixed page sizes within the same memory segment. Every mixed page size segment has a minimum and a maximum page size. At the time of writing, the POWER6 architecture and AIX V6.1 supports 4 KB pages as the minimum and 64 KB pages as the maximum.

AIX will continue to support explicit selection of page sizes using the existing mechanisms of system calls and loader options. When an explicit page size is specified for a memory region, AIX will treat the specified page size as the minimum page size of the memory region. VMM may dynamically use pages larger than the page size specified for a memory region.

> **Note:** VMM will only support dynamically varying the page size of working storage memory.

5.8.2 Page size promotion

The AIX V6.1 default behavior is to divide every memory segment into equal-sized ranges based on the maximum page size for the segment. AIX will decide the page size to use for each range independently of the other ranges in a variable page size segment. This is shown in Figure 5-14.

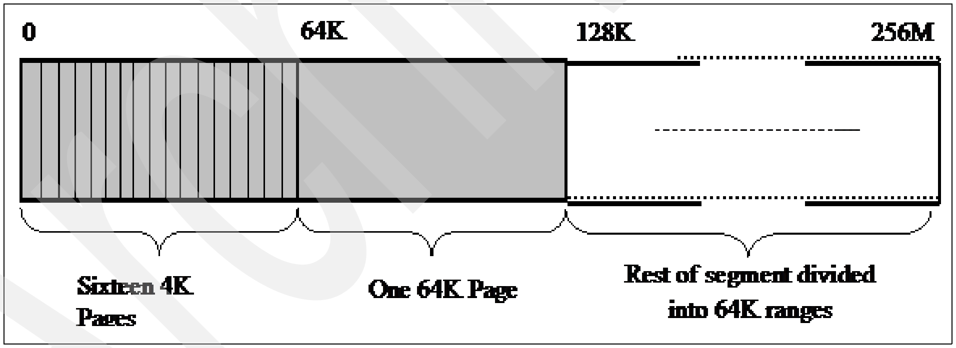


Figure 5-14 Mixed page size memory segment used by VPSS

The VMM starts allocating 4 KB page frames for a memory range until a sufficient number of 4 KB pages in the memory range had been referenced to allow promotion of the memory range to a larger page size. In this case, 16 4 KB pages are needed. Promotion to the larger page size of 64 KB requires that all the pages have the same state (the same read/write page protection, non-exec protection, storage key protection, and not in an I/O state). If this is the case, the 4 KB address translations are removed and replaced with a 64 KB address translation.

The 64 KB address translations are used as long as all 16 4 KB pages continue to have the same state. State changes, such as through the mprotect subroutine, or page stealing of the LRU daemon, cause demotion to the 4 KB page size.

VMM will dynamically adjust page sizes at a page granularity level. Therefore, different data regions of a single process might be allocated in both 4 KB and 64 KB pages at the same time. The AIX operating system's dynamic use of larger page sizes is completely transparent to applications and kernel extensions. When VMM has dynamically selected a larger page size for a memory region, all system calls and kernel APIs will indicate that 4 KB pages are being used for the memory region.

5.8.3 The vmo command tunables

With the support of variable page sizes, AIX V6.1 introduces a new **vmo** tunable setting, vmm_default_pspa, and extended the existing vmm_mpsize_support.

vmm_default_pspa

Some applications perform better with a larger page size, even when the maximum page size (64 KB) region is not fully referenced. The **vmo** tunable page size promotion aggressiveness factor (PSPA) can be used to alter the requirement that all allocated 4 KB pages have to contain data before they get promoted to the larger page size.

You can specify a numeric value between 0 and 100. This percent value is treated as the inverse of the page promotion threshold. In other words, a value of 0 means that all the 16 4 KB pages have to be referenced in order to get promoted to a 64 KB page. With a value of 50, eight 4 KB pages are needed for promotion while a value of 100 forces a promotion at the first reference to that memory region. The default value is 0.

A value of -1 indicates that no page promotion will be done by VMM. Note that the default value is -1 if no hardware support can be detected.

Page size promotion thresholds are only considered at segment creation time. Therefore, changed values of the vmm_default_pspa tunable will only affect new segments.

This setting is valid system-wide. In order to change PSPA on a application level, code changes and a recompile are required. AIX V6.1 introduces new system call named vm_pattr() to alter the PSPA weight.

vmm_mpsize_support

The vmm_mpsize_support tunable toggles the AIX multiple page size support for the extra page sizes provided by POWER5+ and later systems. The new value of 2 is introduced to support dynamic variable page sizes. Table 5-9 shows all the possible values. The default value in AIX V6.1 is 2.

Table 5-9 vmo vmm_mpsize_support tunable

Value	Description
0	Only page sizes of 4 KB and 16 MB are recognized.
1	AIX will take advantage of the additional page sizes supported by a processor.
2	AIX will take advantage of the capability of using multiple page sizes per segment.

To make changes to the vmm_mpsize_support tunable, run the **bosboot** command and reboot AIX.

5.8.4 The svmon command enhancements

The **svmon** command used a single character qualifier ('s', 'm', 'L', and 'S') to represent the segment page size (respectively 4 KB, 64 B, 16 MB, and 16 GB). Starting with AIX V6.1, the **svmon** command supports dynamic variable page sizes by using two characters to represent the minimum and maximum page sizes attributes for each segment. The following example shows the **svmon -P** output for mixed page sizes (4 KB and 64 KB) in a memory segment for the init process:

```
# svmon -P 1

-------------------------------------------------------------------------------
     Pid Command          Inuse      Pin     Pgsp  Virtual 64-bit Mthrd 16MB
       1 init             13834     8085        0    13818      N     N    N

     PageSize                Inuse        Pin       Pgsp    Virtual
     s    4 KB                 186          5          0        170
     m   64 KB                 853        505          0        853

    Vsid      Esid Type Description              PSize  Inuse   Pin Pgsp
Virtual
       0         0 work kernel segment               m    555   505    0 555
   3c02d         d work shared library text          m    298     0    0 298
   15001         2 work process private             sm    101     5    0 101
   6501d         f work shared library data         sm     69     0    0  69
   7d01b         1 clnt code,/dev/hd2:531            s     11     0    -   -
```

```
  1d023         - clnt /dev/hd4:724                 s     5     0    -   -
```

The **svmon -l** command displays separate statistics for every page size in a mixed page size segment:

```
# svmon -P 1 -l

-------------------------------------------------------------------------------
     Pid Command          Inuse      Pin     Pgsp  Virtual 64-bit Mthrd 16MB
       1 init             13834     8085        0    13818      N     N   N

     PageSize                Inuse        Pin       Pgsp    Virtual
     s    4 KB                 186          5          0        170
     m   64 KB                 853        505          0        853

    Vsid      Esid Type Description              PSize  Inuse Pin Pgsp Virtual
       0         0 work kernel segment               m    555   505    0 555
                   System segment
   3c02d         d work shared library text          m    298     0    0 298
                   Shared library text segment
   15001         2 work process private              s    101     5    0 101
                                                     m      0     0    0   0
                   pid(s)=1
   6501d         f work shared library data          s     69     0    0  69
                                                     m      0     0    0   0
                   pid(s)=1
   7d01b         1 clnt code,/dev/hd2:531            s     11     0    -   -
                   pid(s)=1
   1d023         - clnt /dev/hd4:724                 s      5     0    -   -
                   pid(s)=213100, 1
```

The **svmon -q** command has been enhanced to accept the new two character qualifiers, which refer to mixed page size segments:

```
# svmon -S -q sm | head -20

    Vsid      Esid Type Description              PSize  Inuse   Pin PgspVirtual
   20028         - work other kernel segments       sm  15040 15040    0 15040
    8002         - work other kernel segments       sm   1526     0    0  1526
   24009         - work other kernel segments       sm   1518  1433    0  1518
   6925e         - work                             sm   1480     0    0  1480
   3400d         - work other kernel segments       sm    523     0    0   523
   3800e         - work other kernel segments       sm    522     0    0   522
   59272         - work                             sm    371     0    0   371
   1d2e3         - work                             sm    352     5    0   352
   3000c         - work other kernel segments       sm    352   346    0   352
   792da         - work                             sm    258     0    0   258
   15261         - work                             sm    248     5    0   248
    c003         - work other kernel segments       sm    240   240    0   240
   512d0         - work                             sm    213     5    0   213
    1204         - work                             sm    213     0    0   213
   6931e         - work                             sm    202     5    0   202
   292ce         - work                             sm    202     0    0   202
    1224         - work                             sm    196     5    0   196
   3d1cb         - work                             sm    194     0    0   194
```

6

Performance management

The performance of a computer system is evaluated based on clients expectations and the ability of the system to fulfill these expectations. The objective of performance management is to balance between appropriate expectations and optimizing the available system resources.

Many performance-related issues can be traced back to operations performed by a person with limited experience and knowledge who unintentionally restricts some vital logical or physical resource of the system. Most of these actions may at first be initiated to optimize the satisfaction level of some users, but in the end, they degrade the overall satisfaction of other users.

AIX Version 6 introduces many new performance management enhancements:

- 6.1, “Unique tunable documentation” on page 248

  A unique documentation repository for all tunables of the six AIX tuning commands.
- 6.2, “Restricted tunables” on page 249

  The tunable classification *##Restricted parameter* helps you avoid user modification mistakes on critical performance tunables.
- 6.3, “AIX V6 out-of-the-box performance” on page 262

  A new AIX default set of tunables values that helps you avoid setting base operating system parameters for a newly installed system, the so-called tuning out-of-the-box, or default, performance.

- 6.4, “Hardware performance monitors” on page 271

  Enhancements on the AIX low-level performance monitors helps you detect more accurately a server problem-determination issue against a pure performance issue.

6.1 Unique tunable documentation

Because of the large number of tunables available, the need to adjust the tunables default values, their value ranges, and the need to add new tunables as platform complexity evolves, the static nature of the corresponding system documentation and tunable help messages has become increasingly difficult to manage.

The help messages of the tuning commands now contain the complete tunables descriptions and allowed settings. Thus, the full list of the system tunable parameters and details of their use are no longer available at the AIX documentation or man pages level. This method ensures a single method for a user to know the exact functions a command currently has.

The tunable description message for the six tuning commands (**vmo**, **ioo**, **schedo**, **raso**, **no**, and **nfso**) can be displayed through the new -h <tunable> option.

The following example shows a tunable description message:

```
# vmo -h lru_file_repage
Help for tunable lru_file_repage:
Purpose:
Specifies whether the repaging rate will be considered in determining
whether to steal file or computational pages.
Values:
        Default: 0
        Range: 0, 1
        Type: Dynamic
        Unit: boolean
Tuning:
A value of 0 indicates that only file pages will be stolen if file
pages are above minperm. Value of 1 indicates that only file pages will
be stolen if both the file repaging rate is higher than the
computational repaging rate and file pages are above minperm.
```

We recommend that AIX system administrators make a copy of the complete tunables description, using a text file format, to their personal computer if they need to work without an AIX server connection.

Tip: Appendix B, "Sample script for tunables" on page 429 provides a sample shell script named prt_tun_help.sh to output all tunables for each tuning command under a corresponding file with the "*xxx*o_help.txt" name. A tar archive format file, gathering all these output files, named prt_tun_help.tar, can be then uploaded.

6.2 Restricted tunables

Since AIX 5L V5.3, six tuning commands (**vmo**, **ioo**, **schedo**, **raso**, **no**, and **nfso**) have a unified behavior and syntax.

Beginning with AIX Version 6, some tunables are now classified as *restricted use tunables*. They exist and must be modified primarily for specialized intervention by the development support or development teams.

Note: System administrators should not modify *restricted tunables* unless instructed to by IBM Support professionals.

As these parameters are not recommended for user modification, they are no longer displayed by default, but can be displayed with the -F option (force). Thus, in SMIT and Web-based System Manager, they have no visibility by default.

The **no**, **nfso**, **vmo**, **ioo**, **raso**, and **schedo** tuning commands all support the following syntax:

```
command [-p|-r] [-F] -a
command [-L] -F [tunable]
command [-x] -F [tunable]
```

The -F option forces the display of restricted tunable parameters when the options -a, -L, or -x are specified alone on the command line to list all tunables. When -F is not specified, restricted tunables are not included in a display unless specifically named in association with a display option.

When the force -F option is used, the restricted tunables will be displayed after the non-restricted tunables and after a distinctive separator line beginning with the characters *"##"*. In English language locales, this will be *##Restricted tunables*. The Web-based Systems Manager panels do not show restricted tunables by default, but display them with their name followed by *(R),* when the Show Restricted Parameters check box is selected in the menu of a tunable table.

In Figure 6-1, note the restricted tunables are defined as *Development Parameters* to underline that only the IBM AIX development support team is authorized to modify the AIX restricted tunables.

```
                    Tuning Development Parameters

Move cursor to desired item and press Enter.

  Tuning Scheduler & Memory Load Control Development Parameters
  Tuning Virtual Memory Manager, File System and LVM Development Parameters
  Tuning Network Option Development Parameters
  Tuning NFS Option Development Parameters
  Tuning IO Development Parameters
  Tuning RAS Development Parameters

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

Figure 6-1 SMIT panel for AIX Version 6.1 restricted tunables

6.2.1 New warning message for restricted tunables

When a restricted tunable is modified using any -o, -d, or -D option, a warning message is written to stderr (without generating an error) to warn the user that a tunable of the restricted use type has been modified:

```
# vmo -o maxclient%=40
Setting maxclient% to 40
Warning: a restricted tunable has been modified
```

Moreover, if a restricted tunable is modified permanently adding the -p or -r option, the user will be prompted for confirmation of the change:

```
# vmo -p -o maxclient%=40
Modification to restricted tunable maxclient%, confirmation required
yes/no yes
Setting maxclient% to 40 in nextboot file
Setting maxclient% to 40
Warning: a restricted tunable has been modified
```

The saved restricted tunables that have been modified to a value different from the default value, are flagged with a comment *#RESTRICTED not at default value*, appended to the line of the **tunsave** output file.

The following is an example of a **tunsave** output file:

```
#vi /etc/mytuning.out
info:
        Description = "tunsave -F /etc/mytuning.out"
        AIX_level = "6.1.0.0"
        Kernel_type = "MP64"
        Last_validation = "2007-09-20 13:56:37 CDT (current, reboot)"
... (lines removed for clarity)
vmo:
... (lines removed for clarity)
        maxclient% = "40" # RESTRICTED not at default value
```

Current and default values for specific tunables

Although no tunables modification has been made within AIX V6, the **tunsave** command does report some restricted and non-restricted tunables as set without their default values:

```
... (lines removed for clarity)
vmo:
        kernel_heap_psize = "65536"
        kernel_psize = "65536" # RESTRICTED not at default value
        mbuf_heap_psize = "65536" # RESTRICTED not at default value
```

These tunables (for **vmo**: kernel_heap_psize, kernel_psize and mbuf_heap_psize) are reported due to their default ZERO value.

A ZERO default value for these tunables indicates that the operating system will determine and set the most appropriate value as the current one:

```
... (lines removed for clarity)
no:
        net_malloc_police = "16384"     # RESTRICTED not at default
value
```

The **no** tunable net_malloc_police is reported with the default value of 65536.

```
... (lines removed for clarity)
nfso:
        statd_debug_level = "-1"        # RESTRICTED not at default
value
        statd_max_threads = "-1"        # RESTRICTED not at default
value
```

The **nfso** tunables statd_debug_level and statd_max_threads are reported because their current values are set to -1 when statd subsystem is inactive, while their default value is 0 and 50, respectively. If statd is active, these two restricted tunables will not be reported.

6.2.2 New error log entry for restricted tunables

At system reboot, the presence of restricted tunables in the /etc/tunables/nextboot file that have been modified to a value different from their default value (by specifying the -r or -p options) causes the addition of a new TUNE_RESTRICTED error log entry.

This TUNE_RESTRICTED error log entry identifies the list of these modified restricted tunables.

This error log entry is created by calling a new performance tools specific **/usr/lib/perf/tunerrlog** command, which is included in the existing bos.perf.tune package.

The **/usr/sbin/tunrestore -R** command (in /etc/inittab file) calls the **tunerrlog** command, which adds these informational errors in the error log.

The following is an example of a TUNE_RESTRICTED error log message (Note that the date is using the Welsh locale from one of the author's test systems; we believe it was a Monday.):

```
----------------------------------------------------------------------
LABEL:          TUNE_RESTRICTED
IDENTIFIER:     D221BD55

Date/Time:       20 Medi 2007 19:35:36 CDT
Sequence Number: 19
Machine Id:      00C1F1704C00
Node Id:         lpar01
Class:           O
Type:            INFO
WPAR:            Global
Resource Name:   perftune
Description
RESTRICTED TUNABLES MODIFIED AT REBOOT

Probable Causes
SYSTEM TUNING

User Causes
```

```
TUNABLE PARAMETER OF TYPE RESTRICTED HAS BEEN MODIFIED

        Recommended Actions
        REVIEW TUNABLE LISTS IN DETAILED DATA

Detail Data
LIST OF TUNABLE COMMANDS CONTROLLING MODIFIED RESTRICTED TUNABLES AT
REBOOT, SEE FILE /etc/tunables/lastboot.log
vmo
```

6.2.3 AIX V6 tunables lists

In this section, the tunables and the restricted tunables reported by each of the AIX V6 tunables **xxo -F -a** commands are provided. As stated in 6.1, "Unique tunable documentation" on page 248, these lists are no longer available in AIX documentation or the man pages.

The settings of the tunables in the following lists are the default values for AIX V6.1 TL00 SP00.

IOO command tunables

With AIX V6.1 and above, only 21 **ioo** command tunables are available for user modification and 27 **ioo** command tunables are classified as restricted tunables and not available for system administrator modification.

- **ioo** command user tunables:

```
# ioo -a
                   aio_active = 0
                  aio_maxreqs = 65536
                aio_maxservers = 30
                aio_minservers = 3
         aio_server_inactivity = 300
         j2_atimeUpdateSymlink = 0
 j2_dynamicBufferPreallocation = 16
             j2_inodeCacheSize = 400
           j2_maxPageReadAhead = 128
             j2_maxRandomWrite = 0
          j2_metadataCacheSize = 400
           j2_minPageReadAhead = 2
j2_nPagesPerWriteBehindCluster = 32
             j2_nRandomCluster = 0
                    lvm_bufcnt = 9
                     pd_npages = 65536
              posix_aio_active = 0
```

```
            posix_aio_maxreqs = 65536
         posix_aio_maxservers = 30
         posix_aio_minservers = 3
  posix_aio_server_inactivity = 300
```

- **ioo** restricted tunables:

```
##Restricted tunables
                 aio_fastpath = 1
               aio_fsfastpath = 1
                aio_kprocprio = 39
             aio_multitidsusp = 1
              aio_sample_rate = 5
        aio_samples_per_cycle = 6
      j2_maxUsableMaxTransfer = 512
     j2_nBufferPerPagerDevice = 512
     j2_nonFatalCrashesSystem = 0
        j2_syncModifiedMapped = 1
      j2_syncdLogSyncInterval = 1
           jfs_clread_enabled = 0
            jfs_use_read_lock = 1
                   maxpgahead = 8
                   maxrandwrt = 0
                memory_frames = 262144
                   minpgahead = 2
                     numclust = 1
                    numfsbufs = 196
           pgahd_scale_thresh = 0
           posix_aio_fastpath = 1
         posix_aio_fsfastpath = 1
          posix_aio_kprocprio = 39
        posix_aio_sample_rate = 5
  posix_aio_samples_per_cycle = 6
                  pv_min_pbuf = 512
           sync_release_ilock = 0
```

vmo command tunables

With AIX V6.1, only 29 **vmo** tunables are available for user modification and 30 **vmo** tunables are classified as restricted tunables and not available for system administrator modification.

- **vmo** user tunables:

```
# vmo -a
   force_relalias_lite = 0
     kernel_heap_psize = 65536
          lgpg_regions = 0
```

```
                lgpg_size = 0
          low_ps_handling = 1
                  maxfree = 1088
                  maxperm = 210108
                   maxpin = 211796
                  maxpin% = 80
            memory_frames = 262144
            memplace_data = 2
     memplace_mapped_file = 2
   memplace_shm_anonymous = 2
       memplace_shm_named = 2
           memplace_stack = 2
            memplace_text = 2
   memplace_unmapped_file = 2
                  minfree = 960
                  minperm = 7003
                 minperm% = 3
                nokilluid = 0
                  npskill = 1024
                  npswarn = 4096
                numpsblks = 131072
          pinnable_frames = 180093
      relalias_percentage = 0
                    scrub = 0
                 v_pinshm = 0
         vmm_default_pspa = -1
```

- **vmo** restricted tunables:

```
##Restricted tunables
           cpu_scale_memp = 8
    data_stagger_interval = 161
                    defps = 1
                framesets = 2
                htabscale = n/a
             kernel_psize = 65536
     large_page_heap_size = 0
           lru_file_repage = 0
        lru_poll_interval = 10
                lrubucket = 131072
               maxclient% = 90
                 maxperm% = 90
          mbuf_heap_psize = 65536
          memory_affinity = 1
                npsrpgmax = 8192
                npsrpgmin = 6144
```

```
          npsscrubmax = 8192
          npsscrubmin = 6144
      num_spec_dataseg = 0
     page_steal_method = 1
             rpgclean = 0
           rpgcontrol = 2
           scrubclean = 0
  soft_min_lgpgs_vmpool = 0
      spec_dataseg_int = 512
      strict_maxclient = 1
        strict_maxperm = 0
   vm_modlist_threshold = -1
       vmm_fork_policy = 1
     vmm_mpsize_support = 2
```

no command tunables

With AIX V6.1, 133 **no** tunables are available for user modification and five **no** tunables are classified as restricted tunables and not available for system administrator modification.

- **no** user tunables:

```
# no -a
                 arpqsize = 12
               arpt_killc = 20
              arptab_bsiz = 7
                arptab_nb = 149
                bcastping = 0
      clean_partial_conns = 0
                 delayack = 0
            delayackports = {}
         dgd_packets_lost = 3
            dgd_ping_time = 5
           dgd_retry_time = 5
       directed_broadcast = 0
                 fasttimo = 200
        icmp6_errmsg_rate = 10
          icmpaddressmask = 0
ie5_old_multicast_mapping = 0
                   ifsize = 256
               ip6_defttl = 64
                ip6_prune = 1
            ip6forwarding = 0
       ip6srcrouteforward = 1
       ip_ifdelete_notify = 0
```

```
               ip_nfrag = 200
            ipforwarding = 0
              ipfragttl = 2
       ipignoreredirects = 0
              ipqmaxlen = 100
         ipsendredirects = 1
       ipsrcrouteforward = 1
          ipsrcrouterecv = 0
          ipsrcroutesend = 1
         llsleep_timeout = 3
                lo_perf = 1
              lowthresh = 90
               main_if6 = 0
             main_site6 = 0
               maxnip6q = 20
                 maxttl = 255
              medthresh = 95
             mpr_policy = 1
            multi_homed = 1
              nbc_limit = 131072
          nbc_max_cache = 131072
          nbc_min_cache = 1
       nbc_ofile_hashsz = 12841
               nbc_pseg = 0
         nbc_pseg_limit = 262144
         ndd_event_name = {all}
      ndd_event_tracing = 0
          ndp_mmaxtries = 3
          ndp_umaxtries = 3
               ndpqsize = 50
              ndpt_down = 3
              ndpt_keep = 120
             ndpt_probe = 5
         ndpt_reachable = 30
           ndpt_retrans = 1
           net_buf_size = {all}
           net_buf_type = {all}
   net_malloc_frag_mask = {0}
      netm_page_promote = 1
         nonlocsrcroute = 0
               nstrpush = 8
            passive_dgd = 0
       pmtu_default_age = 10
            pmtu_expire = 10
pmtu_rediscover_interval = 30
```

```
             psebufcalls = 20
                psecache = 1
               psetimers = 20
          rfc1122addrchk = 0
                 rfc1323 = 0
                 rfc2414 = 1
            route_expire = 1
         routerevalidate = 0
                rto_high = 64
              rto_length = 13
               rto_limit = 7
                 rto_low = 1
                    sack = 0
                  sb_max = 1048576
      send_file_duration = 300
             site6_index = 0
              sockthresh = 85
                 sodebug = 0
             sodebug_env = 0
               somaxconn = 1024
                strctlsz = 1024
                strmsgsz = 0
               strthresh = 85
              strturncnt = 15
         subnetsarelocal = 1
      tcp_bad_port_limit = 0
                 tcp_ecn = 0
      tcp_ephemeral_high = 65535
       tcp_ephemeral_low = 32768
            tcp_finwait2 = 1200
          tcp_icmpsecure = 0
         tcp_init_window = 0
   tcp_inpcb_hashtab_siz = 24499
             tcp_keepcnt = 8
            tcp_keepidle = 14400
            tcp_keepinit = 150
           tcp_keepintvl = 150
    tcp_limited_transmit = 1
             tcp_low_rto = 0
            tcp_maxburst = 0
             tcp_mssdflt = 1460
         tcp_nagle_limit = 65535
       tcp_nagleoverride = 0
              tcp_ndebug = 100
             tcp_newreno = 1
```

```
         tcp_nodelayack = 0
      tcp_pmtu_discover = 1
          tcp_recvspace = 16384
          tcp_sendspace = 16384
          tcp_tcpsecure = 0
           tcp_timewait = 1
                tcp_ttl = 60
         tcprexmtthresh = 3
                thewall = 524288
        timer_wheel_tick = 0
              tn_filter = 1
     udp_bad_port_limit = 0
     udp_ephemeral_high = 65535
      udp_ephemeral_low = 32768
  udp_inpcb_hashtab_siz = 24499
      udp_pmtu_discover = 1
          udp_recvspace = 42080
          udp_sendspace = 9216
                udp_ttl = 30
               udpcksum = 1
         use_sndbufpool = 1
```

- **no** restricted tunables:

```
##Restricted tunables
         extendednetstats = 0
          inet_stack_size = 16
        net_malloc_police = 16384
             pseintrstack = 24576
                 use_isno = 1
```

schedo command tunables

With AIX V6.1, only 15 **schedo** tunables are available for user modification and 27 **schedo** tunables are classified as restricted tunables and not available for system administrator modification.

- **schedo** user tunables:

```
# schedo -a
         affinity_lim = 7
         big_tick_size = 1
  ded_cpu_donate_thresh = 80
       fixed_pri_global = 0
              force_grq = 0
                maxspin = 16384
               pacefork = 10
                sched_D = 16
```

```
             sched_R = 16
        tb_balance_S0 = 2
        tb_balance_S1 = 2
         tb_threshold = 100
            timeslice = 1
      vpm_fold_policy = 1
           vpm_xvcpus = 0
```

- **schedo** restricted tunables:

```
##Restricted tunables
              %usDelta = 100
        allowMCMmigrate = 0
            fast_locks = 0
        hotlocks_enable = 0
idle_migration_barrier = 4
     krlock_confer2self = 1
   krlock_conferb4alloc = 1
          krlock_enable = 1
     krlock_spinb4alloc = 1
    krlock_spinb4confer = 1024
     n_idle_loop_vlopri = 100
  search_globalrq_mload = 256
   search_smtrunq_mload = 256
   setnewrq_sidle_mload = 384
    shed_primrunq_mload = 64
     sidle_S1runq_mload = 64
     sidle_S2runq_mload = 134
     sidle_S3runq_mload = 134
     sidle_S4runq_mload = 4294967040
     slock_spinb4confer = 1024
       smt_snooze_delay = 0
      smtrunq_load_diff = 2
          v_exempt_secs = 2
          v_min_process = 2
            v_repage_hi = 0
          v_repage_proc = 4
             v_sec_wait = 1
```

nfso command tunables

With AIX V6.1, only 13 **nfso** tunables are available for user modification and 21 **nfso** tunables are classified as restricted tunables and not available for system administrator modification.

- **nfso** user tunables:

```
# nfso -a
           client_delegation = 1
           nfs_max_read_size = 65536
          nfs_max_write_size = 65536
                 nfs_rfc1323 = 1
   nfs_securenfs_authtimeout = 0
    nfs_server_base_priority = 0
           nfs_server_clread = 1
      nfs_use_reserved_ports = 0
   nfs_v3_server_readdirplus = 1
    nfs_v4_fail_over_timeout = 0
                   portcheck = 0
           server_delegation = 1
             utf8_validation = 1
```

- **nfso** restricted tunables:

```
##Restricted tunables
           lockd_debug_level = 0
       nfs_allow_all_signals = 0
        nfs_auto_rbr_trigger = 0
         nfs_dynamic_retrans = 1
        nfs_gather_threshold = 4096
            nfs_iopace_pages = 0
             nfs_max_threads = 3891
         nfs_repeat_messages = 0
              nfs_socketsize = 600000
nfs_tcp_duplicate_cache_size = 5000
          nfs_tcp_socketsize = 600000
nfs_udp_duplicate_cache_size = 5000
                 nfs_v2_pdts = 1
                 nfs_v3_pdts = 1
                 nfs_v4_pdts = 1
              nfs_v2_vm_bufs = 10000
              nfs_v3_vm_bufs = 10000
              nfs_v4_vm_bufs = 10000
           statd_debug_level = 0
           statd_max_threads = 50
                 udpchecksum = 1
```

raso command tunables

With AIX V6.1, nine **raso** tunables are available for user modification and four **raso** tunables are classified as restricted tunables and not available for system administrator modification.

- **raso** user tunables:

```
# raso -a
    kern_heap_noexec = 0
       kernel_noexec = 1
    mbuf_heap_noexec = 0
  mtrc_commonbufsize = 971
        mtrc_enabled = 1
    mtrc_rarebufsize = 50
      tprof_cyc_mult = 1
      tprof_evt_mult = 1
tprof_inst_threshold = 1000
```

- **raso** restricted tunables:

```
##Restricted tunables
recovery_action,1,1,1,0,1,boolean,D,
recovery_average_threshold,5,5,5,0,100,numeric,D,
recovery_debugger,0,0,0,-1,3,numeric,D,
recovery_framework,1,1,1,0,1,boolean,B,
```

6.3 AIX V6 out-of-the-box performance

There have been recurring performance issues seen at enterprise sites that have been resolved through simple modification of AIX tunable values. A significant percentage of these performance issues have occurred in environments running databases on file systems.

For example, on AIX 5L V5.2 and V5.3, Oracle® customers must perform the following tuning steps:

- VMM tuning
  - Reduce minperm, maxperm, and maxclient.
  - Turn off strict_maxclient.
  - Increase minfree and maxfree.
- AIO tuning
  - Enable AIO.
  - Tune minservers and maxservers, and then reboot for them to take effect.

- Oracle tuning
  - Enable CIO.

Another common issue for AIX users is interactive applications that become unresponsive due to other applications on the same system doing large sequential writes to slow storage devices.

In AIX V6, the default settings have been modified accordingly, resulting in a better *out-of-the-box* performance for a majority of our AIX systems.

The following sections explained in detail which tunables have been modified, and provide a side-by-side comparison of their default values for AIX V5 and AIX V6.

6.3.1 Virtual Memory Manager default tunables

A common problem seen in file server environments is system paging when no VMM tuning has been done. File system intensive applications, such as a database server, mail servers, or backup servers, often page out computational pages to the paging space even though the system has enough real memory.

A system has enough memory when its amount of virtual memory does not exceed the amount of physical memory.

In the following example, the amount of virtual memory is 163967 4 KB pages while the amount of physical memory is 262144 4 KB pages; the system has enough real memory:

```
# svmon -G
               size       inuse        free         pin     virtual
memory       262144      226791       35353       96991      163967
pg space     131072        2370

               work        pers        clnt       other
pin           86669           0           0       10322
in use       163967           0       62824

PageSize   PoolSize       inuse        pgsp         pin     virtual
s    4 KB         -      130903        2370       29999       68079
m   64 KB         -        5993           0        4187        5993
```

The cause is that the percentage of memory that is being used for caching persistent or client pages typically is between the minperm% value and maxperm% value or maxclient% value, respectively. Then, the page replacement algorithm steals computational pages when the repage count for computational pages is greater than the repage count for file pages.

The solution is to turn off the repage ratio check lru_file_repage. The lru_file_repage parameter was introduced in ML4 of AIX 5L V5.2 and ML1 of AIX 5L V5.3, but disabled by default.

The following change (lru_file_repage=0) turns off the repage ratio check and forces the page replacement algorithm to steal computational pages only when the percentage of cached file pages is less than the minperm% value.

Thus, the VMM page replacement default is changed with AIX Version 6 to allow AIX to use up to 90% of its real memory for file caching, but favor computational pages as resident pages over file pages.

In addition, the default for minperm% is reduced to 3%. Computational pages will not be stolen unless the amount of active virtual memory exceeds 97% of the size of the real memory. Also, list-based LRU will be enabled by default.

Table 6-1 provides a list of the **vmo** command tunable names and their default values within AIX releases.

Table 6-1 Default tunable values for the vmo command

vmo tunable name	AIX 5L V5.2/V5.3 default values	AIX V6 default values
minperm%	20	3
maxperm% (R)[a]	80	90
maxclient% (R)[a]	80	90
lru_file_repage (R)[a]	1	0
page_steal_method (R)[a]	0	1

a. (R) means that it is a restricted use tunable.

6.3.2 AIX V6 enables I/O pacing by default

Large sequential writes can often cause a system to become unresponsive.

One of the potential causes is that pages that can be used for file caching are modified, and therefore VMM page replacement cannot find any candidate pages to steal. The reason all the pages are in a modified state may be that file pages are being created faster than can be written to disk, either due to extremely fast CPUs, a slow storage subsystem, or both.

The settings of I/O pacing for AIX V4.3 and AIX 5L V5 were defined where a server consisted of a 100 MHz uniprocessor, a 10 Mb Ethernet card, and a 2 GB SCSI disk. These settings are no longer suitable due to the huge performance improvement that our latest processors deliver.

The VMM file I/O pacing will be enabled by default to prevent unresponsive system behavior due to a large number of queued I/Os on the paging device tables.

To enable I/O pacing by default, the minpout and maxpout tunables of sys0 device are set to non-zero values. The new AIX V6 specific default values are based on results of tests and analysis by the IBM performance team.

These values are expected to allow an application run on a system with enough CPU horsepower and storage subsystem bandwidth while not severely impacting the response time of interactive applications to the point where the system seems non-responsive.

Table 6-2 provides minpout/maxpout values within AIX releases.

Table 6-2 minpout/maxpout values within AIX releases

sys0 tunable name	AIX 5L V5.2/V5.3 default values	AIX V6 default values
minpout	0	4096
maxpout	0	8193

6.3.3 AIX V6 new AIO dynamic tunables

AIO stands for Asynchronous Input Output. It is a software subsystem within AIX that allows a process to issue an I/O operation and continue processing without waiting for the I/O to finish.

Therefore, asynchronous I/O operations run in the background and do not block user applications. This improves performance because I/O operations and applications processing can run simultaneously. Many applications, such as databases and file servers, take advantage of the ability to overlap processing and I/O.

There are two AIO subsystems:

- The original AIX AIO, now called *LEGACY* AIO.
- The Portable Operating System Interface (POSIX) compliant AIO, called POSIX AIO.

The major differences between the two involve different parameter passing at the application layer. So, the choice for using one or the other implementation in an application is a software developer decision. The AIO application programming interface is not covered in this section.

Both subsystems can run concurrently on AIX.

With AIX Version 4/Version 5, if an application uses AIO, the corresponding subsystem must be activated by setting *available* in the autoconfig parameter. This requires a reboot because the AIO kernel extension has to be loaded.

Prior to AIX 5L V5.3 TL05, any change to the three tunables, maxreqs, maxservers, and minservers, required a reboot.

AIO tunables for both subsystems in AIX 5L V5.3 are:

```
# oslevel -s
5300-06-01-0000
# lsattr -El aio0
autoconfig defined STATE to be configured at system restart True
fastpath   enable  State of fast path                       True
kprocprio  39      Server PRIORITY                          True
maxreqs    4096    Maximum number of REQUESTS               True
maxservers 10      MAXIMUM number of servers per cpu        True
minservers 1       MINIMUM number of servers                True
# lsattr -El posix_aio0
autoconfig defined STATE to be configured at system restart True
fastpath   enable  State of fast path                       True
kprocprio  39      Server PRIORITY                          True
maxreqs    4096    Maximum number of REQUESTS               True
maxservers 10      MAXIMUM number of servers per cpu        True
minservers 1       MINIMUM number of servers                True
```

With AIX 5L V5.3 TL05, a new **aioo** command was shipped with the AIO fileset (bos.rte.aio) that changes these three tunables (minservers, maxservers, and maxreqs) on a running system. It requires no reboot when you increase maxreqs, maxservers, and minservers, but reducing the values for these tunables does require a reboot. The **aioo** command does not change the ODM attributes to make the changes persistent across boots.

Here is an example of AIX 5L V5.3 TL05 **aioo** command output:

```
# aioo -a
               minservers = 1
               maxservers = 10
                  maxreqs = 4096
               fsfastpath = 0
```

This **aioo** output shows a new AIX 5L V5.3 TL05 tunable, fsfastpath, that is non-persistent across boots.

With AIX Version 6, the tunables fastpath and fsfastpath are classified as restricted tunables[1] and are now set to a value of 1 by default:

- When the fastpath tunable is set to 1, asynchronous I/O requests to a raw logical volume are passed directly to the disk layer using the corresponding strategy routine.
- When the fsfastpath tunable is set to 1, asynchronous I/O requests for files opened with Concurrent I/O (CIO) mode in a JFS2 file system AIO are passed directly to LVM or disk using the corresponding strategy routine.

The following output shows the restricted tunables list of both AIO subsystems:

```
# ioo -F -a
... (lines removed for clarity)
##Restricted tunables
                  aio_fastpath = 1
                aio_fsfastpath = 1
                 aio_kprocprio = 39
              aio_multitidsusp = 1
               aio_sample_rate = 5
         aio_samples_per_cycle = 6
... (lines removed for clarity)
            posix_aio_fastpath = 1
          posix_aio_fsfastpath = 1
           posix_aio_kprocprio = 39
         posix_aio_sample_rate = 5
   posix_aio_samples_per_cycle = 6
                   pv_min_pbuf = 512
            sync_release_ilock = 0
```

In AIX Version 6, both AIO subsystems are loaded by default but not activated; no AIO servers are started at AIX boot time. The AIO servers are automatically started when applications are initiating AIO I/O requests. They stay active as long as they service AIO I/O requests.

There are no more AIO devices in ODM and all their parameters now become tunables using the **ioo** command. The newer **aioo** command is removed.

[1] For more about restricted tunables, see 6.2, "Restricted tunables" on page 249.

The following are the key points of this change in more detail:

- Under the **kdb** command, the lke subcommand shows aio subsystems extensions are loaded by default:

```
# kdb
... (lines removed for clarity)
(0)> lke | grep aio
  8 F10006C001C92F00 F1000000906A1000 00005000 00090242
/usr/lib/drivers/posix_aiopin
  9 F10006C001C92E00 F10000009068B000 00016000 00090252
/usr/lib/drivers/posix_aio.ext
 11 F10006C001C92D00 F100000090683000 00005000 00090242
/usr/lib/drivers/aiopin
 12 F10006C001C92C00 F100000090671000 00012000 00090252
/usr/lib/drivers/aio.ext
(0)>
```

- The AIX V6 **ioo** command has two new aio_active and posix_aio_active parameters. These parameters are static and can be changed only by AIX. These aio_active or posix_aio_active parameters are set to 1 when the corresponding AIO kernel extension has been used and pinned.

```
# ioo -a | grep active
                    aio_active = 0
              posix_aio_active = 0
```

- No AIO servers are started by default. The name of the kernel process managing the AIO subsystem is “aio*L*pool” for *L*egacy and “aio*P*pool” for *P*osix:

```
# pstat -a | grep aio
 22 a   16060      1  16060     0     0     1  aioPpool
 28 a   1c08a      1  1c08a     0     0     1  aioLpool
# ps -k | grep aio
  90208      -  0:00 aioPpool
 114826      -  0:00 aioLpool
```

- In AIX Version 6, AIO subsystems are no longer devices in the ODM:

```
# oslevel -s
6100-00-00
# lsattr -El aio0
lsattr: 0514-519 The following device was not found in the
customized
        device configuration database:
        aio0
# lsattr -El posix_aio0
```

```
lsattr: 0514-519 The following device was not found in the
customized
        device configuration database:
        posix_aio0
```

- In AIX Version 6, all AIO subsystems parameters become the **ioo** command tunables:

```
# ioo -a | grep aio
                    aio_active = 0
                   aio_maxreqs = 65536
                aio_maxservers = 30
                aio_minservers = 3
         aio_server_inactivity = 300
              posix_aio_active = 0
             posix_aio_maxreqs = 65536
          posix_aio_maxservers = 30
          posix_aio_minservers = 3
   posix_aio_server_inactivity = 300
```

- In AIX Version 6, the **aioo** command is removed:

```
# aioo
ksh: aioo:  not found.
# man aioo
Manual entry for aioo not found or not inst
```

AIO servers are started and stay active as long as they service I/O requests. A new tunable, server_inactivity (posix_aio_server_inactivity or aio_server_inactivity), is added to the **ioo** command, and controls how long in seconds an AIO server sleeps waiting for work. If no work is received during the sleep period, the AIO server exits. Both posix_aio_server_inactivity tunable and aio_server_inactivity tunable are not restricted tunables.

The main benefit, at the AIX layer, is to free pinned memory and decrease the number of processes after a period of peak workload activity within the AIO subsystem, which helps lighten the load process scheduling and reduces system resource usage. The minservers tunable becomes an active floor and indicates the number of servers that stay available to service I/O requests.

Note: The default minservers tunable value is 3 and becomes a per-CPU tunable.

The number of active servers stays between the minservers and maxservers values, depending on the number of concurrent I/O requests to service. That is why value changes to minservers and maxservers do not result in a synchronous change in the number of available servers in the system.

The server_inactivity tunable (the number of seconds an AIO server sleeps) can be changed at anytime to any valid value. The servers that are already sleeping with the old time value will continue to sleep for the old time value. Any servers going to sleep after the value is changed will use the new value.

The maxreqs tunable controls the number of requests the AIO subsystem allows. This includes the I/O operations in flight and ones queued for the slow path waiting on servers.

> **Note:** The default maxreqs tunable value is 65536 in AIX Version 6.

There are other AIO tunables, but their use is restricted and should only be changed under the recommendation of a performance support specialist.

Table 6-3 details the values range for each AIO subsystem tunables.

Table 6-3 Values range for each AIO subsystem tunables

Tunable name	Restricted	Type	Default	Minimum	Maximum
fastpath	Yes	Boolean	On		
fsfastpath	Yes	Boolean	On		
kprocprio	Yes	Value	39	0	254
multitidsusp	Yes	Boolean	On		
sample_rate	Yes	Value	5	1	86,400
samples_per_cycle	Yes	Value	6	1	131,072
maxreqs	No	Value	65,536	AIO_MAX	1,048,576
maxservers	No	Value	30	1	20,000
minservers	No	Value	3	0	maxservers
server_inactivity	No	Value	300	1	86,400
active	Read-Only	Boolean			

6.3.4 NFS default tunables

With AIX Version 6, RFC 1323 on TCP/IP stack is enabled by default, as the default read/write size is increased to 64 KB for TCP connections. This allows TCP connections to use the TCP scaling window for NFS client server connections.

The default number of the biod daemon has also increased to 32 biod daemons per NFS V3 mount point.

6.4 Hardware performance monitors

To advance the state of high performance computing offerings for IBM clients in computationally intensive industries, including automotive, aerospace, petroleum, meteorology, and life science, the design of POWER processors has extra hardware components inserted into each processor to count specific performance processor metrics.

These hardware counters are non intrusive, very accurate, and are specific for each processor generation.

Since the POWER3™ processor, AIX provides a suite of performance-related tools and libraries to assist in application tuning by gathering these low-level metrics that are critical to performance on IBM server offerings, including System p, System x™, and Blue Gene® systems running both AIX and Linux.

This performance metrics subsystem is divided into two layers:

- Performance Monitor (PM

  The Performance Monitor provides a service to read these hardware counter registers, and defines several 64-bit context counters (thread context and process context, to name two) to obtain the metrics of a specific process tree instead of all activities on each processor.

- Hardware Performance Monitor (HPM)

  HPM is able to gather for an application the usual timing information, as well as critical hardware performance metrics reported by the PM layer, such as the number of misses on all cache levels, the number of floating point instructions executed, and the number of instruction loads that cause TLB misses. These help the algorithm designer or programmer identify performance issues.

The PM, HPM tools, and APIs are provided by the AIX bos.pmapi.tools fileset.

6.4.1 Performance Monitor (PM)

The performance monitor consists of:

- Three AIX commands:
  - The **pmctl** command, whose description is not included in the AIX documentation, is found in the /usr/pmapi/tools directory. Like other commands in AIX Version 6, the **help** command panel is displayed using the -h flag. This command controls the state of the PMAPI subsystem, and the hardware events that are profiled:

```
# pwd
/usr/pmapi/tools
# ls
hpmcount  hpmstat   pmctl     pmcycles  pmlist
```

  - The **pmcycles** command, which returns the processor clock and decrementer speeds:

```
# pmcycles -m
CPU 0 runs at 4208 MHz
CPU 1 runs at 4208 MHz
# pmcycles -d
This machine runs at 4208 MHz
The decrementer runs at 512.0 MHz (1.95 ns per tick)
```

  - The **pmlist** command, which lists information about supported processors, and displays information about processor clocking, events, events groups and sets, and derived metrics. The following example shows the number of hardware counters and associated profiled events for some selected processors:

```
# pmlist -p POWER3 -c -1 | grep Counter
==== Counter 1, #events: 51
==== Counter 2, #events: 40
==== Counter 3, #events: 31
==== Counter 4, #events: 32
==== Counter 5, #events: 26
==== Counter 6, #events: 23
==== Counter 7, #events: 24
==== Counter 8, #events: 16
# pmlist -p RS64-II -c -1 | grep Counter
==== Counter 1, #events: 114
==== Counter 2, #events: 32
==== Counter 3, #events: 32
==== Counter 4, #events: 32
==== Counter 5, #events: 22
==== Counter 6, #events: 18
```

```
==== Counter 7, #events: 18
==== Counter 8, #events: 16
# pmlist -p POWER5 -c -1 | grep Counter
==== Counter 1, #events: 214
==== Counter 2, #events: 206
==== Counter 3, #events: 193
==== Counter 4, #events: 197
==== Counter 5, #events: 1
==== Counter 6, #events: 1
# pmlist -p POWER6 -c -1 | grep Counter
==== Counter 1, #events: 355
==== Counter 2, #events: 365
==== Counter 3, #events: 347
==== Counter 4, #events: 348
==== Counter 5, #events: 1
==== Counter 6, #events: 1
```

- The libpm.a instrumentation library is a low-level application programming interface providing a service to read hardware counters registers, and several 64-bit context counters (thread context, process context, and so on). Both 32-bit and 64-bit applications are supported, as long all modules are compiled in one of the two modes. The following libraries content show both 32-bit and 64-bit library modules:

```
# ar -t -X32_64 libpmapi.a
shr.o
shr_64.o
```

Note: When using AIX V6.1 and subsequent releases, the following AIX command returns the processor speed in hertz (Hz):

```
# lsattr -El proc0 | grep frequency
frequency   4208000000     Processor Speed       False
```

6.4.2 Hardware Performance Monitor (HPM)

The hardware performance monitor consists of:

- Two tools or AIX commands:
  - A **hpmcount** utility, which starts an application and provides, at the end of execution, wall-clock time, hardware performance counters information, derived hardware metrics, and resource utilization statistics.
  - A **hpmstat** utility to collect system level hardware performance counters information.

- An libhpm.a instrumentation library (or the thread safe version libhpm_r.a for threaded applications). The hpm libraries are higher-level instrumentation libraries based on the pmapi library and libm library. Therefore, the -lpmapi -lm library references must be specified when compiling applications using hpm libraries. Both 32-bit and 64-bit applications are supported, as long as all modules are compiled in one of the two modes. The following libraries show both 32-bit and 64-bit library modules:

```
# ar -t -X32_64 libhpm.a
shr.o
shr_64.o
# ar -t -X32_64 libhpm_r.a
shr.o
shr_64.o
```

6.4.3 AIX V6.1 PM and HPM enhancements

In this section, only the major user enhancements to PM and HPM toolkits are described.

For more detailed information about the Performance Monitoring API enhancements, like the PMAPI subroutines description, review *AIX Version 6.1 Performance Tools Guide and Reference*, SC23-5254.

Enhancing tracing performance

With AIX V6.1, the trace system of the PMAPI library pmsvcs is now implemented with the AIX Component Trace system in standard mode. In previous AIX versions, the trace system was activated at compilation time, setting the DEBUG flag and implemented through a collection of printf() instructions.

One drawback of the previous PMAPI trace system is that when the trace system is compiled, information is output using the printf() routine that involves CPU resources. The goal of any software instrumentation is to minimize its resource usage, as it is not possible to eliminate the necessary tooling required. With these enhanced PMAPI and HPM toolkits, the performance is as close as possible to the current instrumented software.

One additional benefit of this enhancement is to avoid re-compiling software to switch on/off the PMAPI trace system. The **ctctrl** command is now able to switch on/off the PMAPI trace system.

The following example shows how to switch on/off the PMAPI subsystem through the AIX Component Trace system:

```
# ctctrl -q
----------------------------------------+-------+-------+-------+---------------
                                        | Have  |Mem Trc|Sys Trc| Buffer size
     Component name                     | alias | /level| /level| /Allocated
----------------------------------------+-------+-------+-------+---------------
aio                                     |   NO  | OFF/3 |  ON/3 |         0/ NO
dump
...(lines missing for clarity)
pmsvcs                                  |   NO  | OFF/3 |  ON/3 |         0/ NO
# ctctrl memtraceon -c pmsvcs
# ctctrl -q -c pmsvcs
----------------------------------------+-------+-------+-------+---------------
                                        | Have  |Mem Trc|Sys Trc| Buffer size
     Component name                     | alias | /level| /level| /Allocated
----------------------------------------+-------+-------+-------+---------------
pmsvcs                                  |   NO  |  ON/3 |  ON/3 |         0/ NO
# ctctrl systraceoff -c pmsvcs
# ctctrl memtraceoff -c pmsvcs
# ctctrl -q -c pmsvcs
----------------------------------------+-------+-------+-------+---------------
                                        | Have  |Mem Trc|Sys Trc| Buffer size
     Component name                     | alias | /level| /level| /Allocated
----------------------------------------+-------+-------+-------+---------------
pmsvcs                                  |   NO  | OFF/3 | OFF/3 |         0/ NO
# ctctrl systraceoff -c pmsvcs
```

While running the pmsvcs trace system, high amounts of trace launched per seconds can occur (the value of 40,000 traces have been reached during internal tests).

Timing with nanosecond granularity

With AIX V6.1, to acquire accurate timing information, HPM now relies on a libpmapi call where timing information is returned in a timebasestruct_t with nanosecond granularity. This new call has also less performance restrictions. In previous versions, HPM was using gettimeofday() with a microsecond granularity.

Time counter or time measurement

The execution time of a program is the CPU time, which means the amount of time a program uses the CPU.

This amount of time measured, counting the number of execution dispatches that are accumulated in a so-called time counter, is usually implemented as a register in POWER processors. The values of these time counters, real events counters, are converted to time using the time_base_to_time subroutine.

A new option (-b time | purr | spurr) is added to **hpmcount** and **hpmstat** commands to select a time base nomalization for time collected data based on the purr register or spurr register when available, depending on processor type. By default, the **hpmcount** and **hpmstat** report time counter is based on the timebase register, as in the previous AIX versions:

```
hpmcount [-b time | purr | spurr]
hpmstat [-b time | purr | spurr]
```

The time-base register is incremented each time a thread is dispatched for execution on the processor or core.

The purr register

The Processor Utilization Resource Register (purr) is what counts every time the hardware thread is dispatched for execution on the processor. As POWER5™ and POWER6 processors supported two hardware threads, there are two purr registers per processor (or core). The sum of the two purr registers is equal to all the times a thread was dispatched, which is the time base register.

The spurr register

The Scaled Performance Utilization Resources Register (spurr) is new on POWER6. This results from the electrical power and thermal dissipation management technology introduced as part of the POWER6 system design.

This energy management done by the chip allows it to throttle the fetch and dispatch bandwidth to keep power down, increasing the cycles per instruction (CPI). As the cycles per instruction increase, the instruction takes more time and the execution flow decreases. Thus, the processor activity is slower and therefore cooler. The spurr value is similar to the purr value, except that the spurr value scales as a function of the degree of processor throttling. Spurr values are proportional to the fetch or the instruction dispatch rate of the processor.

Measuring time-base data based on purr or spurr registers provides a more accurate measurement to **hpmcount** and **hpmstat** instrumentation tools on servers based on POWER5 and POWER6 processors.

A new variable environment, HPM_NORMALIZE, switches **hpmstat** and **hpmcount** command reports from timebase to purr or spurr normalization. This avoids re-write procedures and scripts taking advantage of this new option. The option -b takes precedence over this variable:

```
HPM_NORMALIZE=[time][purr][spurr]
```

HPM data post processing

With AIX Version 6, the **hpmstat** and **hpmcount** commands produce results in the XML format output file using the new -x option:

```
hpmstat -o <file> -x
hpmstat -o <file> -x
```

This XML output file format allows post-processing by the Visual Performance Analyzer (VPA). VPA is an Eclipse-based visual performance toolkit that runs on Windows®, AIX, and Linux. It is proposed to IBM clients as an alphaWorks® project at:

http://www.alphaworks.ibm.com.tech/vpa

7

Networking

AIX Version 6 provides updates and new networking features that are covered in the following sections:

7.1 Internet Group Management Protocol Version 3

Internet Group Management Protocol (IGMP) is the protocol used by hosts and multicast routers to establish multicast group memberships within a physical network. It allows hosts to participate in IP multicasting according to RFC 1112, where the basics and specifics are described. A sender does not have to be a member of a multicast group to send packets to such a group. The IGMP protocol allows routers to act as members of one or more multicast groups, performing both the *multicast router part* and *group member part* of the protocol.

AIX V6.1 provides the host side function and group member part of the IGMP Version 3 (IGMPv3) protocol. The AIX V6.1 IGMPv3 implementation adheres to RFC 3376 and includes the new Socket Interface Extensions for Multicast Source Filters.

The AIX V6.1 IGMPv3 implementation allows backward compatibility with the previous two versions of the protocol, IGMP version 1 (IGMPv1) and IGMP version 2 (IGMPv2), as they are supported in AIX 5L releases.

IGMPv1 allows hosts to join multicast groups. In this version, there are no leave messages. Routers use a timeout based mechanism to discover which hosts dropped their membership. Routers periodically send host membership queries to the all-hosts group. Each host starts a random delay timer before issuing a host membership report on the interface where they receive the query. Once the smaller timer expires, this host sends a report that all the other hosts receive, causing their timers to stop, since only one report is needed by the router for that group in the sub net.

In IGMPv2, leave messages were added to reduce the bandwidth wasted during the leave latency period. A host leaves a group by sending a leave message to the all-routers group (IP address 224.0.0.2). When the router receives a leave message, it sends a group-specific query to the multicast group that is being left and not to the all-hosts group as in IGMPv1.

IGMPv3 allows hosts to specify a list of sources from which they do not want to receive traffic, blocking any host that is in the list. On the other hand, it also allows a host to specify a list of sources from which they want to receive traffic only. In other words, it allows for source filtering, that is, receive packets only from specific source addresses, or from all but specific source addresses. IGMPv3 protocol sets the standards on how this information is transmitted across hosts and routers and the relevant messages are transmitted in IP datagrams with a protocol number of 2 (IPPROTO_IGMP).

IGMPv3 allows finer control over the multicast packets forwarded to the subnetwork and may conserve link capacity, especially when a system switches from receiving one multicast group to another.

The AIX V6.1 IGMPv3 protocol implementation has two distinct multicast modes:

Any-source multicast
: All sources are accepted by default, and any unwanted source is turned off and back on as needed. This is also called *exclude* mode.

Source-specific multicast
: Only sources in a given definition list are allowed. This is also called *include* mode.

According to the previously mentioned multicast modes, IGMPv3 capable hosts have the ability to set source filters and configure multicast groups by using the following socket options:

IP_ADD_MEMBERSHIP
: This option is used to request that the host joins an any-source multicast group.

IP_DROP_MEMBERSHIP
: This option is used to leave an already joined multicast group.

IP_BLOCK_SOURCE
: This option is used to block data from a given source to a multicast group and refers to the any-source multicast implementation.

IP_UNBLOCK_SOURCE
: This option is used to unblock a previously blocked source address and refers to the any-source multicast implementation.

IP_ADD_SOURCE_MEMBERSHIP
: This option is used to add the membership as well as to allow data from the given source address to the given multicast group. The source-specific multicast implementation is facilitated by this option.

IP_DROP_SOURCE_MEMBERSHIP
: This option is used to remove a source address from the list of included addresses. The source-specific multicast implementation is facilitated by this option.

The setsockopt() system call has to be utilized to set the new options associated with a socket. Hence, this subroutine provides an application with the means to include or exclude source-specific addresses for each multicast group.

Note, that the first two options previously listed, IP_ADD_MEMBERSHIP and IP_DROP_MEMBERSHIP, are also available in IGMPv2, but the remaining four options are only provided through the IGMPv3 protocol implementation.

There are four socket options exclusive to IGMPv3:

- IP_BLOCK_SOURCE
- IP_UNBLOCK_SOURCE
- IP_ADD_SOURCE_MEMBERSHIP
- IP_DROP_SOURCE_MEMBERSHIP

They require you to use the new ip_mreq_source structure. This structure is similar to the traditional ip_mreq structure, but it contains the new variable imr_sourceaddr to pass the source address for source filtering through the setsockopt system call. The ip_mreq_source structure is defined in the /usr/include/netinet/in.h header file as follows:

```
struct ip_mreq_source {
        struct in_addr  imr_multiaddr;  /* IP multicast address of group */
        struct in_addr  imr_sourceaddr; /* IP address of source */
        struct in_addr  imr_interface;  /* local IP address of interface */
};
```

RFC 3376 can be referenced to understand how the multicast reception state is maintained by systems at the socket and the interface layers.

When a multicast packet arrives at the IP layer, the interface reception state is looked up before accepting/dropping the packet. After the packet is accepted by the IP layer and passed up to the UDP layer, the socket reception state is looked up before the packet is delivered on the socket's receive buffer.

Filtering of packets based upon a socket's multicast reception state is a new feature of IGMPv3. The previous protocols [RFC1112] described no filtering based upon the multicast join state; rather, a join on a socket simply caused the host to join a group on the given interface, and packets destined for that group could be delivered to all sockets, whether they had joined or not.

7.2 Network Data Administration Facility enhancements

Network Data Administration Facility (NDAF) is a component introduced in AIX 5L. In AIX V6.1, it is enhanced. For basic information about NDAF itself, see *AIX 5L Differences Guide Version 5.3 Addendum*, SG24-7414.

The additional enhancements discussed are as follows:

- Integration of NDAF to the base AIX V6.1 distribution
- New commands
- NDAF SMIT fast paths
- NDAF logs online information
- NDAF data transfer methods

The enhancements (except 7.2.5, "NDAF data transfer methods" on page 285) described in the following sections are also applied to AIX V5.3 TL6 and later.

7.2.1 Integration of NDAF to the base AIX V6.1 distribution

As previously mentioned, NDAF itself is not new function in Version 6.1. It was shipped as a package of extension packs from AIX 5L V5.3 TL5. NDAF is now integrated on AIX V6.1 base packages (see Example 7-1).

Example 7-1 NDAF packages for AIX V6.1

```
# lslpp -L ndaf*
  Fileset                      Level  State  Type  Description (Uninstaller)
  ----------------------------------------------------------------------------
  ndaf.base.admin            6.1.0.0    C     F    Network Data Administration
                                                   Facility admin server
  ndaf.base.client           6.1.0.0    C     F    Network Data Administration
                                                   Facility client
  ndaf.base.server           6.1.0.0    C     F    Network Data Administration
                                                   Facility server


State codes:
 A -- Applied.
 B -- Broken.
 C -- Committed.
 E -- EFIX Locked.
 O -- Obsolete.  (partially migrated to newer version)
 ? -- Inconsistent State...Run lppchk -v.

Type codes:
```

```
 F -- Installp Fileset
 P -- Product
 C -- Component
 T -- Feature
 R -- RPM Package
E -- Interim Fix
```

7.2.2 NDAF commands

AIX V6.1 and AIX 5L V5.3 TL6 provide new commands to prepare systems for running their processes:

- The **mkndaf** command configures the system to run NDAF.
- The **chndaf** command changes various parameter settings used by the **dms** command and **dmadm** command.
- The **lsndaf** command displays the configuration used by the NDAF daemons.
- The **rmndaf** command configures the system to stop running NDAF daemons.

7.2.3 NDAF SMIT fast paths

You can use SMIT fast paths to go directly to your NDAF panel of choice, rather than navigate there screen by screen. You can see the table that describes the fast path, screen name, and descriptions in the AIX V6.1 InfoCenter.

7.2.4 NDAF logs online information

NDAF logs are described in the online product documentation. In the manual, you can read the following contents:

- Log messages path
- Log detail levels
- Log messages format
- Process types in log files

7.2.5 NDAF data transfer methods

The rsync transfer method was not enabled in AIX 5L V5.3. From AIX 5L V5.3 TL6 and forward, the rsync transfer method can now be used. There are two methods of data transfer now:

copy Performs data transfer using full file tree copy. The copy method implements the data transfer method plug-in interface and performs a data transfer operation by doing a complete walk of the directory tree for the data set and transmitting all objects and data to the target.

rsync Performs data transfer using a rsync-like algorithm. The rsync method performs a data transfer operation by doing a complete walk of the directory tree for the data set and transmitting only deltas for directories and data to the target. It is beneficial when updating replicas because it only sends changed blocks of information, so it reduces network bandwidth considerably.

Important: On AIX V6.1, if you want to use the rsync methods, you need to install the clic.rte fileset (from the expansion CD-ROM). If you do not install it, the copy method will be used.

7.2.6 NDAF case study

The online manual was updated in AIX V6.1 to include use cases descriptions. These use cases deal with:

- Configuring a Kerberos-enabled NDAF domain.
- Federating data from two distant sites and replicating data to enhance network affinity.
- Add an existing server with NFS exported data to an NDAF cell namespace without installing NDAF on it.

7.3 Enabling SSL support for FTP

AIX V6.1 introduces a secure version of **ftp** (and **ftpd**), based on OpenSSL, using Transport Layer Security [1](TLS) to encrypt both the command and the data channel. TLS is a cryptographic protocol that provides secure communication between clients and servers. This enables any user on the system to exchange files in a secure manner if their counterpart offers this extension as well.

While at first look, using secure **ftp** only and no secure **telnet** might not be the most desirable scenario, this method is in fact a reasonable alternative for environments where you are not able to use OpenSSH. For example, if your most trusted systems run on a dedicated and segregated network, it makes sense to use **telnet** for remote access within that very network zone (or working from the console).

But even in such scenarios, you might need to transfer data from or to this secure zone, which can be accomplished now by using secure FTP.

Another scenario might be when you use OpenSSH already, but you still have to exchange data with outside systems that do not support any form of SSH (**scp** or **sftp**). Most often, such systems offer FTP over SSL (often called FTPS) instead.

Since TLS relies on Secure Sockets Layer, make sure OpenSSL is installed on your AIX system (**ftp -s** depends on libssl.a and libcrypto.a). OpenSSL (0.9.8) is needed. It is shipped with AIX V6.1 as the openssl.base fileset (See Example 7-2).

Example 7-2 OpenSSL filesets

```
# lslpp -L openssl*
  Fileset                      Level  State  Type  Description (Uninstaller)
  ----------------------------------------------------------------------------
  openssl.base               0.9.8.4    C     F    Open Secure Socket Layer
  openssl.license            0.9.8.4    C     F    Open Secure Socket License


State codes:
 A -- Applied.
 B -- Broken.
 C -- Committed.
 E -- EFIX Locked.
 O -- Obsolete.  (partially migrated to newer version)
 ? -- Inconsistent State...Run lppchk -v.
```

[1] This extension to FTP is defined in RFC 4217.

```
Type codes:
 F -- Installp Fileset
 P -- Product
 C -- Component
 T -- Feature
 R -- RPM Package
E -- Interim Fix
```

The changes to the **ftp** command and **ftpd** daemon are documented elsewhere. To configure FTP over OpenSSL, see the online product documentation and *AIX V6 Advanced Security Features Introduction and Configuration*, SG24-7430.

7.4 NFS proxy serving enhancements

NFS proxy serving has been introduced with AIX 5L V5.3 Technology Level 5300-05. You can use an NFS proxy server to potentially extend NFS data access over slower or less reliable networks with improved performance and reduced network traffic to the back-end server where the data resides. The NFS proxy server uses a cache file system to provide faster data access to the NFS clients. It supports both NFSv3 and NFSv4 protocols.

AIX V6.1 introduces the following enhancements to NFS proxy serving:

- Comprehensive RPCSEC_GSS Kerberos support from client to proxy and back-end communication.
- Added support for NFSv3 clients at the proxy for an NFSv4 back-end server.
- Support for NFSv4 global namespace exports within one cachefs.
- The NFS proxy cachefs is now persistent across remounts of the cachefs.
- Cachefs performance improvement and increased file limits.

In Figure 7-1, the NFS proxy 1 setup shows the new NFSv3 client support for a back-end NFSv4 server. The NFS proxy 2 setup shows the comprehensive Kerberos support compared to the implementation of previous versions (NFS proxy a).

NFS proxy serving now can be set up with the Web-based System Manager. The new dialogs are explained in 5.1.1, “The mknfsproxy and rmnfsproxy interfaces” on page 202.

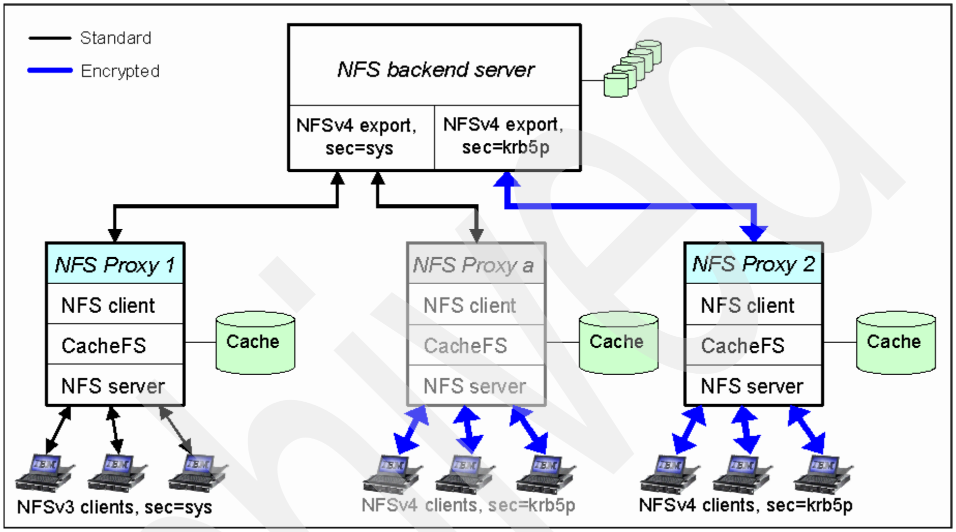


Figure 7-1 NFS proxy serving enhancements

7.4.1 NFS server proxy prerequisites

The following software must be installed on your systems:

- NFS client fileset
  - bos.net.nfs.client
- NFS proxy server filesets
  - bos.net.nfs.client
  - bos.net.nfs.server
  - bos.net.nfs.cachefs
- NFS server fileset
  - bos.net.nfs.server

If you want to use the RPCSEC_GSS Kerberos security method, you must install the following additional filesets and have an configured Kerberos server in your network:

- clic.rte
- krb5.client.rte
- krb5.client.samples
- krb5.lic

7.4.2 Comprehensive RPCSEC_GSS Kerberos support

With the previous NFS proxy serving version, RPCSEC_GSS, Kerberos was only supported between the NFS client and the NFS proxy server. The communication between the proxy server and the back-end server had to be done with the auth_sys security methods. AIX V6.1 introduces the ability to benefit from the stronger Kerberos methods (krb5, krb5i, and krb5p) through all three involved components: NFS client <-> NFS proxy <-> NFS server.

On the NFS client, you have to obtain a valid forwardable ticket. This same ticket is then used by the NFS proxy to authenticate it self and establish a security context with the NFS server.

In this section, we will go trough a step by step tutorial to achieve the following NFS setup:

- The NFS server *lpar03* exports the /projects/project1 file system with the NFS options sec=krb5p and vers=4.
- The NFS proxy server *lpar02* is providing the lpar01 access to the /projects/project1 NFS export.
- The NFS client *lpar01* mounts /projects/project1 to /project1 on his system.

NFS server export

Use the following **mknfsexport** command on the NFS server to export the /projects/project1 file system:

```
# mknfsexp -d /projects/project1 -v 4 -S krb5p
```

NFS proxy setup

In order to set up the proxy file system, you need to meet the following requisites:

- The proxy server has to have a machine principle.
- You need a user principle with a valid ticket. The ticket is used only during the mount operation. If the ticket is expired, the clients will still have access to the NFS data through the NFS proxy.

1. Obtain a valid Kerberos ticket first:

```
# /usr/krb5/bin/kinit nim
```

2. Use the **klist** command to verify if you have obtained a valid ticket:

```
# /usr/krb5/bin/klist
Ticket cache:  FILE:/var/krb5/security/creds/krb5cc_0
Default principal:  nim@REALM1.IBM.COM

Valid starting     Expires            Service principal
10/15/07 13:49:47  10/16/07 13:49:33
krbtgt/REALM1.IBM.COM@REALM1.IBM.COM
```

3. Use the following **mknfsproxy** command to set up the NFS proxy serving:

```
# mknfsproxy -c /cache/projects/project1 -d /projects/project1 \
-m vers=4,sec=krb5p lpar03:/projects/project1 \
-e vers=4,sec=krb5p
```

NFS client mount

To obtain a forwardable ticket, you need to run the following command:

```
# /usr/krb5/bin/kinit -f nim
```

1. Use the **klist** command to verify if you have obtained a valid ticket:

```
# /usr/krb5/bin/klist
Ticket cache:  FILE:/var/krb5/security/creds/krb5cc_0
Default principal:  nim@REALM1.IBM.COM

Valid starting     Expires            Service principal
10/15/07 20:52:22  10/16/07 20:51:49
krbtgt/REALM1.IBM.COM@REALM1.IBM.COM
```

2. Use the **mount** command to make the file system available. No special options are required to mount a proxy NFS export:

```
# mount -o vers=4,sec=krb5p lpar02:/projects/project1 /project1
```

Considerations

The following are considerations when using Kerberos in an NFS environment:

- The NFS proxy does not support a security list. For example, you cannot specify the two security versions krb5i and auth_sys for the front-end export.
- The back-end NFS server does not have to be an AIX system. The system has to be able to handle Kerberos authentication through NFS.
- The actual client does not have to be an AIX system. The system must be able to handle and give out forwardable tickets.

7.4.3 NFSv3 exports for back-end NFSv4 exports

With AIX V6.1 and later, it is possible to create an NFSv4 proxy export for NFSv3 clients. In previous versions, the NFS protocol used at the back-end NFS server had to be the same as for the NFS proxy export. The combinations shown in Table 7-1 are now supported.

Table 7-1 NFS protocol support for NFS proxy serving

Back-end protocol	Front-end protocol
NFSv3	NFSv3
NFS v4	NFSv3
NFSv4	NFSv4

Exporting an NFSv4 back-end as an NFSv3 front-end export improves integration between the two protocols. It provides you a valuable migration tool and adds flexibility to migration scenarios when moving from an NFSv3 to NFSv4 environment.

Use the following **mknfsproxy** command on the NFS proxy server to set up an NFSv4 back-end as an NFSv3 front-end export:

```
# mknfsproxy -c /cache/projects/project2 -d /projects/project2 \
-m vers=4 lpar03:/projects/project2 -e vers=3
```

7.4.4 NFSv4 global namespace

In previous AIX versions, it was not possible to create an NFS proxy export on an NFSv4 back-end export using the global namespace access (also known server pseudo file system or pseudo root) within a single cachefs. You were able to mount the back-end nfsroot and then create manually a separate cachefs for each back-end export.

AIX V6.1 enables the system administrator to mount the global namespace and create a single cachefs. A new **mount** command option mfsid is introduced and can be specified within the **mknfsproxy** command:

```
# mknfsproxy -c /cache/projects -d /projects -m vers=4,mfsid \
lpar03:/ -e vers=4
```

Example 7-3 shows the views of the global namespace export from the NFS server, NFS proxy, and NFS client.

Example 7-3 Global namespace export view from server, proxy, and client

```
lpar03, NFS server:
# exportfs
/projects/project1 -vers=4
/projects/project2 -vers=4

lpar02, NFS proxy server:
# nfs4cl showfs

Server      Remote Path         fsid                Local Path
--------    ---------------     ---------------     ---------------
lpar03.itsc.austin.ibm.com /projects/project1   0:42949672977
   /cache/projects/.cfs_mnt_points/_/projects/project1
lpar03.itsc.austin.ibm.com /projects/project2   0:42949672978
   /cache/projects/.cfs_mnt_points/_/projects/project2
lpar03.itsc.austin.ibm.com /                    0:42949672964
   /cache/projects/.cfs_mnt_points/_
# exportfs
/projects -vers=4

lpar01, NFS client:
# mount -o vers=4 lpar02:/projects /mnt
# nfs4cl showfs

Server      Remote Path         fsid                Local Path
--------    ---------------     ---------------     ---------------
lpar02.itsc.austin.ibm.com /projects/projects/project1 0:47244640287
   /mnt/projects/project1
lpar02.itsc.austin.ibm.com /projects/projects/project2 0:47244640281
   /mnt/projects/project2
lpar02.itsc.austin.ibm.com /projects            0:47244640268
   /mnt
# find /mnt -type d
/mnt
/mnt/projects
```

```
/mnt/projects/project2
/mnt/projects/project2/lost+found
/mnt/projects/project1
/mnt/projects/project1/lost+found
```

7.4.5 Cachefs improvements

The following cachefs improvements are introduced with AIX V6.1:

- Cachefs content is now persistent across remounts.
- Supports caching of files larger than 2 GB.
- Can cache up to 1024 KB files.
- The maximum total amount of cached data is increased to 1 TB.
- Improved overall performance through internal enhancements.

7.5 Network caching daemon

Today, most network-based applications require resolving an Internet host name to an IP address and vice-versa. Latency in this translation procedure directly affects the performance of applications. AIX V6.1 introduces the network caching daemon (**netcd**) to improve performance for resolver lookups. In addition, **netcd** can cache user and group information provided by a NIS server.

7.5.1 The netcd architecture

This section describes the architecture of the **netcd** daemon.

Caching resolver lookups

Applications requiring name resolution place a request to the resolver to do the translation. The resolver does this translation by looking up the corresponding entry in a database. The database is located either on the local machine (for example, /etc/hosts) or on a remote machine that provides a name resolution service (for example, DNS or NIS). For applications requiring frequent name resolutions of a small subset of host and domain names, this process can be inefficient.

The resolver is used by applications to resolve host and domain names to an IP address and vice-versa. The queries can be one of the following types:

- Hosts
- Protocols
- Networks
- Services
- Netgroup

The resolver utilizes one of the following resources to resolve the query of one of the types:

- /etc/hosts
- /etc/networks
- /etc/protocols
- /etc/services
- /etc/netgroup
- Domain Name Server (DNS)
- Network Information Server (NIS)
- Network Information Server+ (NIS+)
- Dynamic user loadable module (ULM)

The **netcd** daemon can be used to cache the resolver lookups. Translations for IPv4 and IPv6 are supported. The communication between the resolver and the **netcd** daemon is done with a UNIX socket (/dev/netcd).

> **Note:** The **netcd** caching will not affect the resolver behavior in the order the resources are queried. The NSORDER environment variable and the /etc/netsvc.conf and the /etc/irs.conf files are consulted by the resolver in the normal manner.

Caching NIS user and group information

In addition, **netcd** can cache user and group information provided by a NIS server. The queries to the following NIS maps can be cached:

- passwd.byname
- passwd.byuid
- group.byname
- group.bygid
- netid.byname
- passwd.adjunct.byname

The ypcall system calls have been modified to use the **netcd** daemon if configured. If the requested information is cached, **netcd** returns the values. If the requested information is not cached, the yplayer requests the information with RPC calls from the NIS server. The response is sent back to the yplayer.

Before normal NIS processing can continue, the yplayer sends the NIS server response to the **netcd** daemon for caching the values. All communication between the yplayer and the **netcd** daemon is achieved with a UNIX socket (/dev/netcd).

Caching

Caches are held as hashed tables to provide fast access. The netcd daemon will maintain two types of caches based on whether the resource it uses is local or network-based.

Local resources, such as /etc/hosts, are loaded into local caches at the startup of the **netcd** daemon. Therefore, local caches contain all entries of the corresponding local resource and a resolver request to it will always result in a cached **netcd** reply. In environments with large local resources, resolver lookups to the hashed cache entries will result in faster response time compared to the traditional linear search of the local resource. The **netcd** daemon will periodically check if the local resources have changed and if necessary reload them.

The **netcd** daemon will also cache resolver lookups to a network resource, such as DNS. In contrast to local caches, the network caches are created with empty entries during the daemon startup. The **netcd** daemon will populate the cache with the result of each query at runtime. Negative answers from the resource are cached as well. When an entry is inserted to the cache, a time-to-live (TTL) is associated to it. For DNS queries, the TTL value returned by the DNS server is used with the default settings. The **netcd** daemon will check periodically for expired entries and remove them.

7.5.2 netcd AIX integration

The **netcd** daemon is delivered as part of the bos.net.tcp.client package. Three new important files are introduced with **netcd**. Table 7-2 shows the function of the new files.

Table 7-2 New netcd files

File	Description
/usr/sbin/netcd	The **netcd** daemon itself.
/usr/sbin/netcdctrl	The command to manage **netcd** daemon caches. Operations include dumping caches, flushing caches, changing the logging level of **netcd**, and display statistics.

File	Description
/usr/samples/tcpip/netcd.conf	A sample configuration file for the **netcd** daemon.

The **netcd** daemon is part of the TCP/IP System Resource Controller (SRC) group. You can use the **startsrc**, **stopsrc**, and **lssrc** command to control the daemon. The **refresh** command is not supported.

The daemon is started in /etc/rc.tcpip script during AIX startup. Note that the daemon is not activated by default in AIX V6.1.

There is no SMIT panel available for **netcd**.

7.5.3 netcd configuration

A **netcd** sample configuration file is installed in /usr/samples/tcpip/netcd.conf. You can copy the file to the /etc/ directory and use it as a template for your configuration. If the **netcd** daemon does not detect a configuration file during startup, it will use its default values. The **lssrc -l netcd** command provides you with an overview of the currently active configuration:

```
# lssrc -ls netcd
Subsystem         Group            PID          Status
 netcd            netcd            421904       active
Debug                         Inactive
Configuration File            /etc/netcd.conf
Configured Cache             local hosts
Configured Cache             dns hosts
```

The /etc/netcd.conf file has four different types of configurations:

- Caching settings
- Security settings
- Log level settings
- Daemon settings

Caching settings

You can specify what resolver or NIS ypcalls should be cached in this section. Use the following syntax:

```
# cache  <type_of_cache>  <type_of_map>  <hash_size>  <cache_ttl>
```

Table 7-3 on page 297 list the possible values.

Table 7-3 Caching settings in /etc/netcd.conf

Attribute	Description
type_of_cache	Declares the type of cache. Possible values are all, local, dns, nis, nisplus, and yp. Any other value will be taken as ulm name.
type_of_map	Declares the map to be used to do the lookup. The possible values depends on the chosen cache type. Consult the netcd.conf man page or look at the sample file for a complete list.
hash_size	Specifies the number of lines used for the cache. An hash table is used to store the cache.
cache_ttl	Declares the time to life for a cache entry. The unit is minutes. The TTL is not used for local resource caches. If you specify an value other than 0 for DNS caches, it will overwrite the TTL of the DNS server response.

The following is an example entry for a DNS cache:

```
cache  dns  hosts  128  0
```

If no cache statement is present in the configuration file, the default setting for the netcd daemon is:

```
cache  all   all    128  60
```

Security settings

You can specify under which user and group context a **netcd** daemon should be run. The default user is root and the default group is system. You are also able to specify an chroot working directory. The default is the / directory.

Declare your settings with the following syntax:

```
owner  <username>
group  <groupname>
home  <homedirectory>
```

Log level settings

The **netcd** daemon creates a log file in /var/tmp/netcd.log. You can specify a different log file location, a log file size limit in KB, and the number of log file rotations. The default setting is no size limit and therefore no rotations are taken. Use this syntax to change the settings:

```
log_file  <file>
log_rotate  <number>
log_size  <number>
```

Daemon settings

These settings influence the daemon operations. Table 7-4 lists the valid key and value pairs.

Table 7-4 netcd daemon settings

Key	Valid values	Default	Description
net_scan_frequency	<number>	1	Specifies how often the **netcd** daemon looks for expired cache entries in network caches. The unit is minutes.
local_scan_frequency	<number>	1	Specifies how often the **netcd** daemon checks for changes to the local resources. The unit is minutes.
socket_queue_size	<number>	256	Indicates the message queue size. The unit is the number of outstanding requests.

7.5.4 Managing netcd

You can use the new **netcdctrl** command to manage the **netcd** daemon. This section gives you examples of all operations supported with the **netcdctrl** command. All operations, except for the logging level change, accept those flags to control the cache selection:

```
-t <type_of_cache>
-e <type_of_map>
```

Dump cache content

With the **netcdctrl** command, you can dump the cache contents to a file. The dump can be either in binary or ASCII format. To dump the DNS cache in ASCII format, use the following command:

```
# netcdctrl -t dns -e hosts -a /tmp/netcd.cache.out
```

This output shows a sample single cache entry:

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>> ELEM #3
Expiration date : Thu Oct  4 11:58:42 2007
Ulm or resolver name : dns
Query type : 10100002
Query length : 6
Answer (0: positive; 1 : negative) : 0
Query key : 1190916695
String used in query : lpar04
Additional parameters in query:
        query param1 : 2
        query param2 : 0
Length of cached element : 48
################### hostent
Number of aliases = 0
Number of addresses = 1
Type = 2
Length = 4
Host name = lpar04.itsc.austin.ibm.com
Alias =
Address = 9.3.5.114
#################### end of hostent
>>>>>>>>>>>>>>>>>>>>>>>>>>>> END ELEM #3
```

Exchange the -a flag with the -b flag to create a dump in binary format. Every time you restart the **netcd** daemon, it will use new caches unless you specify the -l flag pointing to a previous binary cache dump taken with the **netcdctrl** command.

Display statistics of cache usage

You can display statistics of the cache usage. The output of the command will be directed to the specified file. Use the statistics to verify the value of hash_size attribute in the **netcd** configuration:

```
# netcdctrl -t dns -e hosts -s /tmp/netcd.cache.out
```

This output shows an extract of a statistic file:

```
CACHE dns, hosts, name
Hash index : 0, Max number of entries : 0, Current number of entries : 0
Hash index : 1, Max number of entries : 0, Current number of entries : 0
Hash index : 2, Max number of entries : 0, Current number of entries : 0
.......
Hash index : 53, Max number of entries : 1, Current number of entries : 1
Hash index : 54, Max number of entries : 1, Current number of entries : 1
Hash index : 55, Max number of entries : 0, Current number of entries : 0
Hash index : 56, Max number of entries : 1, Current number of entries : 0
END CACHE dns, hosts, name
```

Flush caches

You can manually flush the caches with the following command:

```
# netcdctrl -t dns -e hosts -f
```

If you flush a local resource cache, the local resource will be reloaded automatically. Use the following command if you changed the /etc/hosts local resource and you want to notify the **netcd** daemon immediately:

```
# netcdctrl -t local -e hosts -f
```

Change the logging level of the netcd daemon

You can change the logging level of the **netcd** daemon dynamically. No restart of the daemon is necessary:

```
# netcdctrl -l 7
```

Table 7-5 lists the available and default log levels.

Table 7-5 netcd logging levels

Log level	Log detail
0	No logging
3 (the default)	Errors (the default)
4	Warnings
5	Notice
6	Info
7	Debug

7.6 IPv6 RFC compliances

The IPv6 implementation in AIX V6.1 is compliant with RFC 4007 and RFC 4443, as published by the Internet Engineering Task Force (IETF).

7.6.1 RFC 4007 - IPv6 Scoped Address Architecture

RFC 4007 describes the scoped address architecture. AIX V6.1 introduces scope zone support, as specified in the RFC.

AIX will automatically assign an unique, consecutive number as the zone ID. If you need to provide a specific zone ID, you can specify the desired zone ID value within the **ifconfig** command:

```
# ifconfig en1 inet6 fe80::6888:8eff:fe61:6606%9/64
```

You can use the **netstat** command to display the assigned zone IDs:

```
# netstat -in
Name  Mtu   Network     Address            ZoneID    Ipkts Ierrs    Opkts Oerrs  Coll
en0   1500  link#2      6a.88.8e.61.66.2              5944     0      329     0     0
en0   1500  9.3.4       9.3.5.112                     5944     0      329     0     0
en0   1500  fe80::6888:8eff:fe61:6602          1      5944     0      329     0     0
sit0  1480  link#3      9.3.5.112                        0     0        0     0     0
sit0  1480  ::9.3.5.112                        3         0     0        0     0     0
en1   65394 link#4      6a.88.8e.61.66.5               156     0        2     0     0
en1   65394 fe80::6888:8eff:fe61:6606          9       156     0        2     0     0
lo0   16896 link#1                                     350     0      353     0     0
lo0   16896 127         127.0.0.1                      350     0      353     0     0
lo0   16896 ::1                                2       350     0      353     0     0
```

More information about the IPv6 Scoped Address Architecture can be found at:

http://www.ietf.org/rfc/rfc4007.txt

7.6.2 RFC 4443 - Internet Control Message Protocol (ICMPv6)

RFC 4443 describes the Internet Control Message Protocol (ICMPv6). ICMPv6 is based on ICMPv4 with enhancements made for the use with IPv6. AIX V6.1 implements the message type and message code changes as defined in RFC 4443, which obsoletes the older ICMPv6 RFC 2463.

More information about the Internet Control Message Protocol can be found at:

http://www.ietf.org/rfc/rfc4443.txt

8

Security, authentication, and authorization

The following enhancements are available in AIX Version 6.1 regarding security, authentication, and authorization:

8.1 The /admin/tmp system directory

Beginning with AIX V6.1, the operating system provides a dedicated system directory /admin/tmp where privileged processes can securely create temporary files. The /admin/tmp directory resides within the /admin file system, which in turn is defined on the newly implemented /dev/hd11admin system logical volume. You can use the standard **ls**, **lsfs**, and **lslv** AIX commands to list the properties of the directory, mount point, file system, and logical volume, respectively:

```
hhaix6:root:/ # ls -el /admin
total 0
drwxr-xr-x-    2 root     system          256 Nov 05 11:23 lost+found
drwxr-xr-x-    2 root     system          256 Oct 05 21:54 tmp

hhaix6:root:/ # ls -eld /admind
rwxr-xr-x-    4 root     system          256 Nov 05 11:23 /admin

hhaix6:root:/ # lsfs /admin
Name            Nodename   Mount Pt   VFS   Size    Options  Auto Accounting
/dev/hd11admin  --         /admin     jfs2  262144  --       yes  no

hhaix6:root:/ # lslv hd11admin
LOGICAL VOLUME:     hd11admin              VOLUME GROUP:   rootvg
LV IDENTIFIER:      00cc72be00004c000000011610d0c243.10 PERMISSION:
read/write
VG STATE:           active/complete        LV STATE:       opened/syncd
TYPE:               jfs2                   WRITE VERIFY:   off
MAX LPs:            512                    PP SIZE:        64 megabyte(s)
COPIES:             1                      SCHED POLICY:   parallel
LPs:                2                      PPs:            2
STALE PPs:          0                      BB POLICY:      relocatable
INTER-POLICY:       minimum                RELOCATABLE:    yes
INTRA-POLICY:       center                 UPPER BOUND:    32
MOUNT POINT:        /admin                 LABEL:          /admin
MIRROR WRITE CONSISTENCY: on/ACTIVE
EACH LP COPY ON A SEPARATE PV ?: yes
Serialize IO ?:     NO
```

As shown by the previous **ls** command listings, the /admin mount point and the /admin/tmp directory are owned by the root user and the system group and have the discretionary access control mode of 755. This makes the /admin/tmp directory only writable by the root user, while the traditional /tmp directory is world writable.

All new LVM objects are created by the **/usr/lpp/bosinst/bi_main** script during the base operating system installation. The configuration parameters for the hd11admin logical volume and the /admin file system are taken from the relevant lv_data and fs_data stanzas in /var/adm/ras/image.data. The **bi_main** script also adds the appropriate stanza for the /admin file system to the /etc/filesystems file:

```
hhaix6:root:/ # pg /etc/filesystems

... omitted lines ...

/admin:
        dev       = /dev/hd11admin
        vol       = "/admin"
        mount     = true
        check     = false
        free      = false
        vfs       = jfs2
        log       = /dev/hd8

... omitted lines ...

hhaix6:root:/ # pg /var/adm/ras/image.data

... omitted lines ...

lv_data:
    VOLUME_GROUP = rootvg
    LV_SOURCE_DISK_LIST =
    LOGICAL_VOLUME = hd11admin
    TYPE = jfs2
    MAX_LPS = 512
    COPIES = 1
    LPs = 2
    BB_POLICY = relocatable
    INTER_POLICY = minimum
    INTRA_POLICY = center
    WRITE_VERIFY = off
    UPPER_BOUND = 32
    SCHED_POLICY = parallel
    RELOCATABLE = yes
    LABEL = /admin
    MIRROR_WRITE_CONSISTENCY = on
    LV_SEPARATE_PV = yes
    MAPFILE =
    PP_SIZE = 64
    STRIPE_WIDTH =
    STRIPE_SIZE =
    SERIALIZE_IO =
    FS_TAG =
    DEV_SUBTYP =

... omitted lines ...
```

```
fs_data:
    FS_NAME = /admin
    FS_SIZE = 262144
    FS_MIN_SIZE = 262144
    FS_LV = /dev/hd11admin
    FS_JFS2_BS =
    FS_JFS2_SPARSE =
    FS_JFS2_INLINELOG =
    FS_JFS2_SIZEINLINELOG =
    FS_JFS2_EAFORMAT =
    FS_JFS2_QUOTA =
    FS_JFS2_DMAPI =
    FS_JFS2_VIX =
    FS_JFS2_EFS =
    FS_IMAGE_TYPE =
```

8.2 AIX Security Expert enhancements

AIX Security Expert has been enhanced with new features to further improve the system security and prevent intrusions. These features include:

- Centralized Policy Distribution through Light Weight Directory Access Protocol (LDAP)
- Ability to customize and include user-defined policies
- More stringent checks for weak root passwords
- Enable stack execution disable (SED) in AIX Security Expert
- File permission manager (**fpm**) command for managing SUID programs
- Invoking Secure by default for the high security setting
- SOX-COBIT assistant
- Performance enhancements for the graphical interface

The following sections discuss these enhancements in turn.

8.2.1 Centralized policy distribution through LDAP

The support for a centralized policy file that is stored in LDAP is one of the important security enhancements to AIX Security Expert in AIX V6.1.

An AIX Security Expert policy file can be created and saved in a central location on an LDAP server. The LDAP server stores the policy file containing the XML rules that is read by AIX Security Expert to determine security settings. Then, as other machines in the network need to be hardened, the policy file is fetched from the LDAP server and a consistent policy is distributed and maintained throughout the enterprise.

8.2.2 User-defined policies

It is possible to define your own security policy or rules that are automatically integrated into the AIX Security Expert tool and GUI. Therefore, any security configuration policy unique to your environment or relating to third-party software can be easily brought under the control and management of AIX Security Expert. If you use this customization, you need to create an XML file in the /etc/security/aixpert/custom directory.

For more information, see “The predefined SOX-COBIT security policy” in *AIX V6 Advanced Security Features Introduction and Configuration*, SG24-7430.

8.2.3 More stringent check for weak root passwords

This feature checks for weak root passwords. This feature checks for easily guessed root passwords. The location of this option is illustrated in Figure 8-1 on page 308.

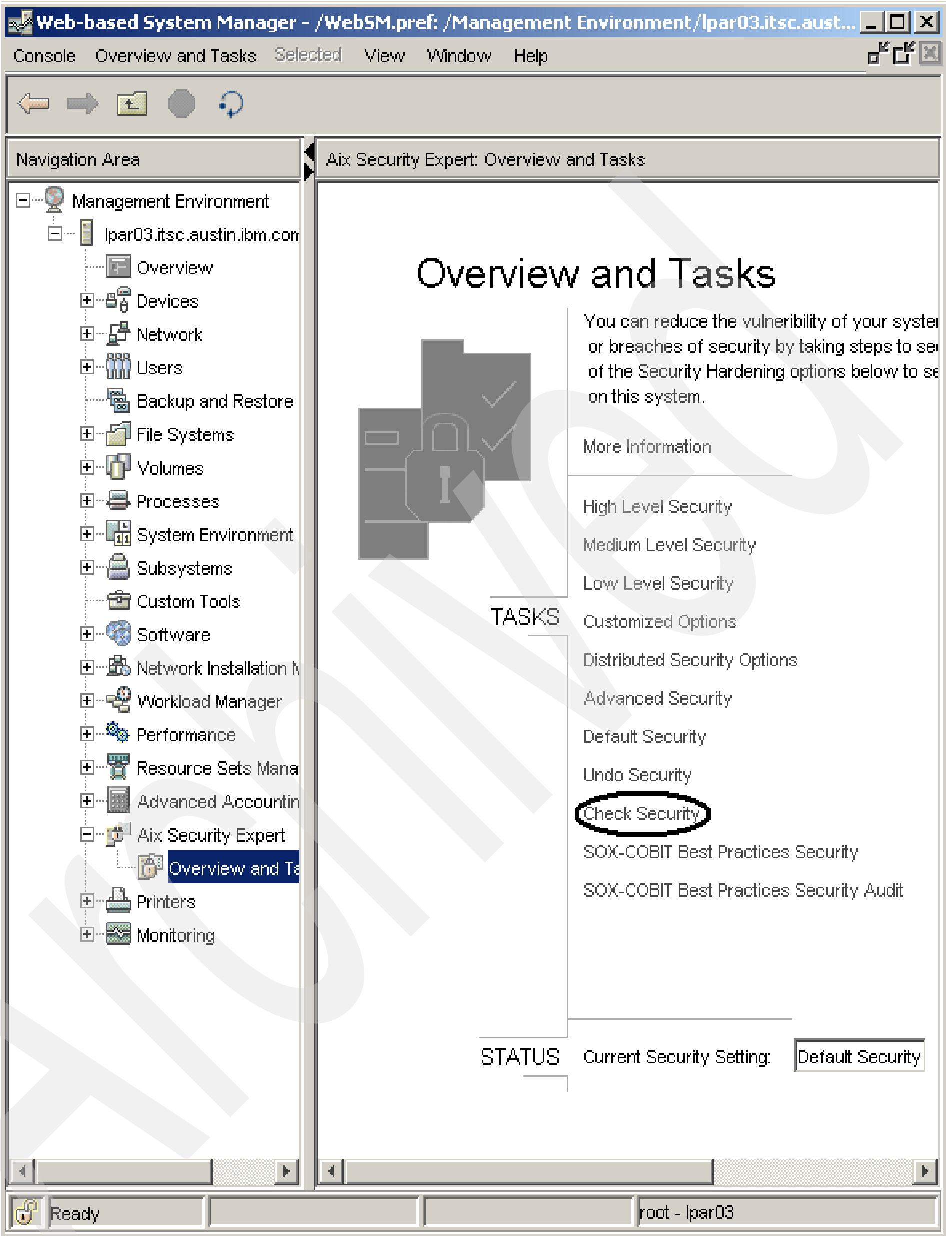


Figure 8-1 Management environment tasks

This feature reads the encrypted password from /etc/security/passwd for root. For example:

```
root:
password = ni3nZoDlxC52c
```

For each entry in the dictionary (located in /etc/security/aixpert/dictionary directory), the password is read. This encrypted output is compared with the stored encrypted password; if there is a match, AIX Security Expert must report that root has a weak password.

Before this is done, an administrator has to check the Root Password Integrity Check check box, as shown in Figure 8-2.

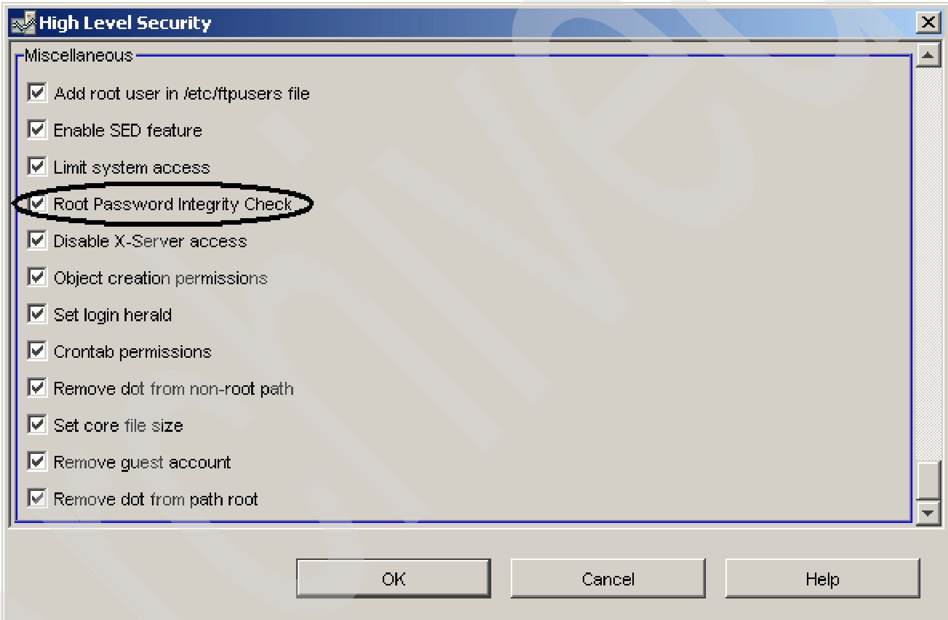


Figure 8-2   Root Password Integrity Check interface

The check box can be seen when an administrate selects the **Miscellaneous** section of High Level Security or Medium Level Security.

> **Note:** This feature provides a command option to check any user password integrity, but AIX Security Expert does not provide menu options to check other users. Instead, a dictionary is developed as part of this feature, and when root or other users change their passwords, their new password must not be found in this dictionary. The dictionary is installed in /etc/security/aixpert/dictionary/English. The file is shipped with AIX Security Expert (bos.aixpert.cmd fileset).

8.2.4 Enabling Stack Execution Disable (SED)

Stack Execution Disable itself is introduced in AIX 5L 5.3 TL4. In AIX V6.1, it is added to the graphic interface; you can now see the Enable Stack Execution Disable check box in Miscellaneous section of High Level Security, as shown in Figure 8-3.

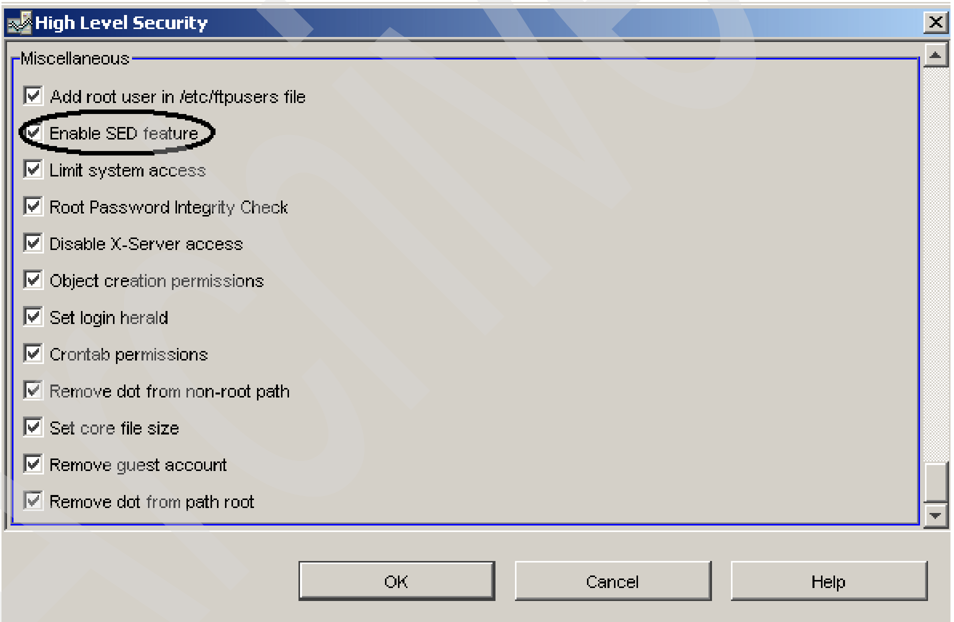


Figure 8-3 Enable SED Feature Interface

8.2.5 File permission Manager (fpm) for managing SUID programs

File Permission Manager (**`fpm`**) manages the permissions on commands and daemons owned by privileged users with setuid or setgid permissions. This command will be provided in AIX V6.1 and AIX 5L V5.2 TL10 and AIX 5L V5.3

TL7 at the time of writing. AIX Security Expert provides the interface of the File Permissions Manager, as shown in Figure 8-4.

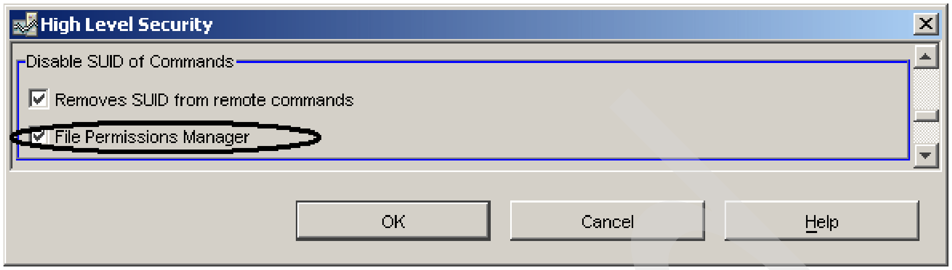


Figure 8-4 File Permissions Manager Interface on AIX Security Expert

The **fpm** command allows administrators to harden their system by disabling the setuid and setgid bits on many commands in the operating system. This command is intended to remove the setuid permissions from commands and daemons owned by privileged users, but you can also customize it to address the specific needs of unique computer environments with the command options.

> **Note:** The **fpm** command cannot run on TCB-enabled hosts.

Example 8-1 shows an example of the **fpm** command.

Example 8-1 Changing Level and Restore setting scenarios

```
##### Check current status
# fpm -s
Default level security.
# more 10192007_12:20:49

##### Check current file permissions
# ls -l /usr/bin/acctctl
-r-sr-s---    1 root     adm          203601 Sep 24 18:24 /usr/bin/acctctl

##### Change Low Level
# fpm -l low
One or more file is already secure.  Therefore, the current file permissions may not
match the default permissions.  If you wish to return to the snapshot of permissions
prior to running this command, then use the command:
 /usr/bin/fpm -l default -f /var/security/fpm/log/10192007_13:02:57

 fpm will now continue to remove the SUID permissions.

##### Check current file permissions: suid is removed
```

```
# ls -l /usr/bin/acctctl
-r-xr-s---    1 root     adm          203601 Sep 24 18:24 /usr/bin/acctctl

##### Change Medium Level
# fpm -l medium
One or more file is already secure.  Therefore, the current file permissions may not
match the default permissions.  If you wish to return to the snapshot of permissions
prior to running this command, then use the command:
 /usr/bin/fpm -l default -f /var/security/fpm/log/10192007_13:03:18

 fpm will now continue to remove the SUID permissions.

##### Check current file permissions: sgid is removed
# ls -l /usr/bin/acctctl
-r-xr-x---    1 root     adm          203601 Sep 24 18:24 /usr/bin/acctctl

##### Change Default Status
# fpm -l default
fpm will restore the AIX file permissions to the installed settings and any
customized defaults listed in /usr/lib/security/fpm/custom/default. If you had done
other customizations outside fpm and wish to return the file permissions to a state
representing a particular time and date, use the command:
 fpm -l default -f /var/security/fpm/log/<in_file>
Where <in_file> is a previously saved timestamped file representing this system's
file permission state at a particular date and time.

##### Check current file permissions: suid, sgid are restored
# ls -l /usr/bin/acctctl
-r-sr-s---    1 root     adm          203601 Sep 24 18:24 /usr/bin/acctctl
```

Attention: The **fpm** command writes a log in the /var/security/fpm/log directory. Ensure that there is free space for the directory and log. If there is no space to log, the command will fail.

8.2.6 Secure by Default

Secure by Default takes a bottom-up approach in hardening an AIX system by installing a minimal set of software, because any additional software could increase the potential for a security vulnerability, and then applying a high security level hardening to those components. This approach is opposite to starting with a regular, full-blown AIX installation and then use the AIX Security Expert to apply hardening (top-down approach) by disabling unneeded components.

For more information about the Secure by Default installation, see “Secure by Default” in *AIX V6 Advanced Security Features Introduction and Configuration*, SG24-7430.

8.2.7 SOX-COBIT assistant

AIX Security Expert supports the SOX-COBIT Best Practices Security level in addition to the High, Medium, Low, AIX Default, and Advanced Security settings.

The United States Congress enacted the 'Sarbanes-Oxley Act of 2002 to protect investors by improving the accuracy and reliability of financial information disclosed by corporations. The COBIT control objectives feature will help system administrators to configure, maintain, and audit their IT systems for compliance with this law. The SOX Configuration Assistant is accessed through the AIX Security Export Web-based Systems Manager menus or the **aixpert** command line. The feature assists with the SOX Section 404 of the Sarbanes-Oxley Act, but The AIX Security Expert SOX Configuration Assistant automatically implements security settings commonly associated with COBIT best practices for SOX Section 404 (Internal Controls). Additionally, the AIX Security Expert provides a SOX audit feature that reports to the auditor whether the system is currently configured in this manner. The feature allows for the automation of system configuration to aid in IT SOX compliance and in the automation of the audit process.

Since SOX does not offer guidance on how IT must comply with Section 404, the IT industry has focused on the existing governance detailed at http://www.isaca.org/, more specifically, the IT governance covered by Control Objectives for Information and related Technology (COBIT).

AIX Security Expert supports the following control objectives (see Figure 8-5):

- Password policy enforcement
- Violation and Security Activity Reports
- Malicious software prevention, detection and correction, and unauthorized software
- Firewall architecture and connections with public networks

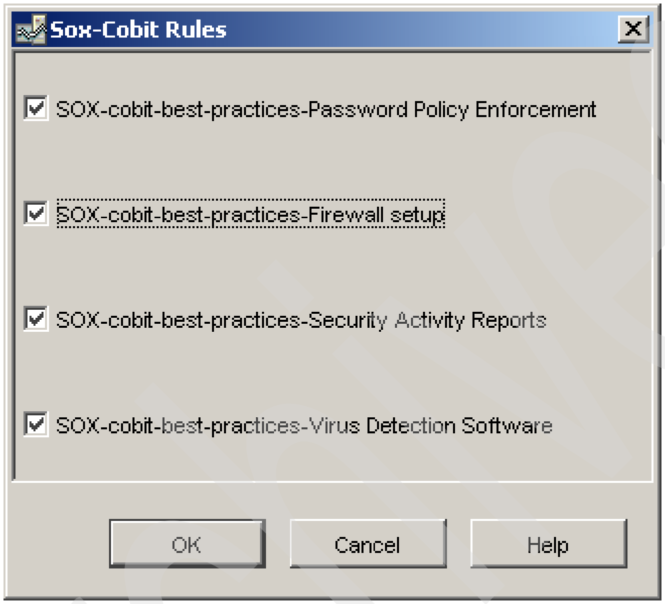


Figure 8-5 Sox-Cobit Rules interface

> **Important:** AIX Security Expert does not support all of the attributes specified under each control objective. For supported attributes, see the COBIT control objectives supported by AIX Security Expert in the AIX V6.1 online manual.

You can use the **`aixpert -c -l s`** command to check a system's SOX-COBIT compliance. AIX Security Expert only checks for the supported control objectives compliance. Any violations found during the checking are reported. By default, any violations are sent to stderr.

You can also use the same command (**`aixpert -c -l s`**) to generate the SOX-COBIT compliance audit report. To generate an audit report, set up and enable the audit subsystem. Ensure that the AIXpert_check audit event is turned on. After setting up the audit subsystem, rerun the **`aixpert -c -l s`** command. The command generates the audit log for every failed control objective. The

Status field of the audit log will be marked as failed. The log also contains the reason for the failure, which can be viewed using the -v option of the **auditpr** command.

Adding the -p option to the **aixpert -c -l s** command also includes successful control objectives in the audit report. Those log entries have OK in the status field.

The **aixpert -c -l s -p** command can be used to generate a detailed SOX-COBIT compliance audit report.

Whether or not the -p option is specified, there will be a summary record. The summary record includes information about the number of rules processed, the number of failed rules (instances of non-compliance found), and the security level that the system is checked for (in this instance, this would be SCBPS).

8.2.8 Performance enhancements for the graphical interface

Performance enhancements for the graphical interface is implemented by replacing some JAVA calls with C code in areas that provide additional performance.

8.3 Enhanced Role Based Access Control

To make the AIX operating system more robust, Role Based Access Control (RBAC) is enhanced in AIX to reduce the complexity of managing the system, and also to provide for finer granular privilege control. The older versions (>=4.2.1 and <=5.3) of AIX have RBAC implemented in the user space. RBAC implementation for AIX V6.1 provides for an enhanced mechanism covering both user and kernel spaces. Enhanced RBAC provides for a framework that allows clients to define administrative roles and delegate the role to regular users. The RBAC framework consists of the followings features:

- Authorizations
- Privileges (command and device)
- Roles

User space framework

The configuration files (they are called as user-level databases) shown in Table 8-1 are provided to support enhanced RBAC.

Table 8-1 File lists for enhanced RBAC facility

File name	Description
/etc/security/authorizations	User-level Authorization Database
/etc/security/roles	User-level Role Database
/etc/security/privcmds	User-level Privileged Command Database
/etc/security/privdevs	User-level Privileged Device Database
/etc/security/privfiles	Privileged File Database

Kernel security tables

After the user-level databases are changed, these changes must be sent to Kernel Security Table (KST) to be applied.

KST consists of the following tables (see Figure 8-6 on page 317):

- User-defined Kernel Authorization Table (KAT)
- System-defined Kernel Authorization Table (KAT)
- Kernel Role Table (KRT)
- Kernel Command Table (KCT)
- Kernel Device Table (KDT)

RBAC security decisions are enforced by the kernel. User-level databases must be sent to KST:

- User-defined Authorization Database → User-defined KAT
- User-level Role Database → KRT
- User-level Privileged Command Database → KCT
- User-level Privileged Device Database → KDT

> **Note:** Privileged File Database is only used by the `pvi` command, so the contents of the file is not sent to KST.

Here are the kernel security tables management commands:

`setkst` Update the KST with data in the user-level databases. Only an entire table update is supported. A way to update single entries in a table is not provided. (KAT requires the KRT and KCT update.)

`lskst` List the data from the KST.

A binary version of KST is saved each time the **`setkst`** command is executed. It is used for reboot and Workload Partition mobility.

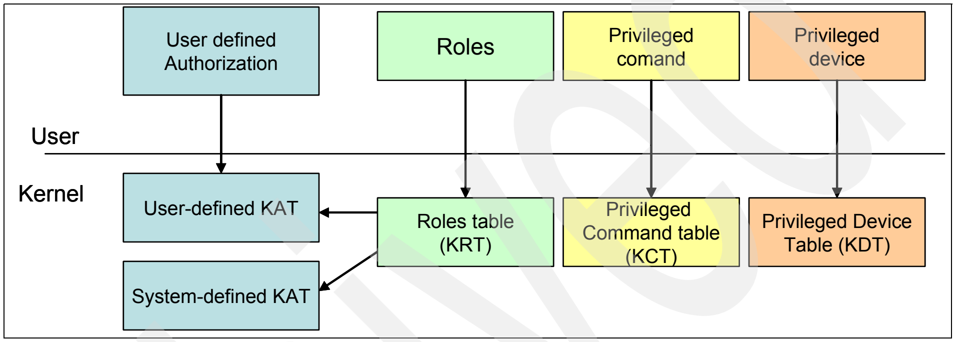


Figure 8-6 Enhanced RBAC Framework on AIX V6.1.

8.3.1 Authorizations

Authorizations are authority attributes for a user. These authorizations allow a user to do certain tasks. An authorization can be thought of as a key that is able to unlock access to one or more commands (see Figure 8-7).

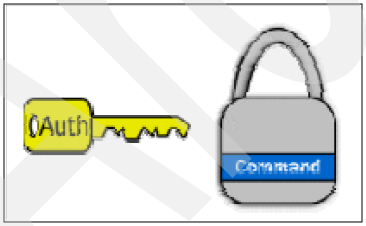


Figure 8-7 Authorizations concept

Authorization in AIX 5L V5.3

In AIX 5L V5.3 and earlier, 11 authorizations were provided in the system (see Table 8-2). These cannot be customized. The commands and authorizations are tightly bound.

Table 8-2 Authorizations in AIX 5L V5.3

Authorization	Description
Backup	Performs a system backup. The **backup** command uses this authorization.
Diagnostics	Allows a user to run diagnostics. This authority is also required to run diagnostic tasks directly from the command line. The **diag** command uses this authorization.
DiskQuotaAdmin	Performs a disk quota. The following commands use this authorization: ► **quotacheck** ► **edquota** ► **j2edlimit** ► **quota** ► **quotaoff** ► **quotaon** ► **repquota**
GroupAdmin	Performs the functions of the root user on group data. The following commands use this authorization: ► **chgroup** ► **chgrpmems** ► **chsec** ► **mkgroup** ► **rmgroup**
ListAuditClasses	Views the list of valid audit classes.
PasswdAdmin	Performs the functions of the root user on password data. The following commands use this authorization: ► **chsec** ► **lssec** ► **pwdadm**
PasswdManage	Performs password administration functions on non-administrative users. The **pwdadm** command uses this authorization.

UserAdmin	Performs the functions of the root user on user data. Only users with the UserAdmin authorization can modify the role information of a user. You cannot access or modify user auditing information with this authorization. The following commands use this authorization; ► **chfn** ► **chsec** ► **chuser** ► **mkuser** ► **rmuser**
UserAudit	Allows the user to modify the user-auditing information. The following commands use this authorization: ► **chsec** ► **chuser** ► **lsuser** ► **mkuser**
RoleAdmin	Performs the functions of the root user on role data. The following commands use this authorization: ► **chrole** ► **lsrole** ► **mkrole** ► **rmrole**
Restore	Performs a system restoration. The **restore** command uses this authorization.

Authorization in AIX V6.1

In AIX V6.1, authorizations are divided into granular parts. The current number of system authorizations is 252. The administrator can specify these authorizations to roles more frequently. Table 8-3 shows the major authorizations for AIX V6.1.

Table 8-3 Top Level authorization on AIX V6.1

Authorizations (Top)	Descriptions
aix.devices	Device Administration
aix.fs	File System Administration
aix.lvm	Logical Volume Manager Administration
aix.mls	Trusted AIX Administration
aix.network	Network Administration
aix.proc	Process Administration

Authorizations (Top)	Descriptions
aix.ras	Reliability, Availability, Serviceability Administration
aix.security	Security Administration
aix.system	System Administration
aix.wpar	System and Application Workload Partition Administration

The authorization name is a hierarchical naming support and the dotted notation denotes hierarchy (See the aix.system.boot.info example in Figure 8-8). There are nine levels of hierarchy allowed, and the parent authorization is a super-set of the children authorizations.

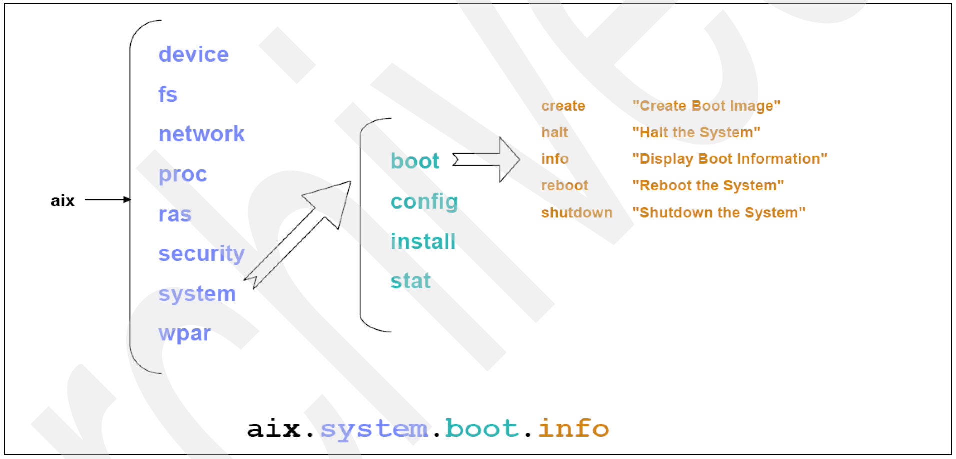


Figure 8-8 Authorization hierarchy

Table 8-4 shows maps for previous authorizations to new authorizations.

Table 8-4 Maps for authorization from AIX 5L V5.3 to AIX V61

Existing Legacy Mode Authorization	Enhanced Mode Authorization
Backup[a]	aix.fs.manage.backup
Diagnostics[a]	ais.system.config.diag
DiskQuotaAdmin[a]	aix.fs.manage.quota
GroupAdmin[a]	aix.security.group
ListAuditClasses[a]	aix.security.audit.list

Existing Legacy Mode Authorization	Enhanced Mode Authorization
PasswdAdmin[a]	aix.security.passwd
PasswdManage[a]	aix.security.passwd.normal
UserAdmin[a]	aix.security.user
UserAudit[a]	aix.security.user.change
RoleAdmin[a]	aix.security.role
Restore[a]	aix.fs.manage.restore

a. Legacy Mode Authorizations remain on AIX V6.1 for compatibility reasons.

To manipulate authorizations, the following commands should be used:

`lsauth` Displays attributes of user-defined and system-defined authorizations from the authorization database.

`mkauth` Creates a new user-defined authorization in the authorization database.

`chauth` Modifies attributes for the existing user-defined authorization.

`ckauth` Checks whether the current user session has the authorizations.

`rmauth` Removes the user-defined authorization.

For more details, refer to *AIX V6 Advanced Security Features Introduction and Configuration*, SG24-7430.

8.3.2 Privileges

A privilege is a process attribute that allows the process to bypass specific system restrictions and limitations. Privileges are the restriction mechanism used in the kernel to determine if a process is allowed to perform a particular action. A privilege can be thought of as an ability that allows a process to overcome a specific security constraint in the system (see Figure 8-9).

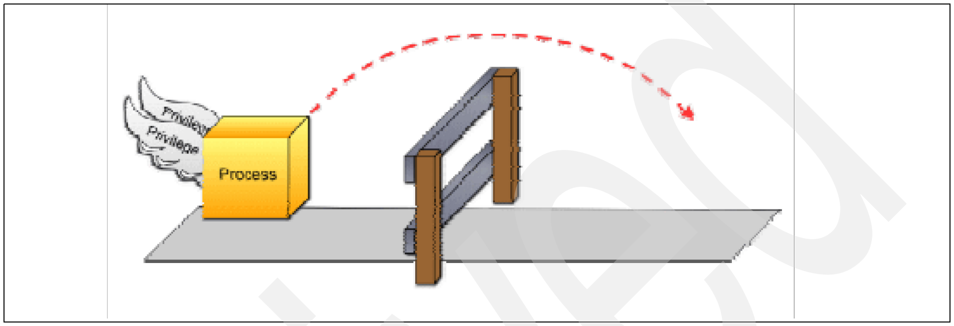


Figure 8-9 Concept of privileges

Process Privilege Sets

Process Privilege Sets are used to dynamically restrict or limit privileged operations. Multiple sets of privileges are defined in the kernel to provide for varied controls in regards to privileged operations.

A process will now have these new privilege sets:

Effective Privilege Set: (EPS)
Used to actually override system restrictions. A process can add or remove privileges from its own EPS, subject to the limitations imposed by the MPS.

Maximum Privilege Set (MPS)
A set of privileges over which a process has control. The MPS is always a super-set of the process' EPS. A process can always remove a privilege from its MPS. A process' MPS can only be increased if the process has the appropriate privilege, and even then it is restricted by the LPS of the process. The MPS of process can also be modified when the process runs an executable file, but this too is limited by the process' LPS.

Limiting Privilege Set (LPS)
Represents the maximum possible privilege set that the process can have. The LPS is always a super-set of the MPS. No process can increase its LPS and any process can reduce its LPS.

Used Privilege Set: (UPS)
This is mainly used by the **tracepriv** command. This set keeps all privileges that are used during the life of a process. It goes away when a process dies.

Inheritable Privilege Set (HPS)
This is set of privileges that are inherited from parent to child. A process can always remove a privilege from its HPS. A process' HPS can only be increased if the process has the appropriate privilege and even then it is restricted by the LPS of the process. The HPS of the process can also be set when the process runs an executable file, but this is also limited by the process' LPS.

Privilege commands

To manipulate privileges, the following commands are introduced:

pvi Provides a privileged editor so that you can access privileged files.

lspriv Displays the privileges available on the system.

tracepriv Traces the privileges that a command needs for a successful run.

The following commands are used to manipulate privileges and used for other security settings:

lssecattr Displays the security attributes of a command, a device, a privileged file, or a process.

setsecattr Sets the security attributes of a command, a device, a privileged file, or a process.

rmsecattr Removes the definition of the security attributes for a command, a device, or a privileged file in the database.

8.3.3 Roles

Roles are mechanism used to assign authorizations to a user and to group a set of system administration tasks together. An AIX role is primarily a container for a collection of authorizations.

AIX supports the direct assignment of authorizations to a role or the indirect assignment of authorizations through a sub-role. A sub-role can be specified for a role in the rolelist attribute of a role. Configuring a role to have a designated sub-role effectively assigns all of the authorizations in the sub-role to the role.

Assigning a role to a user allows the user to access the role and use the authorizations that are contained in the role. A system administrator can assign a role to multiple users and can assign multiple roles to a user. A user who has been assigned multiple roles can activate more than one role (up to a maximum of eight roles) simultaneously if necessary to perform system management functions.

AIX provides a set of predefined roles for system management. However, it is expected that customers will need to create their own custom roles or modify the existing predefined roles. Several role-management commands are available to list, create, modify, and remove AIX roles. Roles can be created with the **mkrole** command, modified with the **chrole** command, removed with the **rmrole** command, and displayed with the **lsrole** command.

The roles allows a set of management functions in the system to be grouped together. Using the analogy that an authorization is a key, a role can be thought of as a key ring that can hold multiple authorizations (see Figure 8-10).

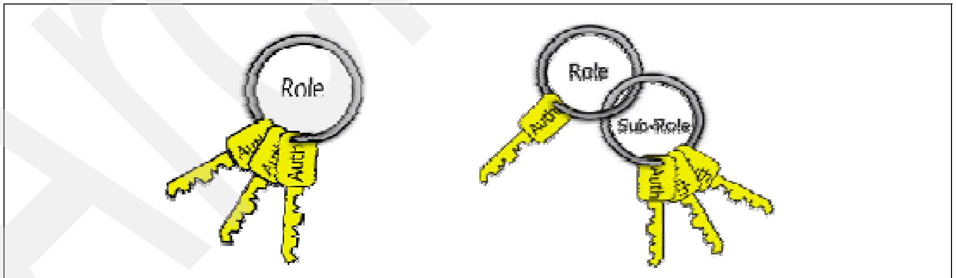


Figure 8-10 Concept of roles

The function of the role itself is not different from the previous one on AIX 5L V5.3. But the contents of roles are completely different. The following tables (Table 8-5 on page 325 and Table 8-6 on page 325) shows roles that the system provides by default.

Table 8-5 List of roles provided by default on AIX 5L V5.3

Roles	Descriptions
ManageBasicUsers	Performs the functions of the root user on user data. Views the list of valid audit classes.
ManageAllUsers	Performs the functions of the root user on role, password data, group data, and user data. Views the list of valid audit classes.
ManageBasicPasswds	Performs password administration functions on non-administrative users.
ManageAllPasswds	Performs the functions of the root user on password data. Performs password administration functions on non-administrative users.
ManageRoles	Performs the functions of the root user on role data.
ManageBackupRestore	Performs a system backup and a system restoration.
ManageBackup	Performs a system backup.
ManageShutdown	Shuts down the system.
RunDiagnostics	Runs diagnostics.
ManageDiskQuota	Performs a disk quota.

Table 8-6 List of roles provided by default on AIX V6.1

Roles	Descriptions
AccountAdmin	User and Group Account Administration
BackupRestore	Backup and Restore Administration
DomainAdmin	Remote Domain Administration
FSAdmin	File System Administration
SecPolicy	Security Policy Administration
SysBoot	System Boot Administration
SysConfig	System Configuration
isso	Information System Security Officer

Roles	Descriptions
sa	System Administrator
so	System Operator

By default, AIX does not activate any roles. A **swrole** command can be used to assume the proper role in order to execute any privileged command or function.

8.3.4 Summary of differences

Table 8-7 shows a summary of differences between AIX 5L V5.3 and AIX V6.1 RBAC functions.

Table 8-7 Differences summary between AIX 5L V5.3 and AIX V6.1

Feature	AIX 5L V5.3	AIX V6.1
Implementation region	Mostly User space	User and Kernel Space
Role		
► Create new roles	Yes	Yes
► Enablement	Default active	Need to activate (**swrole**)
Authorization		
► Structure	Flat	Hierarchical
► Create new authorizations	No	Yes
Privilege		
► Create new privileges	No	No (system provides only)
► Assign privileges to targets	No	Yes (file, device, process)

8.4 Web-based GUI for RBAC

The Web-based GUI for RBAC runs in a browser/server-client environment using Web Services and the Light Weight Infrastructure (LWI) as a plug-in to the IBM Systems Director Console for AIX.

The Web-based GUI for RBAC is provided in AIX V6.1 (see Figure 8-11 on page 327).

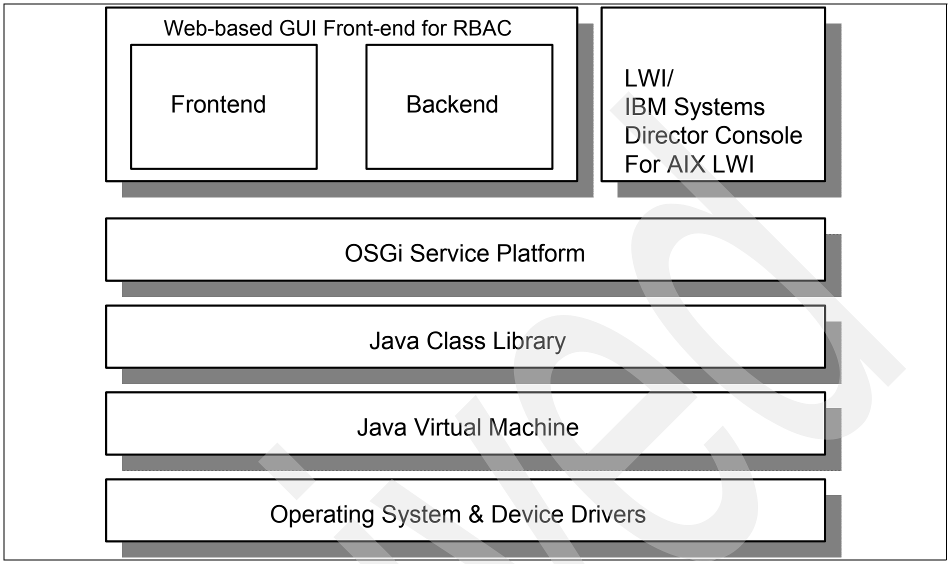


Figure 8-11 IBM Systems Director Console for AIX and RBAC modules

Components

The Web-based GUI for RBAC application is implemented using the schema of a three-tier Web application. The Web-based GUI front end for the RBAC GUI is composed of four parts or layers (see Figure 8-12 on page 328):

Presentation Layer This layer is composed of objects that interact directly with the user (forms, HTML pages, portlets, and so on.)

Application Layer This layer supports the presentation layer by presenting the objects received from the business layer in a way directly usable by the presentation layer.

Business Layer This layer is the heart of the Web-based GUI front end for the RBAC system. It responds to requests from the application layer and manages the persistency, currency, and consistency of data by utilizing the services of the integration layer.

Integration Layer This layer interacts directly with the RBAC subsystem installed on the endpoint.

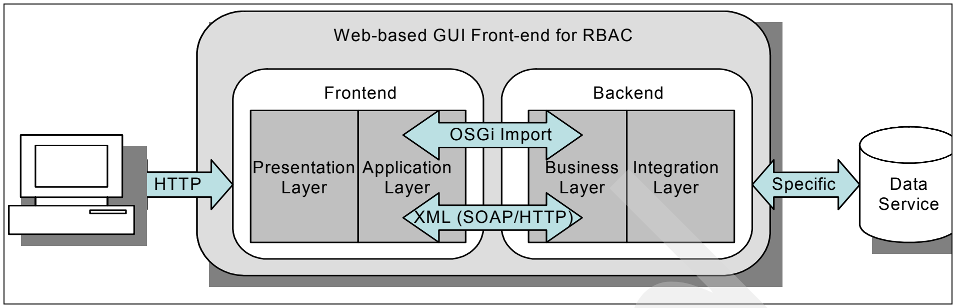


Figure 8-12 Web-Base GUI Component with RBAC

8.4.1 Tasks and roles

An administrator assigns the roles to the users who use specific IBM Systems Directors Console for AIX tasks. After assigning the Console Role to a user, there is more to do. A user can administrate some tasks, but authorizations are also required. For example, the user *nobu* is assigned the aixUser Console role. So, nobu can log into the IBM System Director Console for AIX, but when he moves to the Security & Users tasks, he gets the following messages:

```
WARNING: Some of the commands in this application require system
authorizations which you don't have:
    * aix.security.user
    * aix.security.role
    * aix.security.group
    * aix.security.passwd
```

So, the user nobu needs authorizations. To get authorizations, a new role that uses these authorizations must be created and be assigned to the user nobu.

Table 8-8 shows a mapping list of Console task, Console Role, and AIX authorizations that are needed.

Table 8-8 Task, console role, and authorization map

Task	Console Role	AIX authorizations
Software Installation and Maintenance	aixSoftware	aix.system.install aix.system.stat aix.system.boot aix.network.config
Software License Management	aixLicenses	aix (This authorization is equivalent to root authority.)

Task	Console Role	AIX authorizations
Devices	aixDevices	aix.device
System Storage Management	aixStorage	aix.fs aix.lvm
Security and Users	aixUsers	aix.security.user aix.security.role aix.security.group aix.security.passwd
Communication Applications and Services	aixNetwork	aix.network
Workload Partition Administration	aixWorkloadPartitions	aix.wpar
Print Spooling	aixPrinters	aix.device.config.printer aix.device.stat.printer
Advanced Accounting	aixAdvancedAccounting	aix (This authorization is equivalent to root authority.) aix.system.config.acct
Problem Determination	aixProblemDetermination	aix.ras
Performance and Scheduling	aixPerformanceAndScheduling	aix.system.stat aix.system.config.perf aix.system.config.cron aix.system.config.wlm aix.system.config.init aix.system.config.dlpar aix.proc.status aix.ras.trace
System Environments	aixSystemEnvironments	aix.system aix.ras.dump aix.ras.error aix.device.manage.change
Processes and Subsystems	aixProcessesAndSubsystems	aix.proc aix.system.config.src
Cluster Systems Management	aixCSM	aix (This authorization is equivalent to root authority.)

Task	Console Role	AIX authorizations
SMIT - Classic View	aixSMITclassic	aix (This authorization is equivalent to root authority.)
DCEM	aixDCEM	aix (This authorization is equivalent to root authority.)
Role Based Access Control	aixRBAC	aix (This authorization is equivalent to root authority.) aix.security.role aix.security.auth aix.security.cmd aix.security.device aix.security.file aix.security.proc aix.security.kst
Web-based System Manager	aixWebSysMgr	aix (This authorization is equivalent to root authority.)

8.5 LDAP support enablement

In AIX V6.1, a new framework is used to store the RBAC database tables, including the authorization table, the role table, the privileged command table, and the device table, in a centralized location in LDAP. AIX security libraries can be made aware of the remote tables, and if configured to use them, data can be pulled from these LDAP tables, and then be used the same way as the data is from local files. The LDAP tables are transparent to applications. Some RBAC commands will be made to work with LDAP tables explicitly for the purpose of managing these remote tables.

Name service control file

This is a mechanism to configure the priority of local and LDAP tables, and possibly remote tables other than LDAP. A new control file /etc/nscontrol.conf is provided. This file is a stanza file, with the stanza name being authorizations, roles, privileged commands, and privileged devices. The file only supports one search order attribute. The search orders defined in this file are system-wide.

The content of the file is in the format:

```
stanzakey:
       searchorder = <lookup mechanism>,<lookupmechanims>...
```

An example of this file is as follows:

```
authorizations:
     searchorder = LDAP,files

roles:
     searchorder = LDAP,files

privcmds:
     searchorder = files,LDAP

privdevs:
     searchorder = files

privfiles:
     searchorder = files,LDAP
```

LDAP support enablement commands

To support the LDAP environment, AIX V6.1 includes new commands and enhances existing commands.

The **rbactoldif** command is introduced in AIX V6.1. This command reads RBAC configurations files, and generates RBAC security database data for LDAP.

The following commands supports LDAP databases;

- **mkauth**, **chauth**, **lsauth**, and **rmauth**
- **mkrole**, **chrole**, **lsrole**, and **rmrole**
- **setsecattr**, **lssecattr**, and **rmsecattr**

These commands have the new option -R. The **setkst** command does not have an option, but it recognizes that RBAC information is located in LDAP databases.

LDAP commands are also enhanced to support RBAC tables.

The **lsldap** command supports new RBAC tables:

- auths
- roles
- privcmds
- privdevs
- privfiles

The **mksecldap** command updates the RBAC related setup during LDAP client configuration on the following topics: The base DN update for authbasedn, rolebasedn, privcmdbasedn, and privdevbasedn.

8.6 RBAC and Workload Partition environments

Since creating and managing a System Workload Partition requires the authority to manage resources such as file system, network, and devices, the root user must be used for these tasks. Working as the root user raises some security concerns in a consolidated environment. When working with applications, it is a best practice to not be the root user in order to avoid errors. The proper way to handle this requirement is to delegate privileges based on roles such as those of the application administrator for a particular Workload Partition. The Role Based Access Control (RBAC) mechanism of AIX is being used for this purpose.

> **Note:** The **lspriv** command displays privileges available to the system. If it is run within a workload partition, the **lspriv** command displays only the privileges available to the partition. If the -v flag is specified, the **lspriv** command also displays privilege descriptions.

Figure 8-13 on page 333 shows the relationship between Workload Partition and RBAC.

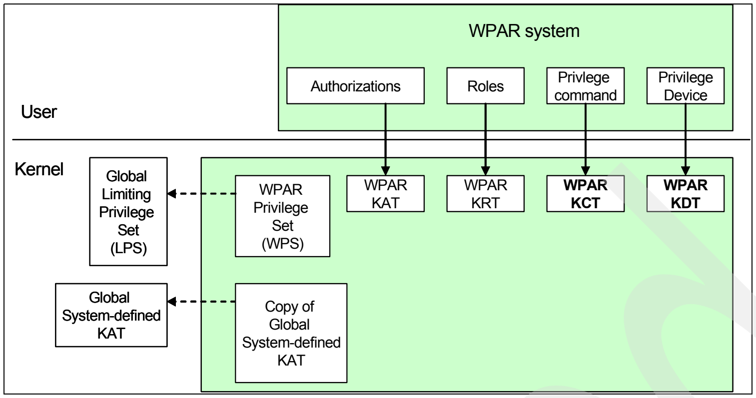


Figure 8-13 RBAC and Workload Partition framework

The Workload Partition system also has its own Authorizations, Roles, and Privileges (command and device) as Global System has them defined. The Workload Partition system has a private copy of Global System-defined KAT and the Workload Partition Privilege Set (WPS). The WPS defines all the privileges that the Workload Partition can have. The WPS is equal to the Global Limiting Privilege Set (LPS).

Considerations for the Workload Partition environment

The following considerations apply to using RBAC with a Workload Partition (see Figure 8-14 on page 334):

- The RBAC mode is only configurable in the Global Environment. The setting in the Global Environment applies to all Workload Partitions on the system
- Application Workload Partitions do not have the Workload Partition Privilege Set (WPS). The entire set of privileges will be assigned to root owned processes on Application Workload Partition, as is the same case for Global.
- The system defined authorizations are contained in the Global Environment.
- Each Workload Partition has its own Workload Partition user-defined KAT.
- Any Workload Partition has a privilege limited by the use of the Workload Partition Privilege Set (WPS). To extend the privilege set, use the **chwpar** command as follows:

```
chwpar -S privs+=privileges wpar_name
```

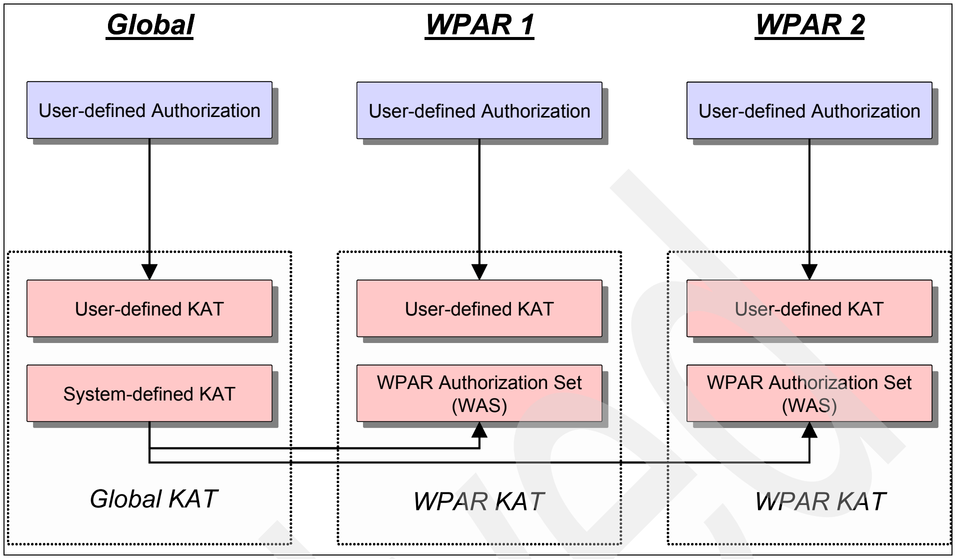


Figure 8-14 Kernel Authorization Tables mapping for Workload Partitions

8.7 Enhanced and existing mode switch

In order to disable the enhanced RBAC capabilities and to revert back to the existing RBAC behavior, a system wide configuration switch is provided. You select the option to not use the enhanced RBAC features through a system wide configuration switch in the kernel, which denotes that the Enhanced RBAC Mode is disabled for the system. A system administrator may select this mode by invoking the **chdev** command on the sys0 device and specifying the enhanced_RBAC attribute with a value of false and then rebooting the system. The mode can be switched back to Enhanced RBAC Mode by setting the enhanced_RBAC attribute to true. Programmatically, the mode can be set or queried through the sys_parm() system call. An example invocation of the **chdev** command is shown here:

```
chdev -l sys0 -a enhanced_RBAC=false
```

In order to list the value of the attribute, run:

```
lsattr -E -l sys0 -a enhanced_RBAC
```

In a Workload Partition environment, this mode will only be configurable from the global system and will affect the global as well as all the Workload Partitions. Both the new and existing interfaces will be modified to check this configuration and either execute the new code or follow the old behavior based on the value of the switch. In Legacy RBAC Mode, only authorizations that are checked within the code of the command itself will be enforced. The Kernel Security Tables (KST) will not have any affect on command execution or authorization checks. The determination of whether a user has an authorization will follow the existing behavior of retrieving all the user's authorizations and checking if there is a match. New features being added like **`swrole`** and the default_roles and auth_mode attributes will not be available in Legacy RBAC Mode. However, the new privileges, authorizations, and management commands for authorizations will be supported in Legacy RBAC Mode.

Note: Be aware that disabling the enhanced RBAC feature may lower the security threshold of your system, especially in Workload Partition. The enhanced RBAC option only works under the 64-bit kernel.

8.8 Trusted AIX

Trusted AIX enables Multi Level Security (MLS) capabilities in AIX. As compared to regular AIX, Trusted AIX label-based security implements labels for all subjects and objects in the system.

Note: The Trusted AIX install option enables the Labeled Security AIX environment. Access controls in the system are based on labels that provide for a Multi Level Security (MLS) environment and includes support for the following:

- Labeled objects: Files, IPC objects, network packets, and other labeled objects
- Labeled printers
- Trusted Network: Support for RIPSO and CIPSO in IPv4 and IPv6

Note that once you choose this mode of installation, you will not be able to go back to a regular AIX environment without performing an overwrite install of regular AIX. Evaluate your need for a Trusted AIX environment before choosing this mode of install. More details about Trusted AIX can be found in the AIX publicly available documentation.

Standard AIX provides a set of security features to allow information managers and administrators to provide a basic level of system and network security. The primary AIX security features include the following:

- Login and password controlled system and network access
- User, group, and world file access permissions
- Access control lists (ACLs)
- Audit subsystem
- Role Based Access Control (RBAC)

Trusted AIX builds upon these primary AIX operating system security features to further enhance and extend AIX security into the networking subsystems.

Trusted AIX is compatible with the AIX application programming interface (API). Any application that runs on AIX can also run on Trusted AIX. However, due to additional security restrictions, MLS-unaware applications may need privileges to operate in a Trusted AIX environment. The `tracepriv` command can be used to profile applications in such scenarios.

Trusted AIX extends the AIX API to support additional security functionality. This allows customers to develop their own secure applications that can be developed using the AIX API and new Trusted AIX extensions.

Trusted AIX enables AIX systems to process information at multiple security levels. It is designed to meet the US Department of Defense (DoD) TCSEC and European ITSEC criteria for enhanced B1 security.

8.8.1 Introduction

Trusted AIX enhances system security through four primary elements of information security:

- Confidentiality
- Integrity
- Availability
- Accountability

In addition to the security features provided by AIX, Trusted AIX adds the following capabilities:

Sensitivity labels (SLs) All processes and files are labeled according to their security level. Processes can only access objects that are within the process' security range.

Integrity labels (TLs) All processes and files are labeled according to their integrity level. Files cannot be written by processes that have a lower integrity level label than the file. Processes cannot read from files that have a lower integrity level label than the process.

File security flags Individual files can have additional flags to control security related operations.

Kernel security flags The entire system can have different security features enabled or disabled.

Privileges Many commands and system calls are only available to processes with specific privileges.

Authorizations Each user can be granted a unique set of authorizations. Each authorization allows the user to perform specific security-related functions. Authorizations are assigned to users through roles.

Roles Role Based Access Control function, as part of Trusted AIX, provides for selective delegation of administrative duties to non-root users. This delegation is achieved by collecting the relevant authorizations into a Role and then assigning the role to a non-root user.

Confidentiality

Threats centered around disclosure of information to unauthorized parties are a confidentiality issue.

Trusted AIX provides object reuse and access control mechanisms for protecting all data resources. The operating system ensures that protected data resources can only be accessed by specifically authorized users and that those users cannot make the protected resources available to unauthorized users either deliberately or accidentally.

Administrators can prevent sensitive files from being written to removable media, from being printed on unprotected printers, or from being transferred over a network to unauthorized remote systems. This security protection is enforced by the operating system and cannot be bypassed by malicious users or rogue processes.

Integrity

Threats centered around modification of information by unauthorized parties are an integrity issue.

Trusted AIX offers numerous security mechanisms that ensure the integrity of trusted computing base and protected data, whether the data is generated on the system or imported using network resources. Various access control security mechanisms ensure that only authorized individuals can modify information. To prevent malicious users or rogue processes from seizing or disabling system resources, Trusted AIX eliminates the root privilege. Special administrative authorizations and roles allow the separation of administration duties, rather than giving a user root privileges.

Availability

Threats centered around accessibility of services on a host machine are an availability issue. For example, if a malicious program fills up file space so that a new file cannot be created, there is still access, but no availability.

Trusted AIX protects the system from attacks by unauthorized users and processes that can create a denial of service. Unprivileged processes are not allowed to read or write protected files and directories.

Accountability

Threats centered around not knowing which processes performed which actions on a system are an accountability issue. For example, if the user or process that altered a system file cannot be traced, you cannot determine how to stop such actions in the future.

This enhanced security feature ensures identification and authentication of all users prior to allowing user access to the system. The audit services provide the administrator a set of auditable events and an audit trail of all security-related system events.

8.8.2 Considerations

The following are the major considerations pertaining to Trusted AIX:

- Trusted AIX is installed through the AIX install menus. Additional options can be chosen during installation of Trusted AIX. The option related to LSPP EAL4+ configuration is supported.
- A Trusted AIX environment cannot revert to regular AIX environment without performing an overwrite installation of regular AIX.
- Root is disabled from logging in a Trusted AIX environment.

- In a Trusted AIX environment, any WPARs created will also operate in the Labeled Security environment.
- Trusted AIX supports both Mandatory Access Control (MAC) and Mandatory Integrity Control (MIC). A customer can define separate sets of labels for MAC and MIC.
- Label Encodings file is located in the /etc/security/enc directory and captures the label-to-binary translation information. The default Label Encodings file adheres to the Compartmented Mode Workstations (CMW) labels-related naming requirements.
- NIM installs are supported when initiated from the client. A NIM install push from the server is not possible because root is disabled for logins on MLS systems.
- The JFS2 (J2) file system (using Extended Attributes Version 2) has been enabled for storing labels in AIX. Other file systems (such as J1 or NFS) can only be mounted in a Trusted AIX environment as single-level file systems (label assigned to the mount point).
- The X Window System environment is disabled for Trusted AIX.
- Trusted AIX supports CIPSO and RIPSO protocols for network-based label-based communication. These protocols are supported for both IPv4 and IPv6.
- Some AIX security mechanisms are common between regular AIX and Trusted AIX. Two of these common security mechanisms are Role Based Access Control (RBAC) and Trusted Execution for integrity verification.
- Since root is disabled when Trusted AIX is installed, the installer must set up passwords for ISSO, SA, and SO users during the first boot after install. The system remains usable until these passwords are created.

For installation and configuration, see *AIX V6 Advanced Security Features Introduction and Configuration*, SG24-7430.

8.8.3 Identification and authentication

Identification and authentication security mechanisms are responsible for ensuring that each individual requesting access to the system is properly identified and authenticated. Identification requires a user name and authentication requires a password

All Trusted AIX accounts are password protected. The Information Systems Security Officer (ISSO) can configure the system to allow a user to select his/her own password, subject to password length and complexity constraints. The ISSO can also specify minimum and maximum password aging parameters (expiration periods) on a per-user basis, including warning periods prior to password expiration.

The identification and authentication security mechanisms require that all user names and user IDs be unique. Accounts without valid passwords cannot be used for login. A user with the ISSO role must add the initial password for all new users. Each user is assigned an additional unique identifier that is used for auditing purposes.

Only the encrypted form of the password is stored. Passwords are not stored on the system in plain text form. The encrypted passwords are stored in a shadow password file, which is protected against access except by privileged processes. For more information, see the `passwd` command.

Trusted AIX systems recognize two types of accounts: system accounts and user accounts. System accounts are those with a user ID less than 128. Although system accounts may have associated passwords, they cannot be used for logging on to the system.

8.8.4 Discretionary access control

Discretionary access controls (DAC) are the security aspects that are under the control of the file or directory owner.

UNIX permissions

A user with owner access to a resource can do the following:

- Directly grant access to other users.
- Grant access to a copy to other users.
- Provide a program to allow access to the original resource (for example, using SUID programs).

The traditional UNIX permission bit method (owner/group/other and read/write/execute) is an example of this DAC functionality.

Permission bits enable users to grant or deny access to the data in a file to users and groups (based on the need-to-know criterion). This type of access is based on the user ID and the groups to which the user belongs. All file system objects have associated permissions to describe access for the owner, group, and world.

The owner of a file can also grant access privileges to other users by changing the ownership or group of the file with the **chown** and **chgrp** commands.

umask

When a file is created, all permission bits are initially turned on. The file then has certain permission bits removed by the umask process, which has been set during the login process. The default umask applies to every file created by the user's shell and every command run from the user's shell.

By default, the umask setting for kernel items is 000 (which leaves all permissions available to all users). AIX sets the kernel umask to 022 (which turns off group and world write permission bits). However, users may override this setting if needed.

> **Note:** Be very cautious about changing the umask to a setting more permissive than 022. If more permissions are available on files and processes, the system as a whole becomes less secure.

There are two methods to override the default umask setting:

- You can change the umask values in your .profile, .login, or .chsrc files. These changes will affect any file that is created during your login session.
- You can set the umask levels for individual processes with the umask command. After running the **umask** command, all new files that are created will be affected by the new umask value until one of the following two events occur:
  - You run the **umask** command again.

    OR
  - You exit the shell in which the **umask** command was issued.

If you run the **umask** command with no arguments, the **umask** command returns the current umask value for your session.

You should allow the login session to inherit the kernel's 022 umask value by not specifying a umask in your profiles. Umask values less restrictive than 022 should only be used with great caution.

If additional permissions are needed for certain files, these permissions should be set with judicious use of the **chmod** command after the files have been created.

Access Control Lists

In addition to the standard UNIX permission bits and umask value, AIX also supports access control lists (ACLs).

UNIX permission bits only control access for the file owner, one group, and everyone on the system. With an ACL, a file owner can specify access rights for additional specific users and groups. Like permission bits, ACLs are associated with individual system objects, such as file or directory.

The setuid and setgid command permission bits

The setuid and setgid permission bits (set user ID and set group ID) allow a program file to run with the user ID or group ID of the file owner rather than the user ID or group ID of the person who is running the program. This task is accomplished by setting the setuid and setgid bits that are associated with the file. This permits the development of protected subsystems, where users can access and run certain files without having to own the files.

If the setgid bit is set on a parent directory when an object is created, the new object will have the same group as the parent directory, rather than the group of the object's creator. However, objects created in a directory with the setuid bit set are owned by the object's creator, not the directory owner. The setuid/setgid bits of the parent directory are inherited by subdirectories when subdirectories are created.

The setuid and setgid permission bits represent a potential security risk. A program that is set to run with root as the owner could have essentially unlimited access to the system. On Trusted AIX systems, however, the use of privileges and other access controls significantly reduces this security risk.

8.8.5 Role Based Access Control elements

Trusted AIX supports Role Based Access Control (RBAC). RBAC is an operating system mechanism through which the root/system super user specific system functions can also be performed by regular users using the roles that are assigned to them.

The core elements of AIX RBAC are:

Authorizations These strings indicate the privilege operation that they represent and control by name directly. For example, an authorization string aix.network.manage defines the network management function in AIX.

Privileges A privilege is an attribute of a process that allows the process to bypass specific system restrictions and limitations. Privileges are associated with a process and are typically acquired through the execution of a privileged command.

Roles Role elements in AIX RBAC allow users to combine a set of management functions in the system and assign these functions to be managed by a regular user. Roles in AIX consist of a collection of authorizations (these can be both system authorizations as well as custom authorizations) and any other roles (as sub roles).

The authorizations, roles, and privileges introduced for RBAC require additions and modifications for Trusted AIX. These authorizations and privileges are only active in a Trusted AIX environment.

Table 8-9 provides the authorizations that are active in a Trusted AIX system.

Table 8-9 Trusted AIX authorizations

Trusted AIX authorization	Description
aix.mls.lef	Validate LEF file. (**labck**)
aix.mls.pdir.create	Create partition directories. (**pdmkdir**)
aix.mls.pdir.remove	Delete partition directories. (**pdrmdir**)
aix.mls.pdir.link	Create inks in partition directories. (**pdlink**)
aix.mls.pdir.set	Convert regular directories to partition directories. (**pdset**)
aix.mls.pdir.mode	Switch to real mode to access partition directories. (**pdmode**)
aix.mls.label.sl	Change SL of file system objects. (**setsecattr**)
aix.mls.label.sl.downgrade	Downgrade SL of file system objects. (**setsecattr**)
aix.mls.label.sl.upgrade	Upgrade SL of file system objects. (**setsecattr**)
aix.mls.label.outsideaccred	Use labels outside the accreditation range of the system.
aix.mls.label.tl	Change TL of file system objects. (**setsecattr**)

Trusted AIX authorization	Description
aix.mls.label.tl.downgrade	Downgrade TL of file system objects. (**setsecattr**)
aix.mls.label.tl.upgrade	Upgrade TL of file system objects. (**setsecattr**)
aix.mls.stat	View label attributes of file system objects.
aix.mls.network.init	Initialize the trusted network sub-system and maintain the trusted network rules database.
aix.mls.network.config	Command for adding, removing, listing, or querying rules, flags, and security labels for interfaces and hosts.
aix.mls.proc.sl aix.mls.proc.sl.downgrade aix.mls.proc.sl.upgrade aix.mls.proc.stat	Change SL of Processes. Downgrade SL of Processes. Upgrade SL of Processes. View Label Attributes of Processes.
aix.mls.proc.tl aix.mls.proc.tl.downgrade aix.mls.proc.tl.upgrade	Change TL of Processes. Downgrade TL of Processes. Upgrade TL of Processes.
aix.mls.system.config.write	Modify MLS kernel flags. (**setsecconf**)
aix.mls.system.config.read	Read MLS Kernel flags. (**getsecconf**)
aix.mls.system.label.read	Read System labels. (**getsyslab**)
aix.mls.system.label.write	Modify System labels. (**setsyslab**)
aix.mls.tpath	Trusted Path administration. (**tlibadmin**)
aix.mls.clear.read	Read clearance attributes of users. (**lsuser**)
aix.mls.clear.write	Modify clearance attributes of users. (**chuser**)
aix.mls.login	Allow login on restricted consoles.
aix.mls.system.mode	Allows to switch the runmode. (**setrunmode**)

Users, roles, and authorizations

A Trusted AIX installation requires three administrative roles: Information System Security Officer (ISSO), System Administrator (SA), and System Operator (SO).

Table 8-10 on page 345 shows the roles and authorizations map.

Table 8-10 Relations between authorizations and roles

Authorizations	ISSO	ISSO with MLS	SA	SO	SO with MLS
aix.device	-	-	-	-	-
aix.device.config.printer	-	-	-	OK	OK
aix.device.config.random	-	-	-	OK	OK
aix.fs	-	-	OK	-	-
aix.fs.manage.backup	-	-		OK	OK
aix.fs.manage.export	OK	OK		-	-
aix.fs.manage.mount	-	-		OK	OK
aix.fs.manage.quota	-	-		OK	OK
aix.fs.manage.recover	-	-		OK	OK
aix.fs.manage.unmount	-	-		OK	OK
aix.fs.object.create	OK	OK		-	-
aix.fs.object.list	OK	OK		-	-
aix.network.config.arp	OK	OK	-	-	-
aix.network.config.host	OK	OK	-	-	-
aix.network.config.mail	-	-	-	OK	OK
aix.network.config.no	OK	OK	-	-	-
aix.network.config.route	OK	OK	-	-	-
aix.network.config.tcpip	OK	OK	-	-	-
aix.network.status	-	-	-	OK	OK
aix.proc.kill	-	-	-	OK	OK
aix.proc.status	OK	OK	-	-	-
aix.ras.audit	OK	OK	-	-	-
aix.ras.audit.config	OK	OK	-	-	-
aix.security.group\	OK	OK	-	-	-
aix.security.passwd	OK	OK	-	-	-

Authorizations	ISSO	ISSO with MLS	SA	SO	SO with MLS
aix.security.role	OK	OK	-	-	-
aix.system.boot\	OK	OK	-	-	-
aix.system.boot.halt	-	-	-	OK	OK
aix.system.boot.reboot	-	-	-	OK	OK
aix.system.boot.shutdown	-	-	-	OK	OK
aix.system.config.init	-	-	-	OK	OK
aix.system.config.cron	-	-	OK	-	-
aix.system.config.date	OK	OK	-	-	-
aix.system.config.src	OK	OK	-	-	-
aix.system.config.uname	OK	OK	-	-	-
aix.system.config.wlm	-	-	-	OK	OK
aix.security.tsd	OK	OK	-	-	-
aix.mls.clear	-	OK	-	-	-
aix.mls.network.config	-	OK	-	-	-
aix.mls.network.init	-	OK	-	-	-
aix.mls.network.config	-	OK	-	-	-
aix.mls.label	-	OK	-	-	-
aix.mls.lef	-	OK	-	-	-
aix.mls.login	-	OK	-	-	OK
aix.mls.pdir	-	OK	-	-	-
aix.mls.proc	-	OK	-	-	-
aix.mls.stat	-	OK	-	-	-
aix.mls.system.config	-	OK	-	-	-
aix.mls.system.label	-	OK	-	-	-
aix.mls.tpath	-	OK	-	-	-

8.8.6 Trusted AIX packages

Table 8-11 provides the filesets that are installed as part of a Trusted AIX installation.

Table 8-11 Filesets installed in a Trusted AIX environment

Fileset	Remarks
bos.mls.rte	MLS commands are packaged into this fileset.
bos.mls.lib	This is a place holder.
bos.mls.adt	The SDK for MLS is packaged into this fileset.
bos.mls.cfg	MLS configuration files are packaged into this fileset.
bos.mls.smit	SMIT tools and dialogs related to MLS are packaged into this fileset.

The bos.mls.lib is added to BOS.autoi, SbD.BOS.autoi, and CCEVAL.BOS.autoi, and will be always installed during BOS installation.

The bos.mls.rte fileset creates three default administrative users: isso, sa, and so, with default roles ISSO, SA, and SO, respectively, as part of the pre-install scripts (bos.mls.rte.pre_i script) for the fileset.

Note: You are prompted to enter the passwords for these users by the install assistant after the first boot. **Set the ISSO, SA, and SO Passwords** replaces **Set root Password** under the Installation Assistant Main Menu. If the system is installed by the non-prompted install method, the passwords are hardcoded to be same as the default user names.

The following entries will be added to inittab using the bos.mls.rte.config script:

- rc.mls.boot
- rc.mls.net
- rc.mls

8.8.7 Trusted AIX commands

The following security-related commands are provided to manage a Trusted AIX system:

labck	Verifies a LabelEncodings file.
getsecconf	Displays the kernel security flags.
setsecconf	Changes the Trusted AIX kernel security flags.
getsyslab	Shows the kernel maximum and minimum labels.
setsyslab	Sets the kernel maximum and minimum labels.
getrunmode	Displays the current running mode of the system.
setrunmode	Switches the running mode of the system.
pdlink	Links files across partitioned subdirectories.
pdmkdir	Creates partitioned directories and subdirectories.
pdmode	Returns the current partitioned directory access mode or runs a command with a specified partitioned directory access mode.
pdrmdir	Removes partitioned directories and the associated subdirectories.
pdset	Sets/unsets partitioned (sub)directories.
bootauth	Verifies that an authorized user is booting the system.
chuser	Changes the user's clearance attributes.
lsuser	Displays the user's clearance attributes.
chsec	Changes the user's clearance attributes and port labels.
lssec	Displays the user's clearance attributes and port labels.
trustchk	Checks the attributes of files.
lstxattr	Displays the label and security flag attributes of files, processes, and IPC objects.
settxattr	Changes the label and security flag attributes of files, processes, and IPC objects.

8.9 The Trusted Execution environment

The Trusted Execution (TE) mechanism is new in AIX V6.1, and enhances the AIX security environment. Trusted Execution refers to a collection of features that are used to verify the integrity of the system and implement advance security policies, which together can be used to enhance the trust level of the complete system.

This new component introduces a new command to verify a system's integrity while the Trusted Computing Base (TCB) is still available as an alternative. Unlike the TCB, which maintains checksums for crucial files and verifies them periodically (either triggered by cron or CLI), TE does such *offline checking* as well, but also allows for checking a file's integrity at its execution time, every time.

TE refers to a collection of features that are used to verify the integrity of the system's trusted computing base that, in the context of TE, is called the Trusted Signature Database (TSD). In addition, TE implements advanced security policies, which together can be used to enhance the trust level of the complete system. The usual way for a malicious user to harm the system is to get access to the system and then install trojan horses, rootkits, or tamper with some security critical files such that the system becomes vulnerable and exploitable.

The central idea behind the set of features under TE is to be able to prevent such activities or in the worst case be able to identify if any such thing happens to the system. Using the functionality provided by TE, the system administrator can decide upon the actual set of executables that are allowed to execute or the set of kernel extensions that are allowed to be loaded. It can also be used to audit the security state of the system and identify files that have changed, thereby increasing the trusted level of the system and making it more difficult for the malicious user to do harm to the system.

In order for TE to work, the CryptoLight for C library (CLiC) and kernel extension need to be installed and loaded on your system. These filesets are included on the AIX Expansion Pack and are provided at no charge. To check whether they are installed on your system and loaded into the kernel, run the command shown in Example 8-2.

Example 8-2 CLiC filesets

```
# lslpp -l "clic*"
Fileset                      Level  State      Description
  ----------------------------------------------------------------------------
Path: /usr/lib/objrepos
  clic.rte.includes          4.3.0.0  COMMITTED  CryptoLite for C Library
                                                 Include File
  clic.rte.kernext           4.3.0.0  COMMITTED  CryptoLite for C Kernel
  clic.rte.lib               4.3.0.0  COMMITTED  CryptoLite for C Library
  clic.rte.pkcs11            4.3.0.0  COMMITTED  PKCS11 Software Token Support

Path: /etc/objrepos
  clic.rte.kernext           4.3.0.0  COMMITTED  CryptoLite for C Kernel
```

This section describes:

- Trusted Signature Database
- Trusted Execution
- Trusted Execution Path and Trusted Library Path

For auditing and configurations of Trusted Execution, see *AIX V6 Advanced Security Features Introduction and Configuration*, SG24-7430.

8.9.1 Trusted Signature Database

The File Verification mechanism is similar to the existing Trusted Computing Base (TCB) subsystem in certain aspects. While TCB verifies the integrity of the file using check sum values, this mechanism will ensure the trust of the files using hash signatures. These signatures are actually a mathematical hash of the file data. By default, SHA-256 is used as the hashing algorithm; however, the system owner has the option to configure the hashing algorithm from a list of supported algorithms. Since every file has its own unique hash value, a database is needed on the system to store these values. This is in line with TCB, which uses a file named /etc/security/sysck.cfg to store the check sum values. Similarly, a new data file /etc/security/tsd/tsd.dat is introduced to serve as the database for storing different security attributes, like the hash values for the trusted files, as shown in Example 8-3 on page 351.

Example 8-3 Example of stanza of ksh command in the TSD

```
/usr/bin/ksh:
        owner = bin
        group = bin
        mode = TCB,555
        type = FILE
        hardlinks = /usr/bin/sh,/usr/bin/psh,/usr/bin/tsh,/usr/bin/rksh
        symlinks =
        size = 288056
        cert_tag = 00af4b62b878aa47f7
        signature = 27af0e83720a1700a0e96c79ce1944b2e71677be565275a29dacd0ad9504
8c15cda82b3108e4a361193629f788f98f343ee49ad5ae51b9f2bd9e4a0e37fe020f30038967aa14
251c92e36bc912608a63adbad0340749a5eaf003989a977ff2e2c65f73482864cef0e1b5ba36e20c
064a92854a6200af8d0bba556ebb9c08271a
        hash_value = 293b40b6d138daaa5746539f37e1d4e45eba613be6112f5b8f3c69560c8
c306e
        minslabel =
        maxslabel =
        intlabel =
        accessauths =
        innateprivs =
        inheritprivs =
...
```

To learn more about managing the Trusted Signature Database, refer to “Trusted Signature Database” in the AIX V6.1 online manual, or “Signature creation and deployment” in *AIX V6 Advanced Security Features Introduction and Configuration*, SG24-7430.

8.9.2 Trusted Execution

Trusted Execution provides a new command to verify the integrity of the system. The **trustchk** command has the following two methods for integrity checking:

- System integrity check
- Runtime integrity check

System integrity check

System integrity check is a static method to check integrity. It is executed when the **trustchk** command is executed on the command line, by **cron**, or from the **rc.mls.boot** script at boot time. The Trusted Signature Database (TSD) (/etc/security/tsd/tsd.dat) and the certificates (/etc/security/certificates/*) are used to check integrity. This database and tis certificates are created when the trusted system is installed.

An overview of the system integrity check is shown in Figure 8-15.

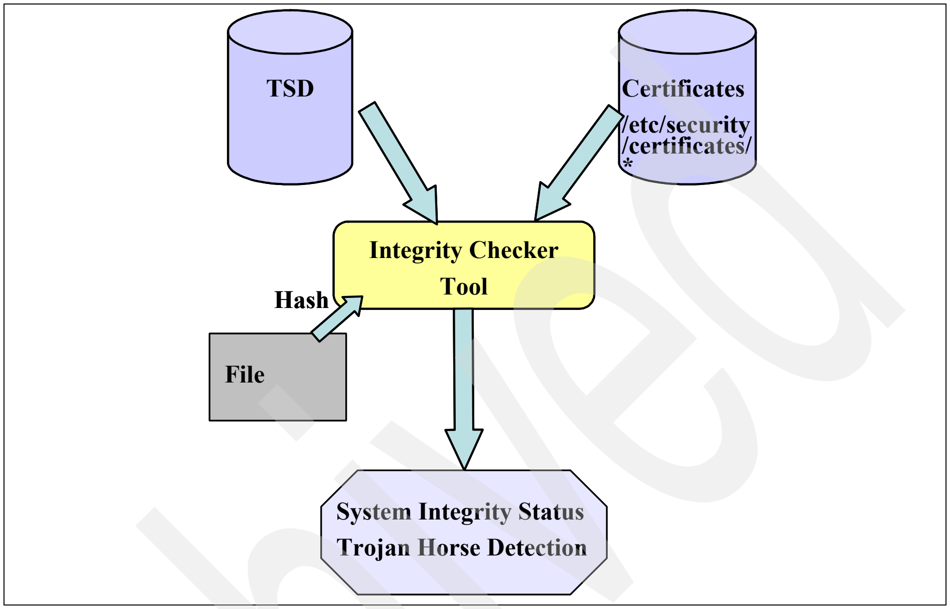


Figure 8-15 System integrity check

Note: An administrator does this check on a normal system by running manually the **`trustchk -t ALL`** command; these automatic checks are done only on an MLS system (Trusted AIX).

Runtime integrity check

The Trusted Execution feature provides you with a runtime file integrity verification mechanism. Using this mechanism, the system can be configured to check the integrity of the trusted files before every request to access those files, effectively allowing only the trusted files that pass the integrity check to be accessed on the system (Figure 8-16 on page 353).

When a file is marked as trusted (by adding its definition to Trusted Signature Database), the Trusted Execution feature can be made to monitor its integrity on every access. Trusted Execution can continuously monitor the system and is capable of detecting tampering of any trusted file (by a malicious user or application) present on the system at runtime (for example, at load time). If the file is found to be tampered, Trusted Execution can take corrective actions based on pre-configured policies, such as disallow execution, access to the file, or logging an error. If a file is being opened or executed, and has an entry in the Trusted Signature Database (TSD), the Trusted Execution performs as follows:

- Before loading the binary, the component responsible for loading the file (system loader) invokes the Trusted Execution subsystem, and calculates the hash value using the SHA-256 algorithm (configurable).
- This runtime calculated hash value is matched with the one stored in the TSD.
- If the values match, the file opens or executes.
- If the values do not match, either the binary has been tampered with or somehow compromised. It is up to the user to decide the action to be taken. The Trusted Execution mechanism provides options for users to configure their own policies for the actions to be taken if the hash values do not match.
- Based on these configured policies, a relevant action is taken.

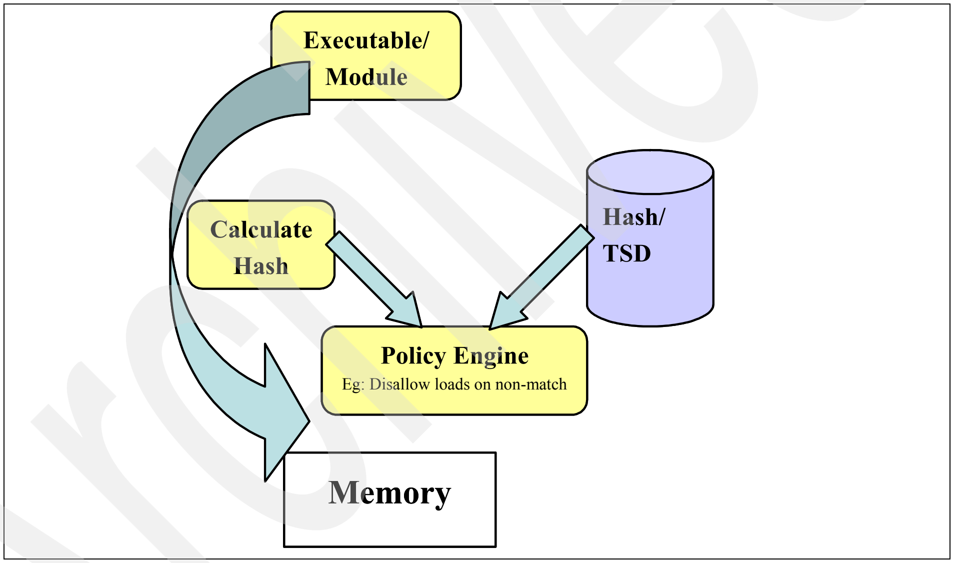


Figure 8-16 Runtime integrity check

8.9.3 Trusted Execution Path and Trusted Library Path

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

```
# trustchk -p tep
TEP=OFF
TEP=/usr/bin:/usr/sbin
# trustchk -p
tep=/usr/bin:/usr/sbin:/etc:/bin:/sbin:/usr/lib/instl:/usr/ccs/bin
# trustchk -p tep
TEP=OFF
TEP=/usr/bin:/usr/sbin:/etc:/bin:/sbin:/usr/lib/instl:/usr/ccs/bin
# trustchk -p tep=on
# trustchk -p tep
TEP=ON
```

The Trusted Library Path (TLP) has the same functionality as TEP; the only difference is that it is used to define the directories that contain trusted libraries of the system. Once TLP is enabled, the system loader will allow only the libraries from this path to be linked to the binaries. The **trustchk** command can be used to enable and disable the TEP/TLP as well as to set the colon-separated path list for both using TEP and TLP command-line attributes of **trustchk**:

```
# trustchk -p tlp
TLP=OFF
TLP=/usr/lib:/usr/ccs/lib:/lib:/var/lib
```

TLP uses a flag to control its operations: FSF_TLIB. If the file has the FSF_TLIB flag set in its TSD stanza, then the process resulting from it will be set as a TLIB process. Processes marked as TLIB processes can link only to *.so libraries that also have the TLIB flag set.

8.10 Password length and encryption algorithms

Recent advancements in computer hardware makes the traditional UNIX password encryption vulnerable to brute force password guessing attacks. A cryptographically weak algorithm can lead to the recovery of even strong passwords. AIX V6.1 and AIX 5L V5.3 TL7 introduces Loadable Password Algorithm (LPA). It also removes the eight character password limitation.

8.10.1 Existing crypt()

The AIX standard authentication mechanism authenticates users using a one-way hash function called crypt(). crypt() is a modified DES algorithm. It performs a one-way encryption of a fixed data array with the supplied password and a *Salt*.

crypt() uses only the first eight characters from the password string. The user's password is truncated to eight characters. If the password is shorter than eight characters, it is padded with zero bits on the right. The 56-bit DES key is derived by using the seven bits from each character.

Salt is a two-character string (12 bits of the Salt is used to perturb the DES algorithm) chosen from the character set "A-Z", "a-z","0-9","."(period) and "/". Salt is used to vary the hashing algorithm, so that the same clear text password can produce 4,096 possible password encryptions. A modification to the DES algorithm, swapping bits i and i+24 in the DES E-Box output when bit i is set in the Salt, achieves this while also making DES encryption hardware useless to password guessing.

The 64-bit all-bits-zero block is encrypted 25 times with the DES key. The final output is the 12-bit salt concatenated with the encrypted 64-bit value. The resulting 76-bit value is recoded into 13 printable ASCII characters in the form of base64.

8.10.2 Password hashing algorithms

The hashing algorithms, like MD5, are harder to break than crypt(). This provides a strong mechanism against brute force password guessing attacks. Since the whole password is used for generating the hash, there is no password length limitation when we use the password hashing algorithms to encrypt the password.

8.10.3 Loadable Password Algorithm

AIX V6.1 and AIX 5L V5.3 TL7 implemented the Loadable Password Algorithm (LPA) mechanism, which can easily deploy new password encryption algorithms.

Each supported password encryption algorithm is implemented as an LPA load module that is loaded at runtime when the algorithm is needed. The supported LPAs, and their attributes, are defined in the system configuration file /etc/security/pwdalg.cfg.

The administrator can set up a system wide password encryption mechanism that uses a specific LPA to encrypt the passwords. After the system wide password mechanism changed, AIX V6.1 and AIX 5L V5.3 TL7 still support the passwords that were encrypted by the previous selected password encryption mechanisms, like the crypt() function.

The MD5, SHA, and Blowfish password algorithms are implemented as LPAs.

The Loadable Password Algorithm (LPA) supports:

- New secure password hash algorithms
- Greater than eight character passwords
- More valid characters in passwords

8.10.4 Support greater than eight character passwords

All the LPAs implemented for AIX V6.1 and AIX 5L V5.3 TL7 support passwords longer than eight characters. The password length limitations are different from LPA to LPA. The maximum length of password supported by AIX V6.1 and AIX 5L V5.3 TL7 is 255.

8.10.5 LPA configuration file

The LPA configuration file is /etc/security/pwdalg.cfg. It is a stanza file that defines the attributes of supported LPAs.

The attributes of an LPA that are defined in the config file include:

- The path to the LPA module
- The optional flags that are passed to the LPA module at runtime

The attribute of the LPA defined in the configuration file can be accessed through the getconfattr() and setconfattr() interfaces.

The following example stanza in /etc/security/pwdalg.cfg defines a LPA named "ssha256":

```
ssha256:
        lpa_module = /usr/lib/security/ssha
        lpa_options = algorithm=sha256
```

8.10.6 System password algorithm

The system administrator can set a system wide password algorithm by selecting an LPA as the password hashing algorithm. There will be only one active system password algorithm at a time. The system password algorithm is defined by a system attribute, pwd_algorithm, in the stanza of usw in the /etc/security/login.cfg file.

The valid values for the pwd_algorithm attribute in /etc/security/login.cfg are LPA stanza names that are defined in the /etc/security/pwdalg.cfg file. Another valid value for the pwd_algorithm attribute is crypt, which refers to the crypt() encryption. If the pwd_algorithm attribute is omitted from the config file, crypt is used as the default value.

The following example of /etc/security/login.cfg shows that the administrator chose to use the "ssha256" LPA as the system wide password encryption algorithm:

```
usw:
        shells =
/bin/sh,/bin/bsh,/bin/csh,/bin/ksh,/bin/tsh,/bin/ksh93,/usr/bin/sh,/usr
/bin/bsh,/usr/bin/csh,/usr/bin/ksh,/usr/bin/tsh,/usr/bin/ksh93,/usr/bin
/rksh,/usr/bin/rksh93,/usr/sbin/uucp/uucico,/usr/sbin/sliplogin,/usr/sb
in/snappd
        maxlogins = 32767
        logintimeout = 60
        maxroles = 8
        auth_type = STD_AUTH
        pwd_algorithm = ssha256
```

The system password algorithm takes effect only on the newly created passwords and changed passwords. After the migration, all subsequent new passwords or password changes will be done using the system password algorithm. The existing passwords before the system password algorithm is chosen, either generated by the standard crypt() or other supported LPA modules, still work on the system. Therefore, mixed passwords that have been generated by different LPAs may coexist on the system.

New secure password hash algorithms

Table 8-12 lists all the supported algorithms and their characteristics.

Table 8-12 Algorithms and their characteristics

Algorithm	Maximum password length	Length of Salt, base 64	Iterations	Length of hashed string, base 64	Maximum Length of Hashed Password, base 64
crypt	8	2-char (12-bit)	25 (built-in)	11-char (64-bit)	13-char (76-bit)
MD5	255	2 to 8-char (48-bit)	1000 (built-in)	22-char (128-bit)	37-char ({smd5}salt$hashed_str)
SHA1	255	8 to 24-char	2^4 to 2^{31} (cost is 4 to 31)	27-char (160-bit)	62-char ({ssha1}nn$salt$hashed_str)
SHA256	255	8 to 24-char	2^4 to 2^{31} (cost is 4 to 31)	43-char (256-bit)	80-char ({ssha256}nn$salt$hashed_str)
SHA512	255	8 to 24-char	2^4 to 2^{31} (cost is 4 to 31)	86-char (512-bit)	123-char ({ssha256}nn$salt$hashed_str)
Blowfish	72	22-char	2^4 to 2^{31} (cost is 4 to 31)	32-char (192-bit)	69-char ({ssha256}nn$salt$hashed_str)

8.10.7 Support more valid characters in passwords

For the nature of the crypt() algorithm, all the characters (>0x80) in the extended ASCII table are not allowed to be in passwords.

Most of the hashing algorithms, like MD5 and SHA, support binary data. Therefore, the characters in the extended ASCII table are allowed in passwords for these new algorithms. The space character is allowed in passwords as well.

8.10.8 Setup system password algorithm

The system administrator can set up the system password algorithm using the **chsec** command, or by manually modifying the pwd_algorithm attribute in /etc/security/login.cfg using an editor such as vi.

We recommend using the **chsec** command to set the system password algorithm, because the command automatically checks the definition of the chosen LPA.

Using the chsec command

Use the following **chsec** command to set the "smd5" LPA as the system wide password encryption module:

```
chsec -f /etc/security/login.cfg -s usw -a pwd_algorithm=smd5
```

When using the **chsec** command to modify the pwd_algorithm attribute, the command checks /etc/security/pwdalg.cfg to verify the chosen LPA. The command fails if the check fails.

Using editor

When an administrator manually changes the pwd_algorithm attribute value in /etc/security/login.cfg using an editor, please make sure that the chosen value is the name of a stanza that is defined in the /etc/security/pwdalg.cfg file.

8.10.9 Changes to support long passwords

The following changes were made in order to support longer passwords.

Changes to limits.h

The previous AIX definition of PASS_MAX is in limits.h (see Example 8-4).

Example 8-4 Password MAX Limit (Before AIX 5L V5.3 TL7)

```
#define PASS_MAX        32
```

The new PASS_MAX is defined as 255 (see Example 8-5).

Example 8-5 Password MAX Limit (AIX 5L V5.3 TL7 and AIX V6.1)

```
#define PASS_MAX        255
```

Changes to userpw.h

The userpw.h file defines password related manifest constants and macros.

Table 8-13 provides the symbols that are used for determining the maximum possible sizes when declaring arrays, memory, and so on.

Table 8-13 Summary of changes to userpw.h

Macro name	Definition	Existing value	New values (AIX V6.1)
MAXIMPL_PW_PASSLEN	Password length in chars	9	256
MAXIMPL_PW_CRYPTLEN	Hashed password length in chars	32	255
MAXIMPL_MAX_HISTSIZE	Maximum number of passwords kept	50	50
MAXIMPL_SALT	Maximum length of salt	4	32
MAX_PASS	PASS_MAX (defined in limits.h)	32 (PASS_MAX)	PASS_MAX
MAXIMPL_MAX_MINALPHA	Alphanumeric characters	MAXIMPL_PW_PASSLEN	MAXIMPL_PW_PASSLEN
MAXIMPL_MAX_MINOTHER	Non-alphabetic characters	MAXIMPL_PW_PASSLEN	MAXIMPL_PW_PASSLEN
MAXIMPL_MAX_MINDIFF	Different characters in the new password	MAXIMPL_PW_PASSLEN	MAXIMPL_PW_PASSLEN
MAXIMPL_MAX_MAXREP	Repeated characters	MAXIMPL_PW_PASSLEN	MAXIMPL_PW_PASSLEN
MAXIMPL_MAX_MINLEN	Minimum length of a password	MAXIMPL_PW_PASSLEN	MAXIMPL_PW_PASSLEN

Table 8-14 on page 361 provides the maximum size in the current configuration of the system at runtime. These are not suitable for use in array declarations, for example.

Table 8-14 Maximum size in the current configuration of the system

Macro Name	Previous AIX	C2 Extension	New Values (AIX V6.1)
PW_PASSLEN	8	8	__get_pwd_len_max()
PW_CRYPTLEN	13	13	13
MAX_HISTSIZE	50	50	50
MAX_SALT	2	2	__get_salt_len()
MAX_MINALPHA	PW_PASSLEN	PW_PASSLEN	PW_PASSLEN
MAX_MINOTHER	PW_PASSLEN	PW_PASSLEN	PW_PASSLEN
MAX_MINDIFF	PW_PASSLEN	PW_PASSLEN	PW_PASSLEN
MAX_MAXREP	PW_PASSLEN	PW_PASSLEN	PW_PASSLEN
MAX_MINLEN	PW_PASSLEN	PW_PASSLEN	PW_PASSLEN

Changes to password policy attributes

Table 8-15 on page 362 provides the password policy attributes that are related to the maximum length of a clear-text password, which are defined in /etc/security/user for users.

The previous value ranges of these password policy attributes are limited by the previous PASS_MAX (8) value. The new value ranges of these password policy attributes should be limited by the macro PW_PASSLEN (this value is defined by the system password algorithm).

The password restriction routines need to be change to replace the PASS_MAX (8) value with the new PW_PASSLEN value.

The comment (header) section of the /etc/security/user has range values for these password policy attributes, and they are modified to reflect the range changes.

Table 8-15 Password policy attributes

Password policy attributes	Meaning	Previous value	New value (AIX V6.1)
maxrepeats	Defines the maximum number of times a given character can appear in a password.	0 -- 8. Default is 8.	0 -- PW_PASSLEN. Default is PW_PASSLE.
minalpha	Defines the minimum number of alphabetic characters in a password.	0 -- 8. Default is 8.	0 -- PW_PASSLEN. Default is 0.
minlen	Defines the minimum length of a password. The minimum length of a password is determined by minlen or 'minalpha + minother', whichever is greater. 'minalpha + minother' should never be greater than 8. If 'minalpha + minother' is greater than 8, then minother is reduced to '8 - minalpha'.	0 -- 8. Default is 8.	0 -- PW_PASSLEN. Default is 0. The minimum length of a password is determined by minlen or 'minalpha + minother', whichever is greater. 'minalpha + minother' should never be greater than PW_PASSLEN. If 'minalpha + minother' is greater than PW_PASSLEN, then minother is reduced to 'PW_PASSLEN - minalpha'.
minother	Defines the minimum number of non-alphabetic characters in a password.	0 -- 8. Default is 8.	0 -- PW_PASSLEN. Default is 0.
mindiff	Defines the minimum number of characters in the new password that were not in the old password.	0 -- 8. Default is 8.	0 -- PW_PASSLEN. Default is 0.

9

Installation, backup, and recovery

The following AIX Version 6.1 topics are covered in this chapter:

9.1 AIX graphical installer

AIX V6.1 introduces a new AIX base OS graphical installer. The graphical installer provides new AIX administrators with an easy and fast way to install the base operating system. If you boot from the AIX installation DVD, you can now choose between the standard text based install menus and the new graphical installer.

Your system or LPAR must meet these hardware prerequisites:

- DVD drive
- 256 MB RAM
- Graphical adapter
- Keyboard and mouse
- Local SCSI or IDE disk

The graphical installer is available only on the AIX installation DVD. You can use it to install AIX on new systems and it will provide you a fast way to use your new hardware. If the installer detects existing data (defined volume groups) on the disks, the standard text based install menus will be displayed.

The graphical installer will take you through the following steps:

- Welcome window and install language selection (Figure 9-1 on page 365)
- Selection of installation type (Figure 9-2 on page 366)
- Summary and AIX language selection (Figure 9-3 on page 367)

If you need to specify or change other installation options, you have to use the traditional text based menus.

After choosing the installation options and selecting the **Quick Install** button, the installation progress is displayed in the standard text based format.

> **Note:** At the time of writing, the graphical installer does not support VSCSI and SAS disks.

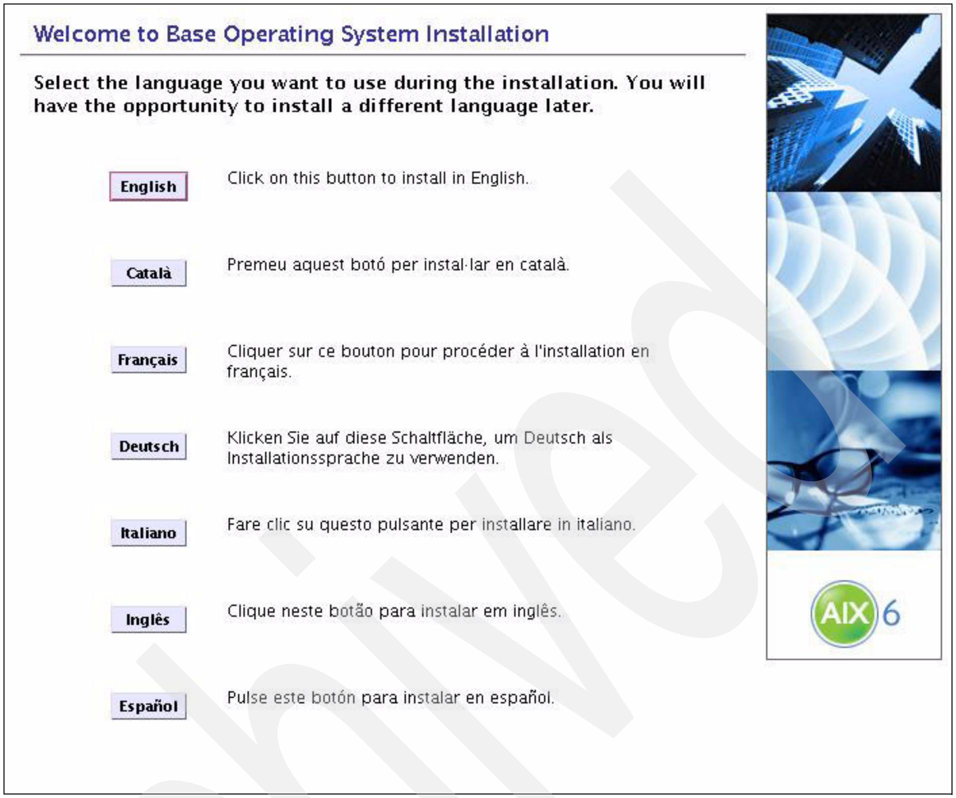


Figure 9-1 AIX graphical installer welcome and installation language window

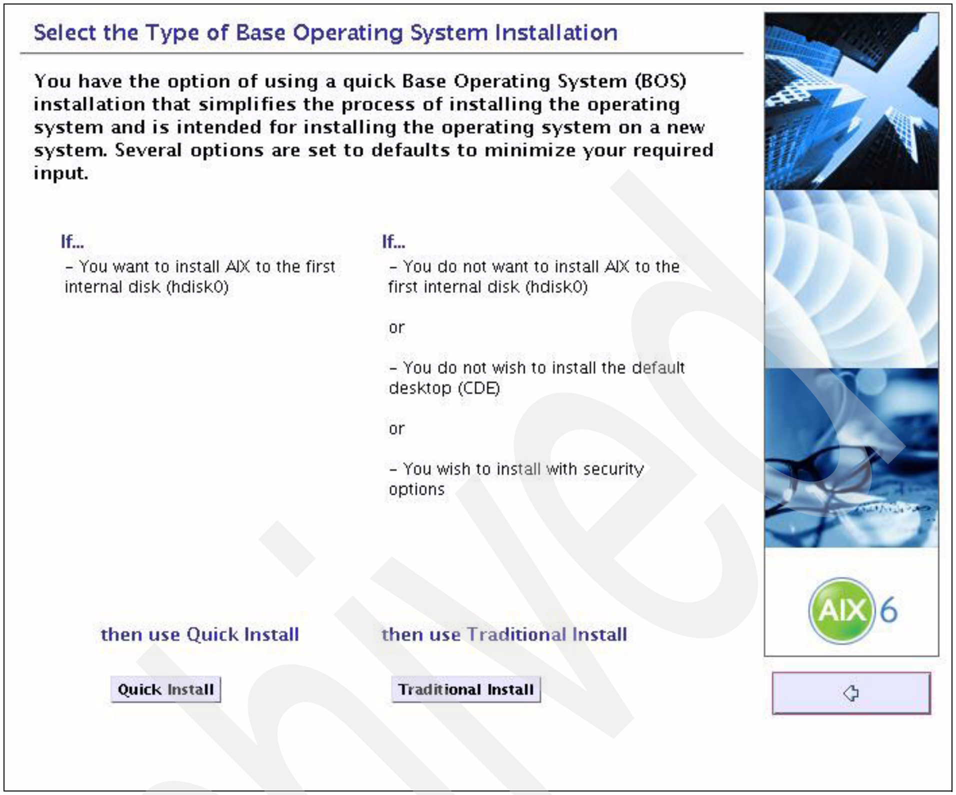


Figure 9-2 AIX graphical installer installation type selection window

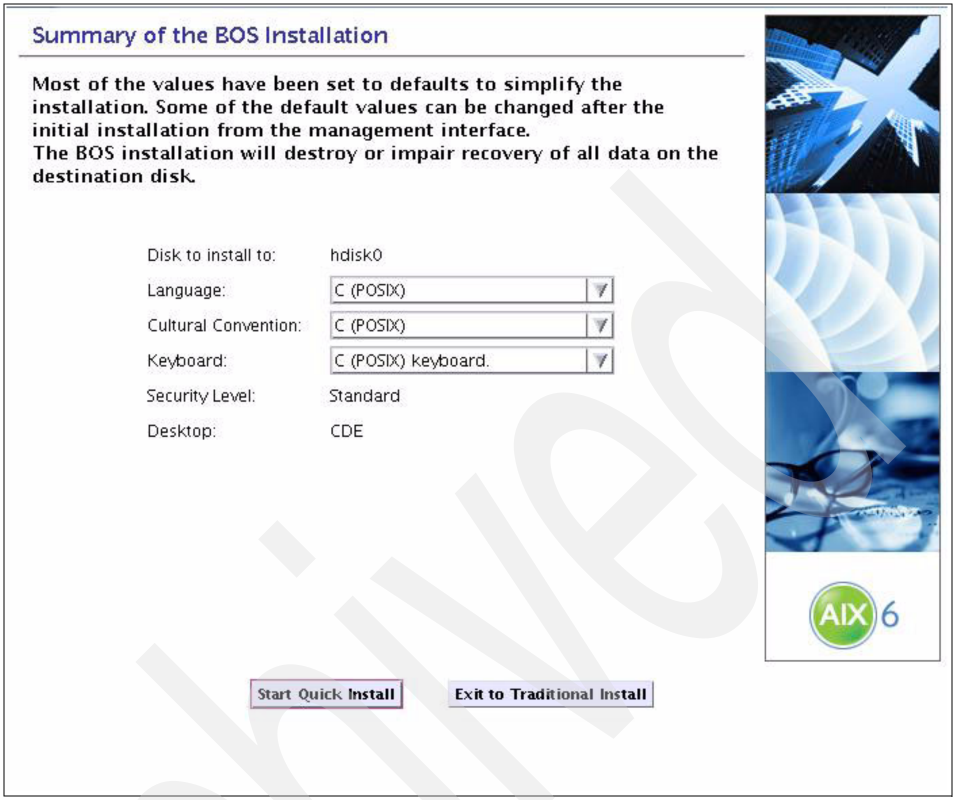


Figure 9-3 AIX graphical installer summary and AIX language selection window

9.2 Network Install Manager NFSv4 support

Network Install Manager (NIM) is used in midsize and large AIX environments to perform base operating system installations over the network. In the past few years, many companies focused on securing their networks. With AIX 5L V5.3, the NIM service handler (nimsh) was introduced to address the following security demands:

- A restricted shell environment that allows only NIM method execution
- Possibility to use OpenSSL encryption with nimsh
- Optional disablement of NIM push operations initiated from the NIM master

All those nimsh features improve the security of remote method execution. The nimsh is introduced as an alternative to the **rsh** and **rcmd** commands, which are considered to be insecure.

AIX V6.1 provides NFSv4 support for NIM environments. With NFSv4 and Kerberos, a securer authentication method is introduced.

The AIX NFSv4 implementation introduces the following general enhancements over NFSv3:

- Built-in security features.
- Pseudo file system concept.
- NFS4 ACL.
- Better performance.
- Locking mechanisms are now part of the protocol itself.

9.2.1 NFSv4 NIM integration

The NIM in AIX V6.1 allows you to specify NFS settings on a NIM resource level. There are two new attributes introduced for that purpose (see Table 9-1).

Table 9-1 New NIM NFS attributes

Attribute	Description
vers	Controls which version of the NFS mounts are allowed. The possible values are 2, 3, and 4. Versions 2 and 3 cannot be enforced separately. Specifying Version 2 or 3 allows access by clients using either NFS protocol versions 2 or 3. Version 4 can be specified independently and must be specified to allow access by clients using Version 4 protocol. The default is 3.
sec	Controls which security methods are allowed. Possible values are sys (UNIX authentication) and krb5 (Kerberos, authentication only). The default is sys.

Note that at the time of writing that the security methods krb5i, krb5p, and dh are not manageable with the **`nim`** command and are not supported.

The attributes can be set on the NIM resource class. The most popular are listed here:

- bosinst_data
- spot
- lpp_source
- installp_bundle
- mksysb
- script

Note: The NFS attributes are not available on any other NIM classes. You cannot distinguish NFS settings on a per machine, network, or groups level. Therefore, you have to make sure that if you restrict, for example, a lpp_source to NFSv4 only, there is no way an AIX 5L V5.2 or older client can use it.

In order to use NFSv4, you must inform the NIM master which NFS domain the local nfsd uses. Use the following command to determine if a NFS domain already exists:

```
# chnfsdom
Current local domain: aix61diff_nfsdom
```

Use the **nim** command to specify which nfs domain name should be used by NIM or specify a new domain to be created:

```
nim -o change -a nfs_domain=aix61diff_nfsdom master
```

Important: This command does not only populate the NIM ODM, it will call the **chnfsdom** <domainname> command afterwards and overwrite the actual domainname. Check for carefully for spelling errors before executing the **nim** command if you want to specify an existent domain.

Example 9-1 shows the changed NFS settings on a lpp_source resource named AIX610_0.

Example 9-1 Change NIM NFS settings on a lpp_source

```
# lsnim -l AIX610_0
AIX610_0:
   class       = resources
   type        = lpp_source
   arch        = power
   Rstate      = ready for use
   prev_state  = unavailable for use
   location    = /export/lpp_source/AIX610_0
   simages     = yes
   alloc_count = 0
   server      = master

# nim -o change -a nfs_sec=sys -a nfs_vers=4 AIX610_0

# lsnim -l AIX610_0
AIX610_0:
   class       = resources
```

```
   type        = lpp_source
   arch        = power
   Rstate      = ready for use
   prev_state  = unavailable for use
   nfs_vers    = 4
   nfs_sec     = sys
   location    = /export/lpp_source/AIX610_0
   simages     = yes
   alloc_count = 0
   server      = master
```

The default is nfs_vers=3 and nfs_sec=sys. Note that if the defaults are active and the attributes have not been changed since the creation of the resource, the **lsnim** command does not display the attributes.

To be able to use NFSv4 during the whole installation, you need to change the bosinst_data and the spot to use NFSv4 and then initiate a bos installation as follows:

```
# nim -o change -a nfs_sec=sys -a nfs_vers=4 bid_ow
# nim -o change -a nfs_sec=sys -a nfs_vers=4 610_0
# nim -o bos_inst -a spot=610_0 -a lpp_source=AIX610_0 \
   -a bosinst_data=bid_ow -a accept_licenses=yes \
   -a force_push=no -a installp_flags=cNgXY lpar02
```

BOS installations cannot be performed with Kerberos security. Changing the sec attribute to krb5 on a spot and then trying to perform **nim** operations like bos_inst will fail and display an error.

9.2.2 NFSv4 security overview

NFSv4 provides information security in the following context:

Identification Establishes the identity of any users, hosts, or services.

Authentication Confirms the identity of a user, host, or service.

Authorization Controls what shared information each user or entity can access.

Table 9-2 on page 371 provides you with an high level overview of differences between the two available security flavors currently supported in AIX V6.1 NIM.

Table 9-2 AUTH_SYS and RPCSEG_GSS Kerberos differences

<table>
<tr><th>Security level</th><th>AUTH_SYS</th><th>RPCSEC_GSS Kerberos</th></tr>
<tr><td rowspan="2">Host Identification</td><td colspan="2">► Domain Name lookup from the IP address of the RPC packets.</td></tr>
<tr><td></td><td>► Machine principal. For example: host/nfs402.itsc.austin.ibm.com.
► Service principal. For example: nfs/nfs402.itsc.austin.ibm.com.</td></tr>
<tr><td>Host Authentication</td><td></td><td>► Service principal. For example: nfs/nfs402.itsc.austin.ibm.com.</td></tr>
<tr><td>Host Authorization</td><td colspan="2">► /etc/exports.
► exportfs command.</td></tr>
<tr><td>User Identification</td><td colspan="2">► Standard UNIX user registry.
► NIS.
► LDAP.</td></tr>
<tr><td>User Authentication</td><td>► Usually logon name and password.
► The NFS server trusts the user and group identities presented by its clients.</td><td>► NFS Service Ticket obtained.
► Established security context for NFS requests.</td></tr>
<tr><td>User Authorization</td><td colspan="2">► Standard UNIX file permissions.
► AIXC ACL.
► NFS V4 ACL.</td></tr>
</table>

9.2.3 RPCSEC_GSS Kerberos sample scripts

NIM includes two sample scripts to set up a basic Kerberos installation. The scripts are provided with the bos.sysmgt.nim.client fileset:

```
# lslpp -f bos.sysmgt.nim.client | grep config_rpcsec
                        /usr/samples/nim/krb5/config_rpcsec_server
                        /usr/samples/nim/krb5/config_rpcsec_client
```

NIM is capable of configuring NFSv4, but due to the variation of Kerberos configurations, you are required to manage KDC configuration and services outside of NIM. Users not familiar with Kerberos can set up a basic Kerberos server on the NIM master.

Prerequisites

In order to use NFSv4 RPCSEC_GSS Kerberos, the following prerequisites must be met on the NIM master:

- The cryptographic library clic.rte must be installed.
- The clickext kernel extension must be loaded.
- AIX V6.1 or later installed.
- Kerberos must be installed (krb5.lic, krb5.client, krb5.server, and modcrypt.base).

On any NIM client that should be able to use a RPCSEC_GSS Kerberos NFS export, the following prerequisites must be fulfilled:

- The cryptographic library clic.rte must be installed.
- The clickext kernel extension must be loaded.
- AIX V6.1 or later must be installed.
- Kerberos must be installed (krb5.lic, krb5.client, and modcrypt.base).

The NIM master and all its clients must be time synchronized. Use the AIX time daemon (timed) or an NTP setup where available.

> **Hint:** Use the following command to check if the clickext module is loaded into the kernel:
>
> ```
> genkex | grep clic
> ```
>
> If it is not loaded, use this command:
>
> ```
> /usr/lib/drivers/crypto/clickext
> ```

Kerberos server on NIM master

Before you run the Kerberos server setup script, look at it. You can customize it to meet your demands. The password at least should be changed. The script will execute the following tasks:

1. Creates a system user (the default is nim).
2. Creates principals for the admin and system user.
3. Creates the nfs host key for the server.
4. Creates realm-to-domain mapping.
5. Creates a tar image of krb5 files for use by KDC slim clients.
6. Cleans up the exports list.
7. Recycles the nfs services.

8. Re-exports nfs file systems and directories.

The script has the following syntax:

```
# /usr/samples/nim/krb5/config_rpcsec_server
```

While running /**usr/samples/nim/krb5/config_rpcsec_server**, you will be prompted for passwords in this order:

1. New system user (standard AIX user registry)
2. Kerberos database master password
3. Principal "admin/admin@REALM1.IBM.COM", as defined in the script Variable PASSWD
4. Principal "admin/admin@REALM1.IBM.COM", as defined in the script Variable PASSWD

Kerberos client on NIM clients

Before you run the Kerberos client setup script, look at it. You can customize it to meet your demands. The script will execute the following tasks:

1. Creates a system user (the default is nim). The user must match an existing user principal on the KDC server.
2. Uses **tftp** to transfer the slim image from the master.
3. Enables the user principal using the **kinit** command.
4. Recycles the NFS services.

The script has the following syntax:

```
# /usr/samples/nim/krb5/config_rpcsec_client
```

While running **/usr/samples/nim/krb5/config_rpcsec_client**, you will be prompted for the password of <nimuser>@REALM1.IBM.COM.

Example installation with a Kerberos NFSv4 export

Check if you have a valid Kerberos TGT on the NIM client. In Example 9-2, we need to request one.

Example 9-2 Obtaining a Kerberos TGT on the NIM client

```
$ /usr/krb5/bin/klist
Unable to get cache name (ticket cache:
/var/krb5/security/creds/krb5cc_10).
        Status 0x96c73ac3 - No credentials cache found.

$ /usr/krb5/bin/kinit

$ /usr/krb5/bin/klist
Ticket cache:  FILE:/var/krb5/security/creds/krb5cc_10
Default principal:  nim@REALM1.IBM.COM

Valid starting     Expires            Service principal
09/28/07 16:10:26  09/29/07 16:09:53
krbtgt/REALM1.IBM.COM@REALM1.IBM.COM
```

On the NIM master, we have an existing lpp_source and installp_bundle for OpenSSL. Example 9-3 shows the following:

- Change the resource attributes to vers=4 and sec=krb5.
- Allocate the NIM resources to the client lpar02.
- Perform the push installation.

Example 9-3 Performing the NIM install steps

```
# nim -o change -a nfs_sec=krb5 -a nfs_vers=4 OPENSSL
# nim -o change -a nfs_sec=krb5 -a nfs_vers=4 openssl_bnd

# nim -o allocate -a lpp_source=OPENSSL lpar02
# nim -o allocate -a installp_bundle=openssl_bnd lpar02
# nim -o cust lpar02
```

During the installation, the /etc/exports file on the NIM master is as follows:

```
# cat /etc/exports
/export/lpp_source/openssl -vers=4,sec=krb5,ro
/export/installp_bundle -vers=4,sec=krb5,ro
```

> **Hint:** The /usr/lpp/bos.sysmgt/nim/README file contains the latest information about the current NIM release.

9.2.4 Considerations

The following are considerations pertaining to NIM NFSv4 support:

- The NFS server calls the rpc.mountd daemon to get the access rights of each client, so the rpc.mountd daemon must be running on the server even if the server only exports file systems with NFSv4.
- NIM supports the pseudo file system concept of NFSv4. For NFSv4 resource exports, the NIM client does a single mount of the servers nfsroot to /tmp/_.nim_mounts._/<host name>.<security flavor>. All data will be accessed under this path.
- You cannot change the nfsroot directory on the NIM NFS server. The NFS default of / (root) must be used for accessing NIM resources.
- The NFSv4 protocol allows no file to file mounts. The NIM server therefore mounts single files (such as scripts and installp_bundles) differently with NFSv4 than with NFSv3. When using NFSv4, the files are accessed through the pseudo file system mount.

10

National language support

AIX Version 6.1 continues to extend the number of nations and regions supported under its national language support. In this chapter, details on the following locales (provided alphabetically) and facilities are provided:

Note: The information included in this section under the discussion of locale support is provided to assist people who are not familiar with the regions better understand the changes made. The accuracy of this information was verified at the time of writing from published information located from education and government Web sites. The colors used for the flags, and their aspect ratios, were carefully chosen to the best of our ability and may appear different depending on the method used to view this publication.

10.1 Azerbaijani locale support

Figure 10-1 shows the flag of the Republic of Azerbaijani.

Figure 10-1 The flag of the Republic of Azerbaijan

There are approximately 30 million native Azerbaijani speakers in the Republic of Azerbaijan, Iran, and other countries in Central Asia. Azerbaijani, also called Azeri Turkish or Azerbaijani Turkish, is the official language of the Republic of Azerbaijan, which is located in southwestern Asia, bordering the Caspian Sea and bounded by the countries of Armenia, Georgia, Iran, Russia, and Turkey. This Turkic language has historical roots in the Turkish, Persian and Arabic languages. The official Azerbaijani language uses Latin alphabets, but it also uses Arabic or Cyrillic scripts in some areas. Azerbaijani-Latin is written from left to right and top to bottom in the same fashion as English.

AIX V6.1 provides full Universal-Coded Character Set (UCS) enablement for the Azerbaijani-Latin language of the Republic of Azerbaijan through a dedicated UCS Transformation Format UTF-8 locale. The UCS language and territory designation for Azerbaijani is AZ_AZ.

The Azerbaijani-Latin script consists of 33 pairs of Latin letters, as shown in Figure 10-2.

```
A Ə B C Ç D E F G Ğ H X I İ J K Q L M N O Ö P R S Ş T U y Ü V Y Z
a ə b c ç d e f g ğ h x ı i j k q l m n o ö p r s ş t u y ü v y z
```

Figure 10-2 Azerbaijani letters

10.1.1 Packaging and installation

The Azerbaijani locale definitions and the underlying support are delivered through the following, separately installable filesets that are grouped and distributed in two entities:

- The bos.loc.utf.AZ_AZ fileset
- The X11.loc.AZ_AZ package, which is comprised of the filesets:
  - X11.loc.AZ_AZ.base.lib
  - X11.loc.AZ_AZ.base.rte
  - X11.loc.AZ_AZ.Dt.rte

The scope of the files in the bos.loc.utf.AZ_AZ fileset is limited to provide the locale support for the AIX base operating system, while the X11.loc.AZ_AZ package will add the locale support for the X11 environment. X11.loc.AZ_AZ.Dt.rte specifically addresses the requirement of the Common Desktop Environment. Several filesets will be automatically installed if the installation process cannot find them on the system:

- bos.loc.com.utf (co-requisite to bos.loc.utf.AZ_AZ)
- X11.fnt.ucs.ttf (co-requisite to X11.loc.AZ_AZ.base.rte)

To verify the complete list of dependencies, you can look at the messages displayed during the installation of the Azerbaijani locale or you can examine the output of the relevant **`lslpp -p`** command after you installed the filesets on your system.

As shown in Figure 10-3 on page 380, during the setup of an AIX installation system, administrators can set the primary language environment to use the predefined triple setup of Azerbaijani-Latin as cultural convention, English (United States) as language, and Azerbaijani-Latin as keyboard.

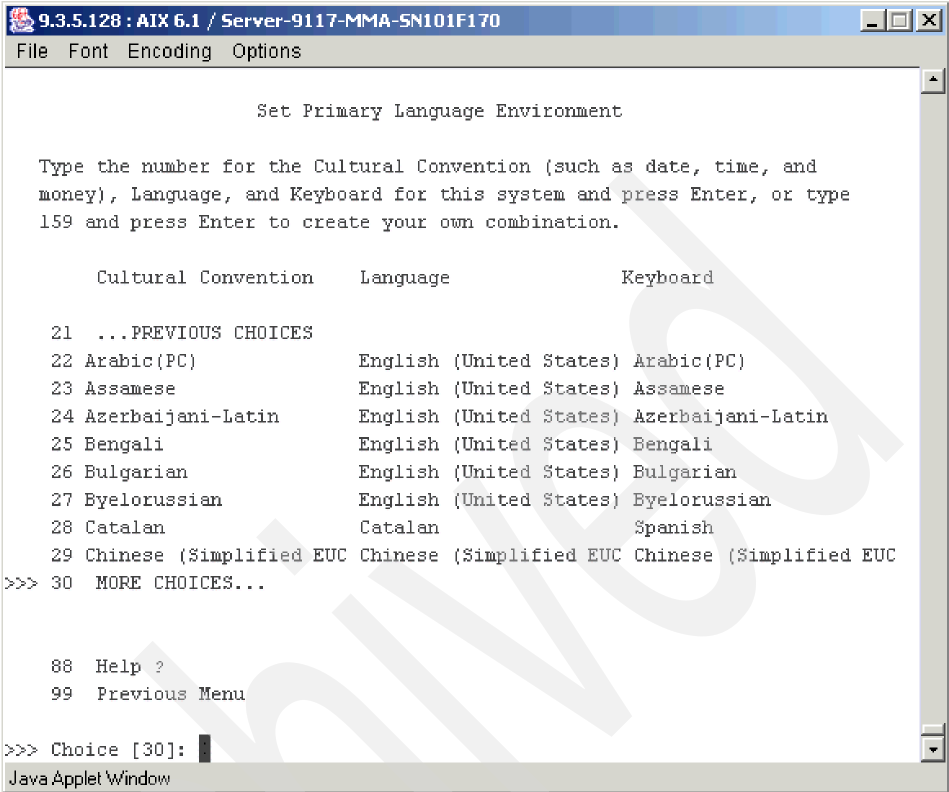


Figure 10-3 Set Primary Language Environment installation menu

After the completion of the base operating system installation, you can use the **lslpp** command to determine which base system locales have been installed as the consequence of the predefined Azerbaijani-Latin primary language environment settings being chosen:

```
# lslpp -l bos.loc*
  Fileset                      Level  State      Description
  ----------------------------------------------------------------------------
Path: /usr/lib/objrepos
  bos.loc.com.utf            6.1.0.0  COMMITTED  Common Locale Support - UTF-8
  bos.loc.utf.EN_US          6.1.0.0  COMMITTED  Base System Locale UTF Code
                                                 Set - U. S. English
  bos.loc.utf.AZ_AZ          6.1.0.0  COMMITTED  Base System Locale UTF Code
                                                 Set - Azerbaijani-Latin
```

Note that in addition to the UTF-8 Azerbaijani-Latin locale, the UTF-8 US English locale has been installed too.

Depending on the graphics related characteristics of a given system, you will also encounter additional filesets in support for Common Desktop Environment and AIXwindows (X11) locales:

```
# lslpp -l X11.loc.*
  Fileset                      Level  State      Description
  ----------------------------------------------------------------------------
Path: /usr/lib/objrepos
  X11.loc.AZ_AZ.Dt.rte       6.1.0.0  COMMITTED  CDE Locale Configuration -
                                                 Azerbaijani-Latin
  X11.loc.AZ_AZ.base.lib     6.1.0.0  COMMITTED  AIXwindows Client Locale
                                                 Config - Azerbaijani-Latin
  X11.loc.AZ_AZ.base.rte     6.1.0.0  COMMITTED  AIXwindows Locale
                                                 Configuration -
                                                 Azerbaijani-Latin
... omitted lines ...

Path: /etc/objrepos
  X11.loc.AZ_AZ.Dt.rte       6.1.0.0  COMMITTED  CDE Locale Configuration -
                                                 Azerbaijani-Latin
```

In case you like to add Azerbaijani-Latin national language support to an existing AIX installation, you can use the SMIT **mlang** locale fast path to access the Change/Show Primary Language Environment or the Add Additional Language Environments SMIT menus.

10.1.2 Locale definitions, keyboard definition, and input methods

A locale is made up of the language, territory, and code set combination used to identify a set of language conventions. The language conventions are grouped in six categories to include information about collation, case conversion, character classification, the language of message catalogs, date-and-time representation, the monetary symbol, and numeric representation. National language support uses the environment variables LC_COLLATE, LC_CTYPE, LC_MESSAGES, LC_MONETARY, LC_NUMERIC, and LC_TIME to define the current values for their respective categories and to influence the selection of locales.

As mentioned previously, the language and territory designation for Azerbaijani national language support is AZ_AZ, and after you configured your environment to use Azerbaijani national language support, you will get the following output by running the **locale** command:

```
# locale
LANG=AZ_AZ
LC_COLLATE="AZ_AZ"
LC_CTYPE="AZ_AZ"
LC_MONETARY="AZ_AZ"
LC_NUMERIC="AZ_AZ"
LC_TIME="AZ_AZ"
LC_MESSAGES="AZ_AZ"
LC_ALL=
```

For example, you can now use the **date** command to verify that the AIX system is actually using the cultural conventions of Azerbaijan for date-and-time representation:

```
# date
2007 Sentyabr 27 16:21:13 CDT
```

The output translates to the following in the US English locale:

```
# date
Thu Sep 27 16:21:13 CDT 2007
```

No AIX message translations are available for Azerbaijani at the time of writing. However, the directory /usr/lib/nls/msg/AZ_AZ will be created during the installation of the Azerbaijani language environment so that applications that desire Azerbaijani translation may provide it. The AIX operating system will use the NLSPATH environment variable to locate any message catalog to be referenced:

```
# echo $NLSPATH
/usr/lib/nls/msg/%L/%N:/usr/lib/nls/msg/%L/%N.cat
```

The Azerbaijani keyboard layout in AIX V6.1 is based on the IBM registered keyboard number 490 (KBD490). In support of the Azerbaijani-Latin locale, KBD490 exhibits a two-layer keyboard layout: group 1 (US English layer) and group 2 (Azerbaijani-Latin layer). The Alt+Left Shift key combination defines the modifier to switch to the group 2 (Azerbaijani-Latin) layer and the Alt+Right Shift key combination will address the group 1 (English 101 key) layer.

AIX supports two different types of keyboards: low function terminal (LFT) and X server keyboards.

The LFT environment is not capable of handling multi-byte code sets such as UTF-8 or complex text (layout oriented) languages. Therefore, the LFT key map for AZ_AZ.lftkeymap in the /usr/lib/nls/loc directory is implemented as a symbolic link to C.lftkeymap:

```
# cd /usr/lib/nls/loc
# ls -l AZ_AZ.lftkeymap | cut -c59-
AZ_AZ.lftkeymap -> /usr/lib/nls/loc/C.lftkeymap
```

Keyboard mapping within the AIX X11 environment is based on the locally attached keyboard. When you start a local X session (through the **xinit** command, the X Display Manager, or the Common Desktop Environment), startup scripts will call the **/usr/bin/X11/xmodmap** command and load the keyboard map for the keyboard language determined by the **/usr/bin/X11/querykbd** command. The **xmodmap** command defines the mapping of the Shift, Lock, and Alt-Graphic (AltGr) keys. The related **xmodmap** command expressions for the Azerbaijani keyboard are defined in the /usr/lpp/X11/defaults/xmodmap/AZ_AZ/keyboard file.

Note: The **xmodmap** command is not called if the display is remote. Keyboard mapping is performed on the local display. Consult your local configuration guides for assistance configuring remote keyboard mapping.

The key events are mapped to a string in the input method mapping imkeymap files.

Single Latin Layer keyboards will be mapped as seen in the **xmodmap** mapping file. For keyboards with additional language groups, the key events for the Right Alt + Right Shift key combination loads the national keyboard layer. So while the key mapped by **xmodmap** is letter a (XK_a), the strings returned by the input method will vary based on the modifier.

For example, the local Azerbaijan (Azeri) keyboard has the following mapping:

```
keycode 36 = bracketright braceright
```

KEYSYM: XK_bracketright is passed to the input method. This key event is mapped as follows, in the Azerbaijan locale input method:

```
BASE:     ']'
SHIFT:   XK_braceright
CAPSLOCK:  XK_bracketright
SHIFT_CAPSLOCK  XK_braceright
```

If the user is in the national layer (using Right Alt + Right Shift), the following strings are mapped to this keysym:

```
Azeri Base  XK_gbreve
Azeri Shift  XK_Gbreve
Azeri CapsLock  XK_Gbreve
Azeri Shift CapsLock XK_gbreve
Control   '\x1d'
Alt   U  (undefined)
```

AIX V6.1 provides the standard UNIVERSAL input method as well as the traditional single-byte input method through the AZ_AZ.im and the AZ_AZ.UTF-8.im files, respectively. Both input methods are related by the use of the same UNIVERSAL input method configuration file UNIVERSAL.imcfg. The input method files, the related configuration files, and the input method keymap definition files are located in the /usr/lib/nls/loc directory:

```
# ch /usr/lib/nls/loc
# ls -l AZ_AZ*im* | cut -c59-
AZ_AZ.im -> /usr/lib/nls/loc/UNIVERSAL.im
AZ_AZ.im__64 -> /usr/lib/nls/loc/UNIVERSAL.im__64
AZ_AZ.imcfg -> /usr/lib/nls/loc/UNIVERSAL.imcfg
AZ_AZ.imcompose
AZ_AZ.imkeymap -> /usr/lib/nls/loc/AZ_AZ.UTF-8.imkeymap
AZ_AZ.UTF-8.im -> /usr/lib/nls/loc/sbcs.im
AZ_AZ.UTF-8.im__64 -> /usr/lib/nls/loc/sbcs.im__64
AZ_AZ.UTF-8.imcfg -> /usr/lib/nls/loc/UNIVERSAL.imcfg
AZ_AZ.UTF-8.imcompose -> /usr/lib/nls/loc/AZ_AZ.imcompose
AZ_AZ.UTF-8.imkeymap
```

10.2 Euro symbol support

Figure 10-4 shows the flag of the European Union.

Figure 10-4 The flag of the European Union

By the end of 2002, the national currencies were effectively withdrawn and replaced by the euro currency in 12 European Union countries: Austria, Belgium, Finland, France, Germany, Greece, Ireland, Italy, Luxembourg, the Netherlands, Portugal, Slovenia, and Spain.

In May 2004, ten new countries joined the European Union: Cyprus, Czech Republic, Estonia, Hungary, Latvia, Lithuania, Malta, Poland, Slovakia, and Slovenia. In January 2007, an additional two countries joined the European Union: Bulgaria and Romania.

Slovenia was the first of the new member states to adopt the euro currency. In this country, the new currency entered circulation on January 1, 2007. The remaining countries will eventually introduce the euro currency as soon as they meet the Maastricht convergency criteria, among other necessary conditions.

For more information about the euro currency, consult the official Web page of the European Monetary Union, found at:

http://ec.europa.eu/euro

AIX V6.1 provides two new euro enabled locales and euro enablement to twelve existing locales mainly in support for the ascension of the new member states to the European Union in 2004 and 2007, and their recent or impending adoption of euro as their national currency.

The two newly added euro enabled locales support the Maltese/Malta and Welsh/United Kingdom language and territory combinations. The new locales are covered in detail in 10.3, “Maltese locale support” on page 388 and 10.5, “Welsh locale support” on page 400.

Table 10-1 gives an overview of the new and enhanced AIX V6.1 locales in support of the euro currency.

Table 10-1 New and enhanced AIX V6.1 locales in support of the euro currency

Language / Territory	UTF-8 locale	ISO locale	IBM-92x locale	Currency
Czech / Czech Republic	CS_CZ.UTF-8	cs_CZ.ISO8859-2	N/A	preeuro
Danish / Denmark	DA_DK.UTF-8	da_DK.ISO8859-15[a]	N/A	preeuro
Estonian / Estonia	ET_EE.UTF-8	et_EE.ISO8859-4	Et_EE.IBM-922	preeuro
Hungarian / Hungary	HU_HU.UTF-8	hu_HU.ISO8859-2	N/A	preeuro
Latvian / Latvia	LV_LV.UTF-8	lv_LV.ISO8859-4	Lv_LV.IBM-921	preeuro
Lithuanian / Lithuania	LT_LT.UTF-8	lt_LT.iso8859-4	Lt_LT.IBM-921	preeuro
Maltese / Malta	MT_MT.UTF-8	N/A	N/A	euro
Polish / Poland	PL_PL.UTF-8	pl_PL.ISO8859-2	N/A	preeuro
Slovak / Slovakia	SK_SK.UTF-8	sk_SK.ISO8859-2	N/A	preeuro
Slovenian / Slovenia	SL_SI.UTF-8	sl_SI.ISO8859-2	N/A	euro
Swedish / Sweden	SV_SE.UTF-8	sv_SE.ISO8859-15[a]	N/A	preeuro
Welsh / United Kingdom	CY_GB.UTF-8	N/A	N/A	preeuro
Bulgarian / Bulgaria	BG_BG.UTF-8	bg_BG.ISO8859-5	N/A	preeuro
Romanian / Romania	RO_RO.UTF-8	ro_RO.ISO8859-2	N/A	preeuro

a. Graphical euro symbol supported

All euro enabled locales are designed to effectively support a dual currency environment as induced by isochronous euro and traditional currency requirements. To that extent, AIX provides the @euro and the @preeuro modifiers to the LC_MONETARY category. The modifiers are appended as suffix to the language and territory designation of a given locale and, as indicated by the keywords, the @euro modifier activates the euro currency symbol and the related formatting rules, while the @preeuro modifier does the same for the traditional currency. As shown by the last column of Table 10-1, the Maltese and the Slovenian locale are the only locales that use the @euro modifier by default. Every other locale listed defaults to the traditional currency symbol and formatting rules. Note that for the given locales, all the IBM locales (IBM-921 and IBM-922) and all the ISO8859 code sets, excluding the ISO8859-15, are not able to support the graphical euro symbol.

IBM is following the recommendation of the European Commission regarding the placement of the euro symbol on keyboards. The Commission recommends placing the euro symbol at the position AltGr+e on all European keyboards, except on those keyboard layouts where the key combination AltGr+e is already assigned to produce a different character. In those cases, a combination of AltGr+4 or AltGr+5 will be assigned as a first choice alternative. The existing logical keyboard layout registrations for the countries listed in Table 10-2 have been updated to facilitate entering the euro sign. For the new Maltese and Welsh language environments, the keyboard registrations have been added. You also can see some cases for euro symbol placement, such as Shift+3 and AltGr+u key combinations, where the first choice alternatives ultimately were not available.

Table 10-2 New and modified AIX keyboards for euro symbol support

Language / Territory	AIX keyboard name	Keyboard ID	Euro placement
Czech / Czech Republic	CS_CZ	243	AltGr+e
Danish / Denmark	DA_DK	159	AltGr+5
Estonian / Estonia	ET_EE	454	AltGr+e
Hungarian / Hungary	HU_HU	208	AltGr+u
Latvian / Latvia	LV_LV	455	AltGr+4
Lithuanian / Lithuania	LT_LT	456	AltGr+e
Maltese / Malta	MT_MT	491	Shift+3
Polish / Poland	PL_PL	214	AltGr+u
Slovak / Slovakia	SK_SK	245	AltGr+e
Slovenian / Slovenia	SL_SI	234	AltGr+e
Swedish / Sweden	SV_SE	153	AltGr+e
Welsh / United Kingdom	CY_GB	166 and 166W[a]	AltGr+4
Bulgarian / Bulgaria	BG_BG	442	AltGr+e
Romanian / Romania	RO_RO	446	AltGr+e

a. Keyboard ID 166W provides a supplementary layout to be used with keyboard ID 166 to support Welsh and Cornish.

10.3 Maltese locale support

Figure 10-5 shows the flag of the Republic of Malta.

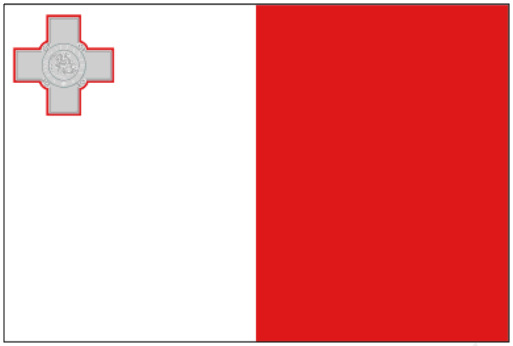


Figure 10-5 The flag of the Republic of Malta

The Republic of Malta is located in the Mediterranean Sea around 60 miles southwest of Sicily, and about 180 miles northwest of the Tunisian coast. The Maltese archipelago primarily consists of three islands: Malta, Gozo, and Camino, and at the time of writing the combined population is estimated to be around 400,000 people. The Republic of Malta joined the European Union in 2004 and plans to adopt the euro currency by January 1, 2008.

As declared by the Constitution of Malta, the national language of Malta is Maltese, but the English language is recognized as an official language too. Maltese is only the Semitic Language written in the Latin alphabet.

AIX V6.1 provides full Universal-Coded Character Set (UCS) enablement for the Maltese language through a dedicated UCS Transformation Format UTF-8 locale. The UCS language and territory designation for Maltese is given by MT_MT.

The Maltese script consists of 30 pairs of Latin letters, as shown in Figure 10-2 on page 378.

```
A B Ċ D E F Ġ G Għ H Ħ I Ie J K L M N O P Q R S T U V W X Ż Z
a b ċ d e f ġ g għ h ħ i ie j k l m n o p q r s t u v w x ż z
```

Figure 10-6 Maltese letters

10.3.1 Packaging and installation

The Maltese locale definitions and the underlying support are delivered through the following, separately installable filesets that are grouped and distributed in two entities:

- bos.loc.utf.MT_MT fileset
- X11.loc.MT_MT package comprised of the filesets
  - X11.loc.MT_MT.base.lib
  - X11.loc.MT_MT.base.rte
  - X11.loc.MT_MT.Dt.rte

The scope of the files in the bos.loc.utf.MT_MT fileset is limited to provide the locale support for the AIX base operating system, while the X11.loc.MT_MT package will add the locale support for the X11 environment. X11.loc.MT_MT.Dt.rte specifically addresses the requirement of the Common Desktop Environment. Several filesets will be automatically installed if the installation process cannot find them on the system:

- bos.loc.com.utf (co-requisite to bos.loc.utf.MT_MT)
- X11.fnt.ucs.ttf (co-requisite to X11.loc.MT_MT.base.rte)

To verify the complete list of dependencies, look at the messages displayed during the installation of the Maltese locale or examine the output of the relevant **`lslpp -p`** command after you install the filesets on your system.

As shown in Figure 10-3 on page 380, during the setup of an AIX installation system, administrators can set the primary language environment to use the predefined triple setup of Maltese as cultural convention, English (United States) as language, and Maltese as keyboard.

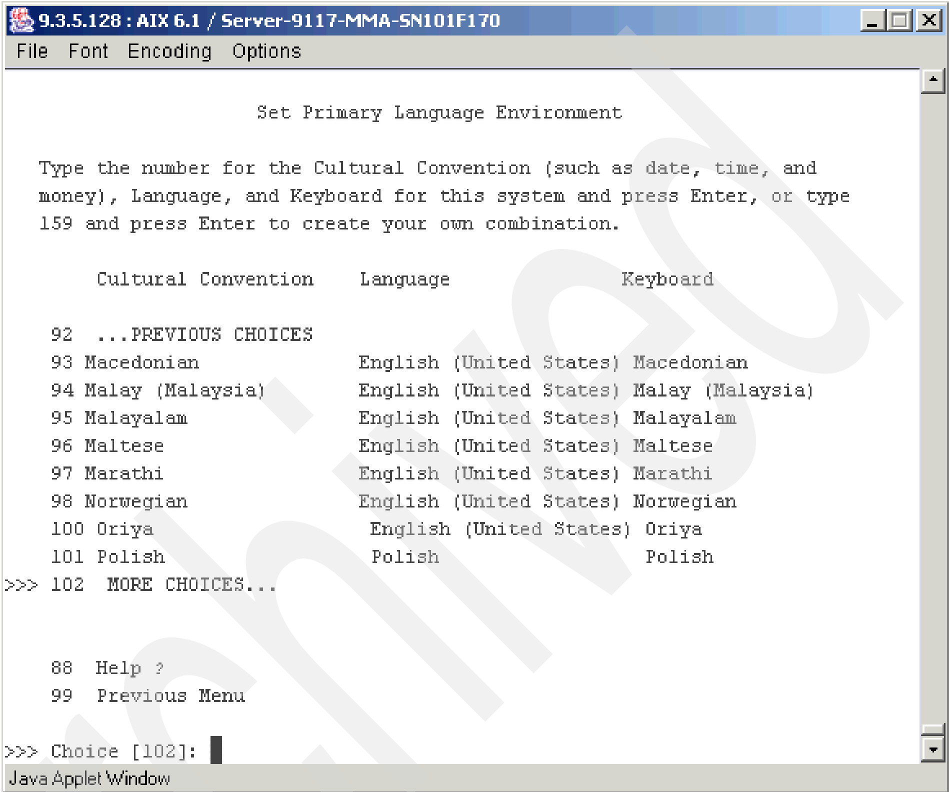


Figure 10-7 Set Primary Language Environment installation menu

After completing the base operating system installation, you can use the **lslpp** command to determine which base system locales have been installed as a consequence of the predefined Maltese primary language environment settings chosen:

```
# lslpp -l bos.loc*
  Fileset                      Level  State      Description
  ----------------------------------------------------------------------------
Path: /usr/lib/objrepos
  bos.loc.com.utf            6.1.0.0  COMMITTED  Common Locale Support - UTF-8
  bos.loc.utf.EN_US          6.1.0.0  COMMITTED  Base System Locale UTF Code
                                                 Set - U. S. English
  bos.loc.utf.MT_MT          6.1.0.0  COMMITTED  Base System Locale UTF Code
                                                 Set - Maltese
```

Note that in addition to the UTF-8 Maltese locale, the UTF-8 US English locale has been installed too.

Depending on the graphics related characteristics of a given system, you will also encounter additional filesets that support the Common Desktop Environment and AIXwindows (X11) locales:

```
# lslpp -l X11.loc.*
  Fileset                      Level  State      Description
  ----------------------------------------------------------------------------
Path: /usr/lib/objrepos
  X11.loc.MT_MT.Dt.rte       6.1.0.0  COMMITTED  CDE Locale Configuration -
                                                 Maltese
  X11.loc.MT_MT.base.lib     6.1.0.0  COMMITTED  AIXwindows Client Locale
                                                 Config - Maltese
  X11.loc.MT_MT.base.rte     6.1.0.0  COMMITTED  AIXwindows Locale
                                                 Configuration - Maltese
... omitted lines ...

Path: /etc/objrepos
  X11.loc.MT_MT.Dt.rte       6.1.0.0  COMMITTED  CDE Locale Configuration -
                                                 Maltese
```

In case you like to add Maltese national language support to an existing AIX installation, you can use the SMIT **mlang** locale fast path to access the Change/Show Primary Language Environment or the Add Additional Language Environments SMIT menus.

10.3.2 Locale definitions, keyboard definition, and input methods

A locale is made up of the language, territory, and code set combination used to identify a set of language conventions. The language conventions are grouped in six categories to include information about collation, case conversion, character classification, the language of message catalogs, date-and-time representation, the monetary symbol, and numeric representation. National language support uses the environment variables LC_COLLATE, LC_CTYPE, LC_MESSAGES, LC_MONETARY, LC_NUMERIC, and LC_TIME to define the current values for their respective categories and to influence the selection of locales.

As mentioned previously, the language and territory designation for Maltese national language support is MT_MT, and after you configured your environment to use Maltese national language support, you will get the following output by running the **locale** command:

```
# locale
LANG=MT_MT
LC_COLLATE="MT_MT"
LC_CTYPE="MT_MT"
LC_MONETARY="MT_MT"
LC_NUMERIC="MT_MT"
LC_TIME="MT_MT"
LC_MESSAGES="MT_MT"
LC_ALL=
```

For example, you can now use the **date** command to verify that the AIX system is actually using the cultural conventions of Malta for date and time representation:

```
# date
26 taâ Settembru 2007 14:59:55 CDT
```

The output translates to the following in the US English locale:

```
# date
Wed Sep 26 14:59:55 CDT 2007
```

No AIX message translations are available for Maltese at the time of writing. However, the directory /usr/lib/nls/msg/MT_MT will be created during the installation of the Maltese language environment so that applications that desire Maltese translation may provide it. The AIX operating system will use the NLSPATH environment variable to locate any message catalog to be referenced:

```
# echo $NLSPATH
/usr/lib/nls/msg/%L/%N:/usr/lib/nls/msg/%L/%N.cat
```

Any UTF-8-based locale installed on an AIX system will provide support for the euro symbol. As Malta is among the countries that have to actively use the euro, the Maltese locale will deliver the input methods and the keyboard maps required to enter the euro symbol through the keyboard. An additional LC_MONETARY locale is available to enable the euro currency formatting. This locale is identified by the suffix @euro. As such, the alternate euro currency format is invoked when LC_MONETARY=MT_MT@euro is specified through the locale environment variables, or with the setlocale subroutine.

The **locale** command output below shows the required environment variable settings in support of euro currency formatting:

```
# locale
LANG=MT_MT
LC_COLLATE="MT_MT"
LC_CTYPE="MT_MT"
LC_MONETARY=MT_MT@euro
LC_NUMERIC="MT_MT"
LC_TIME="MT_MT"
LC_MESSAGES="MT_MT"
LC_ALL=
```

To allow dual currency support, the Maltese locale also provides the MT_MT@preeuro locale for the LC_MONETARY category. The MT_MT@preeuro locale is linked to the default locale for traditional national currency formatting requirements of users and applications.

The Maltese keyboard layout in AIX V6.1 is based on the IBM registered keyboard number 491 (KBD491). In support of the Maltese locale, KBD491 exhibits a two-layer keyboard layout: group 1 (state 0) and group 2 (state 32). The Alt+Left Shift key combination defines the modifier to switch to the group 2 layer and the Alt+Right Shift key combination will address the group 1 layer.

AIX supports two different types of keyboards: low function terminal (LFT) and X server keyboards.

The LFT environment is not capable of handling multi-byte code sets such as UTF-8 or complex text (layout oriented) languages. Therefore, the LFT key map for MT_MT.lftkeymap in the /usr/lib/nls/loc directory is implemented as a symbolic link to C.lftkeymap:

```
# cd /usr/lib/nls/loc
# ls -l MT_MT.lftkeymap | cut -c59-
MT_MT.lftkeymap -> /usr/lib/nls/loc/C.lftkeymap
```

Keyboard mapping within the AIX X11 environment is based on the locally attached keyboard. When you start a local X Window System session (through the **xinit** command, the X Display Manager, or the Common Desktop Environment), startup scripts will call the **/usr/bin/X11/xmodmap** command and load the keyboard map for the keyboard language determined by the **/usr/bin/X11/querykbd** command. The **xmodmap** command defines the mapping of the Shift, Lock, and Alt-Graphic (AltGr) keys. The related **xmodmap** command expressions for the Maltese keyboard are defined in the /usr/lpp/X11/defaults/xmodmap/MT_MT/keyboard file.

Note: The **xmodmap** command is not called if the display is remote. Keyboard mapping is performed on the local display. Consult your local configuration guides for assistance configuring the remote keyboard mapping.

AIX V6.1 provides the standard UNIVERSAL input method as well as the traditional single-byte input method through the MT_MT.im and the MT_MT.UTF-8.im files, respectively. Both input methods are related by the use of the same UNIVERSAL input method configuration file UNIVERSAL.imcfg. The input method files, the related configuration files, and the input method keymap definition files are located in the /usr/lib/nls/loc directory:

```
# ls -l MT_MT*im* | cut -c59-
MT_MT.UTF-8.im -> /usr/lib/nls/loc/sbcs.im
MT_MT.UTF-8.im__64 -> /usr/lib/nls/loc/sbcs.im__64
MT_MT.UTF-8.imcfg -> /usr/lib/nls/loc/UNIVERSAL.imcfg
MT_MT.UTF-8.imkeymap
MT_MT.im -> /usr/lib/nls/loc/UNIVERSAL.im
MT_MT.im__64 -> /usr/lib/nls/loc/UNIVERSAL.im__64
MT_MT.imcfg -> /usr/lib/nls/loc/UNIVERSAL.imcfg
MT_MT.imkeymap -> /usr/lib/nls/loc/MT_MT.UTF-8.imkeymap
```

10.4 Urdu India and Urdu Pakistan locale support

Figure 10-8 shows the Republic of India and the Islamic Republic of Pakistan flags.

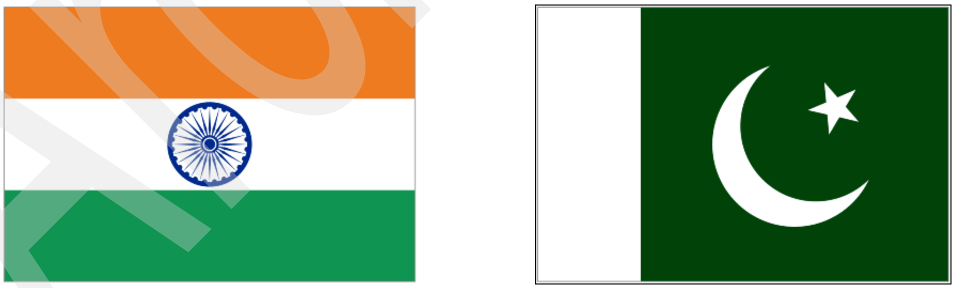


Figure 10-8 Republic of India and Islamic Republic of Pakistan flags

Urdu is spoken in Pakistan (as a national language), northern India, Afghanistan, and other countries in the eastern Asia area. Nearly 100,000,000 people in 20 countries are using Urdu as a first or second language. Urdu uses Urdu script (Persian-Arabic script) to write from right to left and top to bottom

AIX V6.1 provides full Universal-Coded Character Set (UCS) enablement for the Urdu language through dedicated UCS Transformation Format UTF-8 locales for Urdu India and Urdu Pakistan. The UCS language and territory designation for Urdu national language support is given by UR_IN and UR_PK for Urdu India and Urdu Pakistan, respectively.

For the remainder of this section, the substitution characters XX will represent both territory designation: IN (India) and PK (Pakistan).

Figure 10-9 shows some of the Urdu characters.

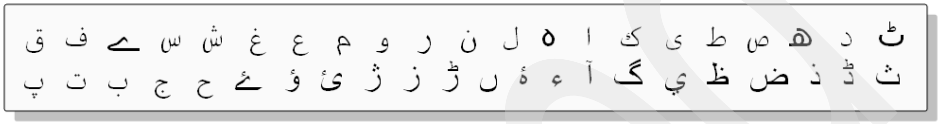


Figure 10-9 Some examples of Urdu characters

10.4.1 Packaging and installation

The Urdu India and Urdu Pakistan locale definitions and their underlying support are delivered through separately installable filesets that are grouped and distributed into two entities for each territory.

The Urdu India national language support filesets are:

- bos.loc.utf.UR_IN fileset
- X11.loc.UR_IN package comprised of the filesets
  - X11.loc.UR_IN.base.lib
  - X11.loc.UR_IN.base.rte
  - X11.loc.UR_IN.Dt.rte

The Urdu Pakistan national language support filesets are:

- bos.loc.utf.UR_PK fileset
- X11.loc.UR_PK package comprised of the filesets
  - X11.loc.UR_PK.base.lib
  - X11.loc.UR_PK.base.rte
  - X11.loc.UR_PK.Dt.rte

The scope of the files in the bos.loc.utf.UR_XX fileset is limited to provide the locale support for the AIX base operating system, while the X11.loc.UR_XX package will add the locale support for the X11 environment. X11.loc.UR_XX.Dt.rte specifically addresses the requirements of the Common Desktop Environment.

Several filesets will be automatically installed if the installation process cannot find them on the system:

- bos.loc.com.utf (co-requisite to bos.loc.utf.UR_XX)
  Common Locale Support - UTF-8
- bos.loc.com.bidi (co-requisite to bos.loc.utf.UR_XX)
  Common Locale Support - Bidirectional Languages
- X11.fnt.ucs.ttf (co-requisite to X11.loc.UR_XX.base.rte)
  AIXwindows Unicode TrueType Fonts
- X11.fnt.ucs.ttf_extb (co-requisite to X11.loc.UR_XX.base.rte)
  AIXwindows Unicode TrueType Fonts - Extension B

To verify the complete list of dependencies, you can look at the messages displayed during the installation of the Urdu locales or you can examine the output of the relevant **lslpp -p** command after you installed the filesets on your system.

System administrators can chose to install an AIX system with a primary language environment setup for Urdu India or Urdu Pakistan. For the Indian territory, the environment is determined by the predefined triple setup of Urdu India as cultural convention, English (United States) as language, and Urdu India as keyboard. For the Pakistani territory, the environment is determined by the predefined triple setup of Urdu Pakistan as cultural convention, English (United States) as language, and Urdu Pakistan as keyboard.

After the completion of the base operating system installation, you can use the **lslpp** command to verify which base system locales have been installed as a consequence of the predefined Urdu primary language environment settings that were chosen. The **lslpp** command listings provided in the following paragraphs are characteristic of a system that has been initially set up to support Urdu Pakistan. By replacing PK with IN in the fileset names, the same listings would apply to the Urdu India environment:

```
# lslpp -l bos.loc*
  Fileset                      Level  State      Description
  ----------------------------------------------------------------------------
Path: /usr/lib/objrepos
  bos.loc.com.bidi           6.1.0.0  COMMITTED  Common Locale Support -
                                                 Bidirectional Languages
  bos.loc.com.utf            6.1.0.0  COMMITTED  Common Locale Support - UTF-8
  bos.loc.utf.EN_US          6.1.0.0  COMMITTED  Base System Locale UTF Code
                                                 Set - U. S. English
  bos.loc.utf.UR_PK          6.1.0.0  COMMITTED  Base System Locale UTF Code
                                                 Set - Urdu (Pakistan)
```

Note that in addition to the UTF-8 Urdu locale, the UTF-8 US English locale has been installed as well.

Depending on the graphics related characteristics of a given system, you will also encounter additional filesets in support of the Common Desktop Environment and AIXwindows (X11) locales:

```
# lslpp -l X11.loc.*
  Fileset                      Level  State      Description
  ----------------------------------------------------------------------------
Path: /usr/lib/objrepos
  X11.loc.UR_PK.Dt.rte       6.1.0.0  COMMITTED  CDE Locale Configuration -
                                                 Maltese
  X11.loc.UR_PK.base.lib     6.1.0.0  COMMITTED  AIXwindows Client Locale
                                                 Config - Maltese
  X11.loc.UR_PK.base.rte     6.1.0.0  COMMITTED  AIXwindows Locale
                                                 Configuration - Maltese
... omitted lines ...

Path: /etc/objrepos
  X11.loc.UR_PK.Dt.rte       6.1.0.0  COMMITTED  CDE Locale Configuration -
                                                 Maltese
```

If you want to add Urdu national language support to an existing AIX installation, use the SMIT **mlang** locale fast path to access the Change/Show Primary Language Environment or the Add Additional Language Environments SMIT menus.

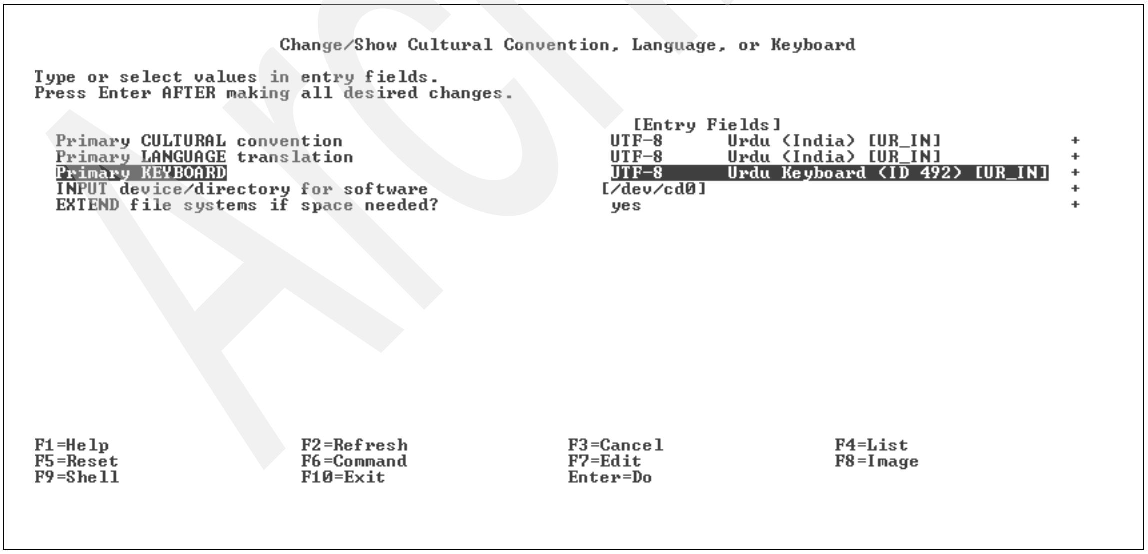


Figure 10-10 SMIT menu to add Urdu national language support for India

AIX V6.1 also provides code set conversion tables for the none-Unicode Urdu codesets IBM-868, IBM-918, and IBM-1006 to support conversion of none-Unicode Urdu characters to and from Unicode. Note that Urdu conversions between none-Unicode and Unicode characters are not a one-to-one mapping.

10.4.2 Locale definitions, keyboard definition, and input methods

A locale is made up of the language, territory, and code set combination used to identify a set of language conventions. The language conventions are grouped in six categories to include information about collation, case conversion, character classification, the language of message catalogs, date-and-time representation, the monetary symbol, and numeric representation. National language support uses the environment variables LC_COLLATE, LC_CTYPE, LC_MESSAGES, LC_MONETARY, LC_NUMERIC, and LC_TIME to define the current values for their respective categories and to influence the selection of locales.

As mentioned previously, the language designation for Urdu locale support is UR and the territory designation is given by IN and PK for India and Pakistan, respectively.

After you configured your environment to use Urdu India national language support, you will get the following output from the **locale** command:

```
# locale
LANG=UR_IN
LC_COLLATE="UR_IN"
LC_CTYPE="UR_IN"
LC_MONETARY="UR_IN"
LC_NUMERIC="UR_IN"
LC_TIME="UR_IN"
LC_MESSAGES="UR_IN"
LC_ALL=
```

After you configured your environment to use Urdu Pakistan national language support, you will get the following output from the **locale** command:

```
# locale
LANG=UR_PK
LC_COLLATE="UR_PK"
LC_CTYPE="UR_PK"
LC_MONETARY="UR_PK"
LC_NUMERIC="UR_PK"
LC_TIME="UR_PK"
LC_MESSAGES="UR_PK"
LC_ALL=
```

No AIX message translations are available for Urdu at the time of writing. However, depending on the territory designation, either the directory /usr/lib/nls/msg/UR_IN (India) or the directory usr/lib/nls/msg/UR_PK (Pakistan) will be created during the installation of the Urdu language environment so that applications that desire Urdu translation may provide it. The AIX operating system will use the NLSPATH environment variable to locate any message catalog to be referenced:

```
# echo $NLSPATH
/usr/lib/nls/msg/%L/%N:/usr/lib/nls/msg/%L/%N.cat
```

The Urdu keyboard layout in AIX V6.1 is based on the same IBM registered keyboard number 492 (KBD492) for both territories, India and Pakistan. In support of the Urdu locale, KBD492 exhibits a two-layer keyboard layout: Latin (state 0) and Urdu (state 32). The Alt+Left Shift key combination defines the modifier to switch to the Latin layer and the Alt+Right Shift key combination will address the Urdu layer. The Latin layer of the keyboard is equivalent to the US English 101 keyboard.

AIX supports two different types of keyboards: low function terminal (LFT) and X server keyboards.

The LFT environment is not capable of handling multi-byte code sets such as UTF-8 nor complex text (layout oriented) languages. Therefore, the LFT key map for UR_XX.lftkeymap in the /usr/lib/nls/loc directory is implemented as a symbolic link to C.lftkeymap. Use the relevant **ls -l** command in the /usr/lib/nls/loc directory to verify the references:

```
# cd /usr/lib/nls/loc
# ls -l UR_IN.lftkeymap | cut -c59-
UR_IN.lftkeymap -> /usr/lib/nls/loc/C.lftkeymap

# ls -l UR_PK.lftkeymap | cut -c59-
UR_PK.lftkeymap -> /usr/lib/nls/loc/C.lftkeymap
```

Keyboard mapping within an AIX X11 environment is based on the locally attached keyboard. When you start a local X session (through the **xinit** command, the X Display Manager, or the Common Desktop Environment), startup scripts will call the **/usr/bin/X11/xmodmap** command and load the keyboard map for the keyboard language determined by the **/usr/bin/X11/querykbd** command. The **xmodmap** command defines the mapping of the Shift, Lock, and Alt-Graphic (AltGr) keys. The related **xmodmap** command expressions for the Urdu keyboard are defined in the /usr/lpp/X11/defaults/xmodmap/UR_XX/keyboard file, where XX represents IN or PK.

Note: The **xmodmap** command is not called if the display is remote. Keyboard mapping is performed on the local display. Consult your local configuration guides for assistance configuring remote keyboard mapping.

AIX V6.1 provides the standard UNIVERSAL input method as well as the traditional single-byte input method through the UR_XX.im and the UR_XX.UTF-8.im files, respectively. Both input methods are related by the use of the same UNIVERSAL input method configuration file UNIVERSAL.imcfg. The input method files, the related configuration files, and the input method keymap definition files are located in the /usr/lib/nls/loc directory:

```
# ls -l UR_PK*im* | cut -c55-
UR_PK.im -> /usr/lib/nls/loc/UNIVERSAL.im
UR_PK.im__64 -> /usr/lib/nls/loc/UNIVERSAL.im__64
UR_PK.imcfg -> /usr/lib/nls/loc/UNIVERSAL.imcfg
UR_PK.imcompose -> /usr/lib/nls/loc/BIM.imcompose
UR_PK.imkeymap -> /usr/lib/nls/loc/UR_PK.UTF-8.imkeymap
UR_PK.UTF-8.im -> /usr/lib/nls/loc/sbcs.im
UR_PK.UTF-8.im__64 -> /usr/lib/nls/loc/sbcs.im__64
UR_PK.UTF-8.imcfg -> /usr/lib/nls/loc/UNIVERSAL.imcfg
UR_PK.UTF-8.imcompose -> /usr/lib/nls/loc/BIM.imcompose
UR_PK.UTF-8.imkeymap
```

By replacing PK with IN in the fileset names, the previous listing would apply to the Urdu India environment.

10.5 Welsh locale support

Figure 10-11 shows the Welsh flag.

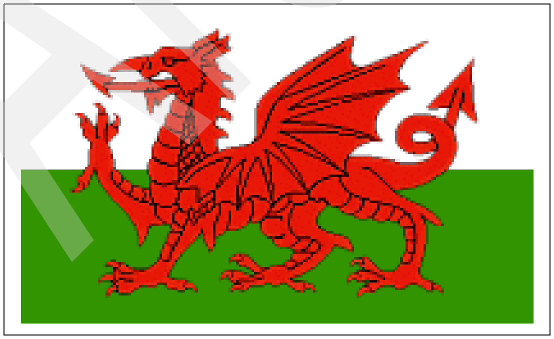


Figure 10-11 The Welsh flag

England, Northern Ireland, Scotland, and Wales are the four constituent countries of the United Kingdom. At the time of writing, the population of Wales is estimated to be around 3 million and the official languages are English and Welsh. In Welsh, the language itself is named Cymraeg and it is a member of the Brythonic branch of the Celtic language. According to the most recent language survey, more than 20% of the population is able to speak Welsh. The percentage of speakers in Wales continues to grow, since the introduction of the Welsh Language Act of 1993 and Government of Wales Act of 1998 of requiring English and Welsh languages to be treated on a equal basis.

Additional information about the Welsh language can be found at the official Web site of the Welsh Language Board:

http://www.bwrdd-yr-iaith.org.uk

AIX V6.1 provides full Universal-Coded Character Set (UCS) enablement for the Welsh (Cymraeg) language through a dedicated UCS Transformation Format UTF-8 locale. The UCS language and territory designation for Welsh is given by CY_GB.

Figure 10-12 provides a complete list of Welsh letters and their related names.

Letter	Name	Letter	Name
A a	â	Ll ll	ell
B b	bî	M m	èm
C c	èc	N n	en
Ch ch	èch	O o	ô
D d	dî	P p	pî
Dd dd	èdd	Ph ph	ffî
E e	ê	R r	èr
F f	èf	Rh rh	rhî, rhô
Ff ff	èff	S s	ès
G g	èg	T t	tî
Ng ng	èng	Th th	èth
H h	âets, hâ	U u	û
I i	î	W w	ŵ
j	jay	Y y	ŷ
L l	èl		

Figure 10-12 Welsh alphabet

The English and Welsh locales for the UK are very similar. The date and time formats, calendar, and time zone information is the same, yet there are some differences to be noted:

- The Welsh alphabet has 29 letters, while the English alphabet has 26.
  - The Welsh locale does not include k, q, v, x, or z.
  - The Welsh alphabet includes digraph ch, dd, ff, ng, ll, ph, rh, and th.

    For a text string, the character count will likely be different than the letter count.
- The Welsh alphabet has a different sort order than the English alphabet and the digraph characters further influence collation.
- The Welsh keyboard has the same layout as the UK keyboard; however, there are alternate key sequences required to input the diacritic marks.

Welsh vowels can be modified by diacritic marks (for example, â and ô).

10.5.1 Packaging and installation

The Welsh locale definitions and the underlying support are delivered through the following, separately installable filesets that are grouped and distributed in two entities:

- bos.loc.utf.CY_GB fileset
- X11.loc.CY_GB package comprised of the filesets:
  - X11.loc.CY_GB.base.lib
  - X11.loc.CY_GB.base.rte
  - X11.loc.CY_GB.Dt.rte

The scope of the files in the bos.loc.utf.CY_GB fileset is limited to providing locale support for the AIX base operating system, while the X11.loc.CY_GB package will add the locale support for the X11 environment. X11.loc.CY_GB.Dt.rte specifically addresses the requirement of the Common Desktop Environment. Several filesets will be automatically installed if the installation process cannot find them on the system:

- bos.loc.com.utf (co-requisite to bos.loc.utf.CY_GB)
- X11.fnt.ucs.ttf (co-requisite to X11.loc.CY_GB.base.rte)

To verify the complete list of dependencies, you can look at the messages displayed during the installation of the Welsh locale or you can examine the output of the relevant **`lslpp -p`** command after you install the filesets on your system.

As shown in Figure 10-3 on page 380, during the setup of an AIX installation system, administrators can set the primary language environment to use the predefined triple setup of Welsh as cultural convention, English (United States) as language, and Welsh as keyboard.

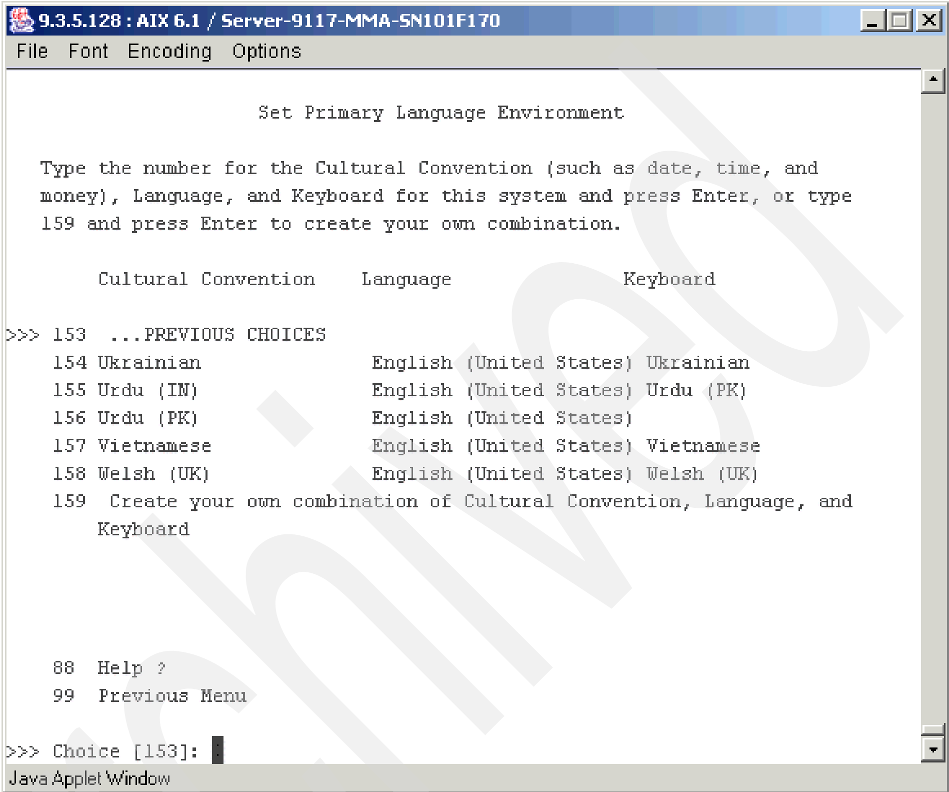


Figure 10-13 Set Primary Language Environment installation menu

After the completion of the base operating system installation, you can use the **lslpp** command to determine which base system locales have been installed as a consequence of the predefined Welsh primary language environment settings that were chosen:

```
# lslpp -l bos.loc*
  Fileset                      Level  State      Description
  ----------------------------------------------------------------------------
Path: /usr/lib/objrepos
  bos.loc.com.utf            6.1.0.0  COMMITTED  Common Locale Support - UTF-8
  bos.loc.utf.EN_US          6.1.0.0  COMMITTED  Base System Locale UTF Code
                                                 Set - U. S. English
  bos.loc.utf.CY_GB          6.1.0.0  COMMITTED  Base System Locale UTF Code
                                                 Set - Welsh
```

Note that in addition to the UTF-8 Welsh locale, the UTF-8 US English locale has been installed too.

Depending on the graphics related characteristics of a given system, you will also encounter additional filesets for support of the Common Desktop Environment and AIXwindows (X11) locales:

```
# lslpp -l X11.loc.*
  Fileset                      Level  State      Description
  ----------------------------------------------------------------------------
Path: /usr/lib/objrepos

  X11.loc.CY_GB.Dt.rte       6.1.0.0  COMMITTED  CDE Locale Configuration -
                                                 Welsh
  X11.loc.CY_GB.base.lib     6.1.0.0  COMMITTED  AIXwindows Client Locale
                                                 Config - Welsh
  X11.loc.CY_GB.base.rte     6.1.0.0  COMMITTED  AIXwindows Locale
                                                 Configuration - Welsh

... omitted lines ...

Path: /etc/objrepos
  X11.loc.CY_GB.Dt.rte       6.1.0.0  COMMITTED  CDE Locale Configuration -
                                                 Maltese
```

If you would like to add Welsh national language support to an existing AIX installation, you can use the SMIT **mlang** locale fast path to access the Change/Show Primary Language Environment or the Add Additional Language Environments SMIT menus.

10.5.2 Locale definitions, keyboard definition, and input methods

A locale is made up of the language, territory, and code set combination used to identify a set of language conventions. The language conventions are grouped in six categories to include information about collation, case conversion, character classification, the language of message catalogs, date-and-time representation, the monetary symbol, and numeric representation. National language support uses the environment variables LC_COLLATE, LC_CTYPE, LC_MESSAGES, LC_MONETARY, LC_NUMERIC, and LC_TIME to define the current values for their respective categories and to influence the selection of locales.

As mentioned previously, the language and territory designation for Welsh national language support is CY_GB and after you configure your environment to use Welsh national language support, you will get the following output by using the **locale** command:

```
# locale
LANG=CY_GB
LC_COLLATE="CY_GB"
LC_CTYPE="CY_GB"
LC_MONETARY="CY_GB"
LC_NUMERIC="CY_GB"
LC_TIME="CY_GB"
LC_MESSAGES="CY_GB"
LC_ALL=
```

Any UTF-8-based locale installed on an AIX system will provide support for the euro symbol. As Wales is among the countries that may have to actively use the euro, the Welsh locale will deliver the input methods and the keyboard maps required to enter the euro symbol through the keyboard. An additional LC_MONETARY locale is available to enable the euro currency formatting. This locale is identified by the suffix @euro. As such, the alternate euro currency format is invoked when LC_MONETARY=CY_GB@euro is specified through the locale environment variables, or with the setlocale subroutine. The **locale** command output below shows the required environment variable settings to support for euro currency formatting:

```
# locale
LANG=CY_GB
LC_COLLATE="CY_GB"
LC_CTYPE="CY_GB"
LC_MONETARY=CY_GB@euro
LC_NUMERIC="CY_GB"
LC_TIME="CY_GB"
LC_MESSAGES="CY_GB"
LC_ALL=
```

To allow dual currency support, the Welsh locales also provide the CY_GB@preeuro locale for the LC_MONETARY category. The CY_GB@preeuro locale is linked to the default locale for traditional national currency formatting requirements of users and applications.

AIX V6.1 introduces the new keyboard ID 166W in support for the Welsh national language environment. As the keyboard ID indicates, the layout of the Welsh 166W keyboard is implemented as an additional layer to the United Kingdom English (en_GB ISO-08859-1) keyboard 166. The AltGr and Shift+AltGr modifiers are used to enter the Welsh layer of the 166W keyboard layout. For a complete description of the Welsh keyboard layout, refer to the Welsh Keyboard Translation Table in Chapter 2, “Keyboard Translation Tables”, in *AIX Version 6.1 Keyboard Technical Reference*, SC23-6614.

AIX supports two different types of keyboards: low function terminal (LFT) and X server keyboards.

The LFT environment is not capable of handling multi-byte code sets such as UTF-8 or complex text (layout oriented) languages. Therefore, the LFT key map for CY_GB.lftkeymap in the /usr/lib/nls/loc directory is implemented as a symbolic link to C.lftkeymap:

```
# cd /usr/lib/nls/loc
# ls -l CY_GB.lftkeymap | cut -c59-
CY_GB.lftkeymap -> /usr/lib/nls/loc/C.lftkeymap
```

Keyboard mapping within the AIX X11 environment is based on the locally attached keyboard. When you start a local X session (through the **xinit** command, the X Display Manager, or the Common Desktop Environment), startup scripts will call the **/usr/bin/X11/xmodmap** command and load the keyboard map for the keyboard language determined by the **/usr/bin/X11/querykbd** command. The **xmodmap** command defines the mapping of the Shift, Lock, and Alt-Graphic (AltGr) keys. The related **xmodmap** command expressions for the Welsh keyboard are defined in the /usr/lpp/X11/defaults/xmodmap/CY_GB/keyboard file.

Note: The **xmodmap** command is not called if the display is remote. Keyboard mapping is performed on the local display. Consult your local configuration guides for assistance configuring remote keyboard mapping.

AIX V6.1 provides the standard UNIVERSAL input method as well as the traditional single-byte input method through the CY_GB.im and the CY_GB.UTF-8.im files, respectively. Both input methods are related by the use of the same UNIVERSAL input method configuration file UNIVERSAL.imcfg. The input method files, the related configuration files, and the input method keymap definition files are located in the /usr/lib/nls/loc directory:

```
# ls -l CY_GB*im* | cut -c59-
CY_GB.UTF-8.im -> /usr/lib/nls/loc/sbcs.im
CY_GB.UTF-8.im__64 -> /usr/lib/nls/loc/sbcs.im__64
CY_GB.UTF-8.imcfg -> /usr/lib/nls/loc/UNIVERSAL.imcfg
```

```
CY_GB.UTF-8.imkeymap
CY_GB.im -> /usr/lib/nls/loc/UNIVERSAL.im
CY_GB.im__64 -> /usr/lib/nls/loc/UNIVERSAL.im__64
CY_GB.imcfg -> /usr/lib/nls/loc/UNIVERSAL.imcfg
CY_GB.imkeymap -> /usr/lib/nls/loc/CY_GB.UTF-8.imkeymap
```

10.6 Olson time zone support

"The public-domain time zone database contains code and data that represent the history of local time for many representative locations around the globe. It is updated periodically to reflect changes made by political bodies to time zone boundaries, UTC offsets, and daylight-saving rules. This database (often called tz or zoneinfo) is used by several implementations, [...].

Each location in the database represents a national region where all clocks keeping local time have agreed since 1970. Locations are identified by continent or ocean and then by the name of the location, which is typically the largest city within the region. For example, America/New_York represents most of the US eastern time zone; America/Phoenix represents most of Arizona, which uses mountain time without daylight saving time (DST); America/Detroit represents most of Michigan, which uses eastern time but with different DST rules in 1975; and other entries represent smaller regions like Starke County, Indiana, which switched from central to eastern time in 1991 and switched back in 2006."[1]

The public-domain time zone database is also widely known as the Olson time zone database and is the architecture on which the International Components for Unicode (ICU) and the Common Locale Data Repository (CLDR) time zone support relies.

In previous AIX releases, the method by which the operating system supports time zone conventions is based on the POSIX time zone specification. In addition to this industry standard approach, AIX V6.1 recognizes and processes the Olson time zone naming conventions to facilitate support for a comprehensive set of time zones.

This enhancement leverages the uniform time zone naming convention of the Olson database to offer an intuitive set of time zone values that can be assigned to the TZ time zone environment variable.

[1] Source: *Source for Time Zone and Daylight Saving Time Data*, found at http://www.twinsun.com/tz/tz-link.htm.

To implement the Olson time zone feature, AIX V6.1 utilizes the ICU library APIs that are shipped in the ICU4C.rte fileset and installed by default on any AIX V6.1 system. For more detailed information about ICU4C support in AIX V6.1, refer to 10.8, “International Components for Unicode” on page 411.

Note: Time zone definitions conforming to the POSIX specification are still supported and recognized by AIX. AIX checks the TZ environment variable to determine if the environment variable follows the POSIX specification rules. If the TZ environment variable does not match the POSIX convention, AIX calls the ICU library to get the Olson time zone translation.

The use of the Olson database for time zone support within AIX provides significant advantages over the traditional POSIX rules. One of the biggest advantages is that Olson database maintains a historical record of what the time zone rules were at given points in time, so that if the rules change in a particular location, dates and times can be interpreted correctly both in the present and past. A good example of this is the US state of Indiana, which just began using daylight saving time in the year 2006. Under the POSIX implementation, Indiana would have to set its time zone value to EST5EDT, which would format current dates correctly using daylight saving time, but would also format times from previous years as though they were on daylight saving time, which is incorrect. Use of the ICU API set for time zones also allows for localized display names for time zones. For example, Central Daylight Saving Time would have an abbreviation of CDT for all locales under a POSIX implementation, but under ICU/Olson, it displays properly as HAC (Heure Avancée du Centre) in a French locale.

As in previous AIX releases, system administrators can rely on the Systems Management Interface Tool (SMIT) to configure the time zone by using system defined values for the TZ environment variable. To accomplish this task, enter the main SMIT menu and select System Environments → Change / Show Date and Time to access the Change Time Zone Using System Defined Values menu. Alternatively, the SMIT fast path **chtz_date** will directly open the Change / Show Date and Time menu. Selecting the Change Time Zone Using System Defined Values option will prompt SMIT to open the Select COUNTRY or REGION menu, as shown in Figure 10-14 on page 409.

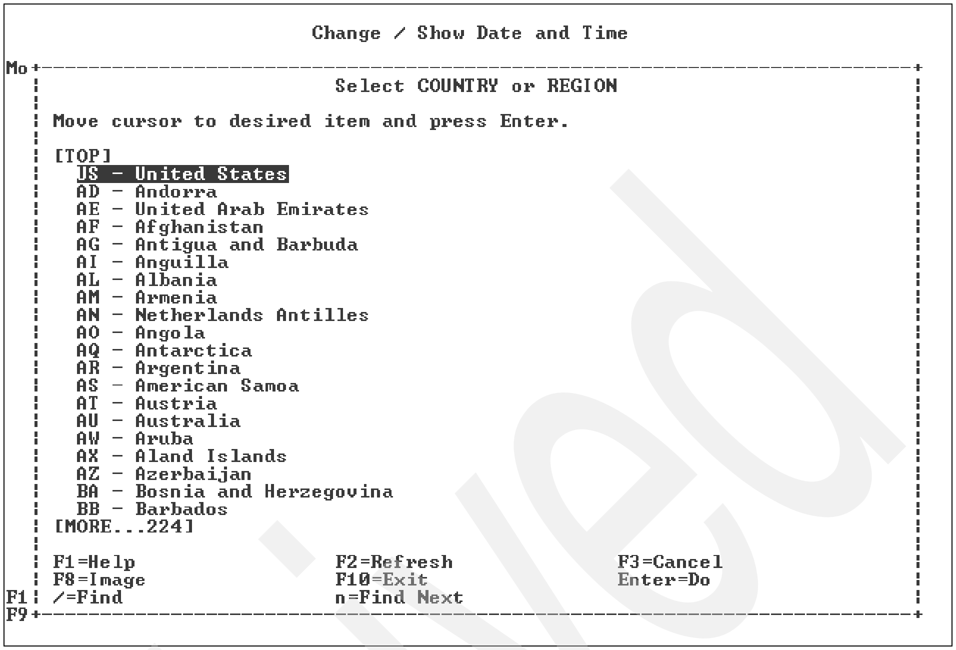


Figure 10-14 SMIT menu to select country or region for Olson time zone

SMIT uses the undocumented `/usr/lib/nls/lstz -C` command to produce the list of available countries and regions. Note that undocumented commands and features are not officially supported for customer use, are not covered by the AIX compatibility statement, and may be subject to change without notice.

After you have chosen the country or region in the Select COUNTRY or REGION menu, a new selection menu will list all available time zones for the country or region in question. Figure 10-15 shows the time zone options that are available for the United States. The selected value of the first column will be passed by SMIT to the **chtz** command, which in turn will change the TZ variable value in the /etc/environment system level configuration file. As with previous AIX releases, time zone configuration changes always require a system reboot to become effective.

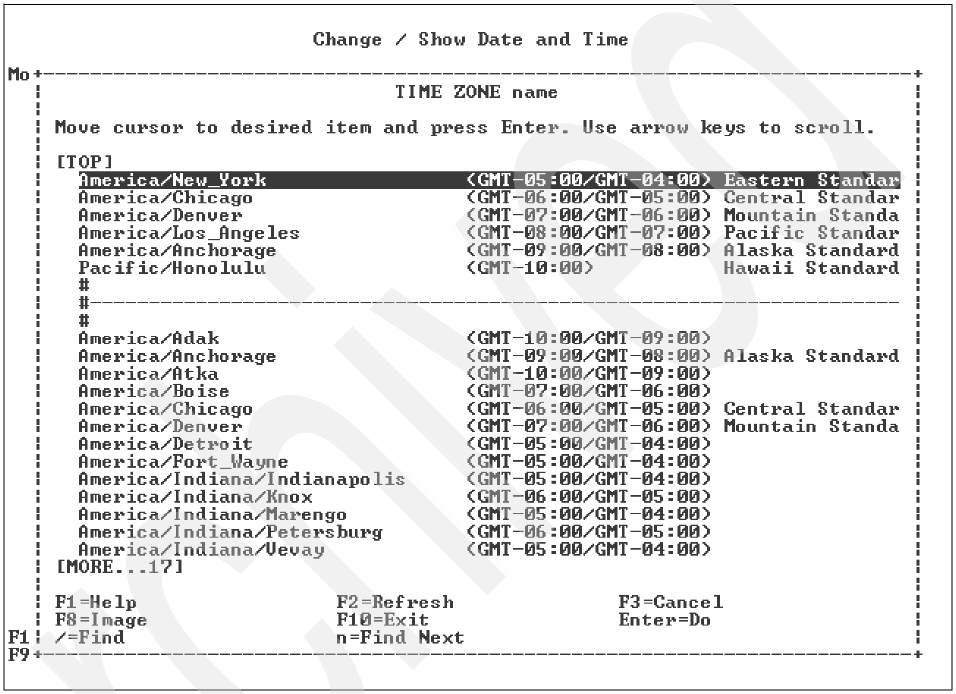


Figure 10-15 SMIT menu to select the time zone for a given country or region

SMIT uses the internal **/usr/lib/nls/lstz -c** command to produce the list of available time zones for a given country and region. The -c flag uses a country or region designation as the input parameter. The **/usr/lib/nls/lstz -C** command provides a list of available input parameters. As such, the listing in Figure 10-15 is produced by the **/usr/lib/nls/lstz -c US** command. The **/usr/lib/nls/lstz** command used without any flag provides a full list of all Olson time zones available on AIX. Note that undocumented commands and features are not officially supported for customer use, are not covered by the AIX compatibility statement, and may be subject to change without notice.

10.7 Unicode 5.0 support

As part of the continuous ongoing effort to adhere to the most recent industry standards, AIX V6.1 provides the necessary enhancements to the existing Unicode locales in order to bring them up to compliance with the latest version of the Unicode standard, which is Version 5.0, as published by the Unicode Consortium. The Unicode is a standard character coding system for supporting the worldwide interchange, processing, and display of the written texts of the diverse languages used throughout the world. Unicode 5.0 defines standardized character positions for over 99,000 glyphs in total.

For in-depth information about Unicode 5.0, visit the official Unicode home page at:

http://www.unicode.org

10.8 International Components for Unicode

International Components for Unicode (ICU) provides, through one of the premier software internationalization packages, robust features that allow programmers to effectively work with Unicode data and create globalized applications.

More detailed information about ICU can be found at the official International Components for Unicode home page at:

http://www.icu-project.org

AIX V6.1 includes the ICU4C 3.6 cross-platform Unicode based globalization libraries for C/C++ as part of the base operating system. AIX V6.1 provides both 32- and 64-bit versions of the ICU libraries, which are tightly integrated into the operating system through system level links in /usr/bin and /usr/lib and system level include files in /usr/include. AIX delivers the ICU4C support through the following filesets:

ICU4C.rte Libraries and commands

ICU4C.adt Header files

ICU4C.man.en_US Manual pages

Because the Olson time zone support in AIX relies on the ICU4C services, the ICU4C.rte fileset is listed in the BOS.autoi file in /usr/sys/inst.data/sys_bundles directory to ensure that the ICU4C support will always be installed by default. 10.6, “Olson time zone support” on page 407 provides additional information about the new AIX V6.1 time zone feature. The ICU4C.rte file set is also listed in the bundle definition file for Common Criteria evaluated systems, CC_EVAL.BOS.autoi, and in the bundle definition file for Secure by Default installations, SbD.BOS.autoi. Consequently, the ICU4C services will also be available in environments with special, highly demanding security requirements.

The shared libraries are installed in the /usr/lib directory and the relevant symbolic links are added to /usr/icu4c/lib:

```
# cd /usr/icu4c/lib
l# ls -l | cut -c59-
libicudata.a -> /usr/lib/libicudata.a
libicui18n.a -> /usr/lib/libicui18n.a
libicuio.a -> /usr/lib/libicuio.a
libicule.a -> /usr/lib/libicule.a
libiculx.a -> /usr/lib/libiculx.a
libicutu.a -> /usr/lib/libicutu.a
libicuuc.a -> /usr/lib/libicuuc.a
```

The header files are installed in the /usr/icu4c/include/unicode or /usr/icu4/include/layout directories and symbolic links are set up in /usr/include, which makes the header files accessible along the normal compilation path:

```
# cd /usr/include
# ls -l layout | cut -c59-
layout -> /usr/icu4c/include/layout
# ls -l unicode | cut -c59-
unicode -> /usr/icu4c/include/unicode
```

The commands are installed in the /usr/icu4c/bin directory and the relevant symbolic links are set up in /usr/bin accordingly:

```
# ls -l /usr/bin/ | grep icu4c | cut -c59-
derb -> /usr/icu4c/bin/derb
genbrk -> /usr/icu4c/bin/genbrk
genccode -> /usr/icu4c/bin/genccode
gencmn -> /usr/icu4c/bin/gencmn
gencnval -> /usr/icu4c/bin/gencnval
genctd -> /usr/icu4c/bin/genctd
genrb -> /usr/icu4c/bin/genrb
gensprep -> /usr/icu4c/bin/gensprep
genuca -> /usr/icu4c/bin/genuca
icu-config -> /usr/icu4c/bin/icu-config
icupkg -> /usr/icu4c/bin/icupkg
icuswap -> /usr/icu4c/bin/icuswap
makeconv -> /usr/icu4c/bin/makeconv
pkgdata -> /usr/icu4c/bin/pkgdata
```

11

Hardware and graphics support

This chapter discusses the new hardware support and graphic topics new in AIX Version 6.1, arranged by the following topics:

11.1 Hardware support

AIX V6.1 exclusively supports 64-bit Common Hardware Reference Platform (CHRP) machines with selected processors:

- PowerPC 970
- POWER4
- POWER5
- POWER6

To see if you have a supported machine, log into the machine as the root user, and run the following command:

```
# prtconf | grep 'Processor Type'
```

AIX V6.1 does not support the following processor architectures:

- RS64
- POWER3
- 604

Certain machines may require a firmware update in order to run AIX V6.1. For the latest prerequisites, refer to the *AIX V6.1 Release Notes*, found at:

http://publib14.boulder.ibm.com/infocenter/systems/aix/topic/com.ibm.aix.resources/61relnotes.htm

AIX V6.1 requires 512 MB of physical memory and a minimum of 2.2 GB of physical disk space for a default base AIX installation (with CDE).

11.2 Universal Font Scaling Technology Version 5

The AIX V6.1 base operating system delivers the Universal Font Scaling Technology (UFST) Version 5.0.1 font rasterizer licensed from the Monotype Imaging company (http://www.monotypeimaging.com). The AIX V6.1 UFST Version 5.0.1 support is an update to the UTFS Version 4.6 as used by AIX 5L V5.3 and provides an advanced TrueType font rasterizer to the X Window System environment on AIX.

The AIX V6.1 UFST Version 5.0.1 functionality is embedded in the AIX X Server (AIXWindows) runtime environment, which is delivered through the X11.base.rte fileset, and in the AIX X font server code, which is shipped in the X11.fnt.fontServer fileset.

"The UFST subsystem reads, interprets and processes hinted font data to rapidly generate scaled character bitmaps, graymaps or grid-aligned outlines. The fast, compact solution offers a lower ROM cost than alternative font rendering systems and is the only one that uses industry-standard trademarked font names and font metrics..."[1]

11.3 X Window System Version 11 Release 7.1

AIX V6.1 contains X Window System libraries, headers, and some applications that have been updated for X Window System Version 11 Release 7.1 (X11R7.1). For detailed release specific information about the AIXWindows runtime environment on AIX V6.1 refer, to the /usr/lib/X11/README file, which is in the X11.base.rte fileset.

11.3.1 X11R5, X11R6.1, and X11R7.1 compatibility issues

The libraries shipped by IBM with X11R7.1 are backward-compatible and the client applications, which access these libraries, will work as on previous releases of AIX, except as noted below.

As on earlier releases of AIX, IBM also ships X11R3, X11R4, X11R5, and X11R6 compatibility installation options for maximum customer flexibility. In this way, client applications experience no problems with compatibility.

There are a few notable differences due to the X11R7.1 updates:

- Most of the new X11 R7.1 header files now only contain full ANSI function prototypes. This may cause the compiler to find problems that were not apparent before.
- The file /usr/lpp/X11/defaults/xserverrc is the script used by the **xinit**, **xdm**, and **dtlogin** commands to start the X Window System. This script has been modified so that the default visual will now be 24-bit TrueColor instead of 8-bit PseudoColor. Some old applications may not work in the 24-bit TrueColor visual. In this case, the old visual can be restored by commenting out the following line in the xserverrc file:

  ```
  EXTENSIONS="$EXTENSIONS -cc 4"
  ```

[1] Source: *UFST – Universal Font Scaling Technology*, found at http://www.monotypeimaging.com/ProductsServices/ufts.aspx

- An updated version of terminal emulator for the X Window System, **xterm** is included in AIX V6.1. This version of **xterm** required an update to the xterm terminfo information. The updated terminfo may cause problems with other terminal emulators that expect the older version. Separate compatibility terminfo definitions (xterm-old and xterm-r6) are provided for use in such situations, and are accessed by setting the TERM environment variable.

The AIX V6.1 X system uses the X Consortium release 6 version of the X Window System.

11.3.2 AIX V6.1 X Client enhancements

AIX V6.1 provides new version of the X Window System terminal emulator **xterm** program and a new version of the X Display Manager **xdm** program. Both applications were updated to X11R7.1.

The new **xterm** terminal emulator requires a new terminfo file that does not work well with older versions of **xterm** in previous AIX releases. However, terminfo compatibility files are provided for use by older versions of **xterm**. You can access these by setting the TERM environment variable to xterm-r6 or xterm-old.

For example, if you using the Korn shell, you would run one of the following commands after you **telnet** into an AIX V6.1 system from an xterm on an AIX 5L system:

```
export TERM=xterm-r6
```

or

```
export TERM=xterm-old
```

Either of these commands will start using a terminfo file designed to support the older (X11R6) version of **xterm**. The xterm-r6 and xterm-old files in the /usr/share/lib/terminfo/x directory are identical.

The new X11R7.1 version of X Display Manger **xdm** only supports PAM authentication. For security reasons, the default **xdm** configuration disables remote login. To enable remote login, the following files need to be modified:

- /usr/lib/X11/xdm/xdm-config
- /usr/lib/X11/xdm/Xaccess

11.3.3 X11R5, X11R6, and X11R7.1 coexistence

X11R7.1 is considered binary compatible with X11R5 and X11R6. Therefore, any applications that used to be running on X11R5/X11R6 should run on X11R7.1 with no problems.

For existing applications that do not run on the X11R71. libraries, the X11R6 Xlibs are shipped in the following fileset:

```
# lslpp -l X11.compat.lib.X11R6
  Fileset                      Level  State      Description
  ----------------------------------------------------------------------------
Path: /usr/lib/objrepos
  X11.compat.lib.X11R6       6.1.0.0  COMMITTED  AIXwindows X11R6 Compatibility
                                                 Libraries
```

In that fileset, you will find the /usr/lpp/X11/lib/R6 directory, which contains all the R6 libraries required for the system.

The X11 toolkit specifically checks to make sure you are not running against X11R5 and X11R6 in the same application by checking the X Version at runtime. The following error is generated when you try to run an applications that is linked to Motif1.2, which was built using X11R7.1, and libXt.a, which was built using X11R6:

```
Warning: Widget class VendorShell version mismatch (recompilation needed):
              widget 11007 vs. intrinsics 11006.
```

Since the Motif1.2 (shr4.o) object shipping in libXm.a is compiled against X11R7.1, a version of Motif1.2 that was compiled against X11R6 is also shipping, so that anyone needing to use the X11R6 libraries would also have a Motif1.2 library to run against. This X11R6 Motif 1.2 is found in /usr/lpp/X11/lib/R6/libXm.a.

11.4 32 TB physical memory support

Previous AIX versions supported physical memory up to a maximum of 16 TB. The virtual memory manager (VMM) in AIX V6.1 is enhanced to address a maximum of 32 TB RAM.

11.5 Withdrawal of the 32-bit kernel

In previous AIX versions, multiple kernels were shipped. The important milestones are:

1993 AIX V4.1 introduces multiprocessor kernel (unix_mp).

2000 AIX 5L V5.0 introduces the 64-bit kernel (unix_64).

2004 The uniprocessor kernel (unix_up) has been removed in AIX 5L V5.3.

In AIX 5L V5.3 and AIX 5L V5.2 ML 5200-03, the 64-bit kernel was installed by default on POWER5 and newer systems.

2007 The 32-bit kernel is removed in AIX V6.1

Beginning with AIX V6.1, the AIX operating system will simplify its kernel environment by providing only the 64-bit kernel. Device drivers and kernel extensions that are 32-bit only are not supported. Dual-mode (32/64-bit) kernel extensions built on AIX 5L will continue to run on AIX V6.1, but only in 64-bit mode.

AIX V6.1 is binary compatible to both 32-bit and 64-bit applications created in previous AIX 5L versions. Further information and a list of restrictions are published on the IBM AIX Binary compatibility site found at:

http://www-03.ibm.com/systems/p/os/aix/compatibility/index.html

A

Transport-independent RPC

AIX V6.1 now formally supports the AIX base operating system related subset of the transport-independent remote procedure call (TI-RPC) routines as ported from the ONC+ 2.2 source distribution. The AIX V6.1 TI-RPC client and server interfaces are listed by API function class in Table A-1 on page 420.

Additionally, the RPCSEC_GSS security services interface routines of the General Security Services (GSS) API are now officially supported and documented in the AIX V6.1 standard publication. The following RPCSEC_GSS subroutines are described in detail by the AIX V6.1 standard documentation:

- rpc_gss_seccreate()
- rpc_gss_set_defaults()
- rpc_gss_max_data_length()
- rpc_gss_set_svc_name()
- rpc_gss_getcred()
- rpc_gss_set_callback()
- rpc_gss_get_principal_name()
- rpc_gss_svc_max_data_length()
- rpc_gss_get_error()
- rpc_gss_get_mechanisms()
- rpc_gss_get_mech_info()
- rpc_gss_get_versions()
- rpc_gss_is_installed()
- rpc_gss_mech_to_oid()
- rpc_gss_qop_to_num()

Table A-1 TI-RPC client and server interfaces

API classification	Description	API names	Routine classification[a]	Implemented in libnsl.a[b]	Exported in libnsl.a	Available in libc.a[b]
RPC_SVC_REG	Routines that allow the RPC servers to register themselves with rpcibnd(), and associate the given program and version number with the dispatch function.	rpc_reg	S	✓	✓	
		svc_reg	E	✓	✓	
		svc_unreg	E	✓	✓	
		svc_auth_reg	O	✓	✓	
		xprt_register	B	✓	✓	✓
		xprt_unregister	B	✓	✓	✓
RPC_SVC_CREATE	Routines that are related to the creation of service handles.	svc_control	T	✓[M]		
		svc_create	T	✓	✓	
		svc_destroy	T	✓[M]		✓[M]
		svc_dg_create	B	✓	✓	
		svc_fd_create	B	✓	✓	
		svc_raw_create	B	✓	✓	
		svc_tli_create	E	✓	✓	
		svc_tp_create	I	✓	✓	
		svc_vc_create	B	✓	✓	

API classification	Description	API names	Routine classification[a]	Implemented in libnsl.a[b]	Exported in libnsl.a	Available in libc.a[b]
RPC_SVC_CALLS	Routines that are associated with the server side of the RPC mechanism. Some of them are called by the server side dispatch function, while others are called when the server is initiated.	svc_dg_enablecache	O	✓	✓	
		svc_done	O	✓	✓	
		svc_exit	O	✓	✓	
		svc_fdset	O	✓[G]	✓	✓[G]
		svc_freeargs	O	✓[M]		✓[M]
		svc_getargs	O	✓[M]		✓[M]
		svc_getreq_common	B	✓	✓	
		svc_getreq_poll	B	✓	✓	
		svc_getreqset	B	✓	✓	✓
		svc_getrpccaller	O	✓[M]		
		svc_max_pollfd	O	✓[G]	✓	
		svc_pollfd	O	✓[G]	✓	
		svc_run	O	✓	✓	✓
		svc_sendreply	O	✓	✓	✓
RPC_SVC_ERR	Routines called by the server side dispatch function if there is any error in the transaction with the client.	svcerr_auth	O	✓	✓	✓
		svcerr_decode	O	✓	✓	✓
		svcerr_noproc	O	✓	✓	✓
		svcerr_noprog	O	✓	✓	✓
		svcerr_progvers	O	✓	✓	✓
		svcerr_systemerr	O	✓	✓	✓
		svcerr_weakauth	O	✓	✓	✓

API classification	Description	API names	Routine classification[a]	Implemented in libnsl.a[b]	Exported in libnsl.a	Available in libc.a[b]
RPC_CLNT_CREATE	Routines that facilitate services related to the creation of client handles.	clnt_control	T	✓[M]		✓[M]
		clnt_create	T	✓	✓	✓
		clnt_create_timed	T	✓	✓	
		clnt_create_vers	T	✓	✓	✓
		clnt_create_vers_timed	T	✓	✓	
		clnt_destroy	T	✓[M]		✓[M]
		clnt_dg_create	B	✓	✓	
		clnt_door_create	I	✓	✓	
		clnt_pcreateerror	O	✓	✓	✓
		clnt_raw_create	B	✓	✓	
		clnt_spcreateerror	O	✓	✓	✓
		clnt_tli_create	E	✓	✓	
		clnt_tp_create	I	✓	✓	
		clnt_tp_create_timed	I	✓	✓	
		clnt_vc_create	B	✓	✓	
		rpc_createerr	O	✓[G]	✓	✓[G]

API classification	Description	API names	Routine classification[a]	Implemented in libnsl.a[b]	Exported in libnsl.a	Available in libc.a[b]
RPC_CLNT_CALLS	Routines that handle the client side of the remote procedure calls and related error conditions.	clnt_call	T	✓[M]		✓[M]
		clnt_freeres	O	✓[M]		✓[M]
		clnt_geterr	O	✓[M]		✓[M]
		clnt_perrno	O	✓	✓	
		clnt_perror	O	✓	✓	
		clnt_sperrno	O	✓	✓	✓
		clnt_sperror	O	✓	✓	✓
		rpc_broadcast	S	✓	✓	
		rpc_broadcast_exp	S	✓	✓	
		rpc_call	S	✓	✓	
RPC_CLNT_AUTH	Routines normally called in support of authentication after creation of the client handle.	auth_destroy	SC	✓[M]		✓[M]
		authnone_create				✓
		authsys_create	SC	✓	✓	
		authsys_create_default	SC	✓	✓	

API classification	Description	API names	Routine classification[a]	Implemented in libnsl.a[b]	Exported in libnsl.a	Available in libc.a[b]
SECURE_RPC	Routines supporting DES encryption-based authentication.	authdes_getucred	SC	✓	✓	✓
		authdes_seccreate	SC	✓	✓	
		getnetname	SC	✓	✓	✓
		host2netname	SC	✓	✓	✓
		key_decryptsession	SC	✓	✓	✓
		key_encryptsession	SC	✓	✓	✓
		key_gendes	SC	✓	✓	✓
		key_secretkey_is_set	SC	✓	✓	
		key_setsecret	SC	✓	✓	✓
		netname2host	SC	✓	✓	✓
		netname2user	SC	✓	✓	✓
		user2netname	SC	✓	✓	✓
RPC_CONTROL	Function that allows applications to set and modify global attributes that apply to clients as well as server functions.	rpc_control	T	✓	✓	
RPCBIND	Routines that allow you to make procedure calls to the RPC bind service.	rpcb_getaddr	E	✓	✓	
		rpcb_getmaps	E	✓	✓	
		rpcb_gettime	E	✓	✓	
		rpcb_rmtcall	E	✓	✓	
		rpcb_set	E	✓	✓	
		rpcb_unset	E	✓	✓	

API classification	Description	API names	Routine classification[a]	Implemented in libnsl.a[b]	Exported in libnsl.a	Available in libc.a[b]
GETRPCBYNAME	Functions to obtain entries for RPC services. An entry may come from any of the sources for rpc specified in the /etc/nsswitch.conf file.	endrpcent				✓
		getrpcbyname				✓
		getrpcbyname_r				✓
		getrpcbynumber				✓
		getrpcbynumber_r				✓
		getrpcent				✓
		getrpcent_r				✓
		setrpcent				✓

API classification	Description	API names	Routine classification[a]	Implemented in libnsl.a[b]	Exported in libnsl.a	Available in libc.a[b]
RPC_SOC	Obsolete routines provided in support of backward compatibility.	authdes_create		✓		✓
		authunix_create		✓M		✓
		authunix_create_default		✓M		✓
		callrpc		✓		✓
		clnt_broadcast		✓		✓
		clntraw_create		✓		✓
		clnttcp_create		✓	✓	✓
		clntudp_bufcreate		✓	✓	✓
		clntudp_create		✓	✓	✓
		get_myaddress		✓		✓
		getrpcport		✓		✓
		pmap_getmaps		✓	✓	✓
		pmap_getport		✓	✓	✓
		pmap_rmtcall		✓	✓	✓
		pmap_set		✓	✓	✓

API classification	Description	API names	Routine classification[a]	Implemented in libnsl.a[b]	Exported in libnsl.a	Available in libc.a[b]
RPC_SOC (cont.)	Obsolete routines provided in support of backward compatibility.	pmap_unset		✓	✓	✓
		registerrpc		✓		✓
		svc_fds		✓M		✓M
		svc_getcaller		✓M		✓M
		svc_getreq		✓		✓
		svc_register		✓	✓	✓
		svc_unregister		✓	✓	✓
		svcfd_create		✓		✓
		svcraw_create		✓		✓
		svctcp_create		✓	✓	✓
		svcudp_bufcreate		✓		✓
		svcudp_create		✓	✓	✓
		xdr_authunix_parms		✓		✓

a. Routine classification legend:

B Bottom-level routines (standard interface)
E Expert-level routines (standard interface)
I Intermediate-level routines (standard interface)
O Other routines (standard interface)
S Simplified interface routines
SC Secure TI-RPC interface routines
T Top-level routines (standard interface)

b. Implementation legend:

M Macro
G Global variable

B

Sample script for tunables

The following script can be used to output all tunables for each tuning command under a corresponding file with the "*xxx*o_help.txt" name. A tar archive format file named prt_tun_help.tar that gathers all these output files can be then uploaded.

```
#!/bin/ksh
#
#   COMPONENT_NAME: PRT_TUN_HELP
#
#   FUNCTIONS: ./prt_tun_help.sh
#
#   ORIGINS: ITSO AUSTIN - SEPTEMBER 2007
#
#   DOCUMENTATION EXAMPLE FOR AIX Version 6 DIFFERENCES GUIDE
#
#   USE "AS IS"
#
#----------------------------------------------------------------------
#
# NAME:         prt_tun
#
# FUNCTION:     Print all tunables for one tuning command.
#
# PARAMETERS:   Tuning command
#
# RETURNS:      None
```

```
#
#---------------------------------------------------------------------#
function prt_tun {
   typeset cmd=$1
   typeset CMD=$2
   echo "Printing $1 tunable description.... $1_help.txt"
   rm ./$1_help.txt
   AIX_LVL=`oslevel -s`
   echo "\t\t----------------------------------------" >>./$1_help.txt
   echo "\t\t\t $2 TUNABLES DESCRIPTION" >>./$1_help.txt
echo "\t\t\t AIX LEVEL : " $AIX_LVL  >>./$1_help.txt
   echo "\t\t----------------------------------------" >>./$1_help.txt
   user_use=
for i in `$1 -F -x | cut -f1 -d ',' `
do
   if [ $i != "##Restricted" ] && [ $i != "tunables" ]; then
   echo "$user_use--------------------------" >>./$1_help.txt
   $1 -h $i >>./$1_help.txt
   else
      if [ $i = "##Restricted" ] ; then
        echo "-----------------------------------------"
>>./$1_help.txt
        echo " ## RESTRICTED PARAMETERS " >>./$1_help.txt
        echo "-----------------------------------------"
>>./$1_help.txt
        user_use="----- Restricted Tunable"
      fi
   fi
done
}
#---------------------------------------------------------------------#
# NAME:        main
#---------------------------------------------------------------------#
prt_tun vmo VMO
prt_tun ioo IOO
prt_tun no NO
prt_tun schedo SCHEDO
prt_tun nfso NFSO
prt_tun raso RASO
echo "Generating prt_tun_help.tar file...."
tar -cf./prt_tun_help.tar ./*_help.txt
```

Next is provided lines abstract of vmo_help.txt file:

```
                ----------------------------------------
```

```
                 VMO TUNABLES DESCRIPTION
                 AIX LEVEL :  6100-00-00
        ----------------------------------------
--------------------------
Help for tunable force_relalias_lite:
Purpose:
If set to 0, a heuristic will be used, when tearing down an mmap
region, to determine when
to avoid locking the source mmapped segment.
Values:
        Default: 0
        Range: 0, 1
        Type: Dynamic
        Unit: boolean
Tuning:
This is a scalability tradeoff, controlled by relalias_percentage,
possibly costing more co
mpute time used. If set to 1, the source segment lock is avoided
whenever possible, regardl
ess of the value of relalias_percentage.
--------------------------
Help for tunable kernel_heap_psize:
... (lines missing for clarity)
-----------------------------------------
 ## RESTRICTED PARAMETERS
-----------------------------------------
----- Restricted Tunable--------------------------
Help for tunable cpu_scale_memp:
Purpose:
Determines the ratio of CPUs per-mempool. For every cpu_scale_memp
CPUs, at least one mempo
ol will be created.
Values:
        Default: 8
        Range: 4 - 64
        Type: Bosboot
        Unit:
Tuning:
Can be reduced to reduce contention on the mempools. (use in
conjunction with the tuning of
 the maxperm parameter).
----- Restricted Tunable--------------------------
... (lines missing for clarity)
```

Abbreviations and acronyms

ABI	Application Binary Interface
AC	Alternating Current
ACL	Access Control List
ACLs	Access Control Lists
AFPA	Adaptive Fast Path Architecture
AIO	Asynchronous I/O
AIX	Advanced Interactive Executive
APAR	Authorized Program Analysis Report
API	Application Programming Interface
ARP	Address Resolution Protocol
ASMI	Advanced System Management Interface
AltGr	Alt-Graphic
Azeri	Azerbaijan
BFF	Backup File Format
BIND	Berkeley Internet Name Domain
BIST	Built-In Self-Test
BLV	Boot Logical Volume
BOOTP	Boot Protocol
BOS	Base Operating System
BSD	Berkeley Software Distribution
CA	Certificate Authority
CATE	Certified Advanced Technical Expert
CD	Compact Disk
CD	Component Dump facility
CD-R	CD Recordable
CD-ROM	Compact Disk-Read Only Memory
CDE	Common Desktop Environment
CEC	Central Electronics Complex
CHRP	Common Hardware Reference Platform
CID	Configuration ID
CLDR	Common Locale Data Repository
CLI	Command-Line Interface
CLVM	Concurrent LVM
CLiC	CryptoLight for C library
CMW	Compartmented Mode Workstations
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CSM	Cluster Systems Management
CT	Component Trace
CUoD	Capacity Upgrade on Demand
DAC	Discretionary Access Controls
DCEM	Distributed Command Execution Manager
DCM	Dual Chip Module
DES	Data Encryption Standard
DGD	Dead Gateway Detection
DHCP	Dynamic Host Configuration Protocol
DLPAR	Dynamic LPAR
DMA	Direct Memory Access
DNS	Domain Name Server
DR	Dynamic Reconfiguration

DRM	Dynamic Reconfiguration Manager
DST	Daylight Saving Time
DVD	Digital Versatile Disk
DoD	Department of Defense
EC	EtherChannel
ECC	Error Checking and Correcting
EGID	Effective Group ID
EOF	End of File
EPOW	Environmental and Power Warning
EPS	Effective Privilege Set
ERRM	Event Response Resource Manager
ESS	Enterprise Storage Server®
EUID	Effective User ID
F/C	Feature Code
FC	Fibre Channel
FCAL	Fibre Channel Arbitrated Loop
FDX	Full Duplex
FFDC	First Failure Data Capture
FLOP	Floating Point Operation
FRU	Field Replaceable Unit
FTP	File Transfer Protocol
GDPS®	Geographically Dispersed Parallel Sysplex™
GID	Group ID
GPFS™	General Parallel File System™
GSS	General Security Services
GUI	Graphical User Interface
HACMP	High Availability Cluster Multiprocessing
HBA	Host Bus Adapters
HMC	Hardware Management Console
HPC	High Performance Computing
HPM	Hardware Performance Monitor
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
Hz	Hertz
I/O	Input/Output
IBM	International Business Machines
ICU	International Components for Unicode
ID	Identification
IDE	Integrated Device Electronics
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IGMP	Internet Group Management Protocol
IP	Internetwork Protocol
IPAT	IP Address Takeover
IPL	Initial Program Load
IPMP	IP Multipathing
IQN	iSCSI Qualified Name
ISC	Integrated Solutions Console
ISSO	Information System Security Officer
ISV	Independent Software Vendor
ITSO	International Technical Support Organization
IVM	Integrated Virtualization Manager
J2	JFS2
JFS	Journaled File System
KAT	Kernel Authorization Table
KCT	Kernel Command Table
KDT	Kernel Device Table
KRT	Kernel Role Table
KST	Kernel Security Table

L1	Level 1
L2	Level 2
L3	Level 3
LA	Link Aggregation
LACP	Link Aggregation Control Protocol
LAN	Local Area Network
LDAP	Light Weight Directory Access Protocol
LED	Light Emitting Diode
LFS	Logical File System
LFT	Low Function Terminal
LMB	Logical Memory Block
LPA	Loadable Password Algorithm
LPAR	Logical Partition
LPP	Licensed Program Product
LPS	Limiting Privilege Set
LUN	Logical Unit Number
LUNs	Logical Unit Numbers
LV	Logical Volume
LVCB	Logical Volume Control Block
LVM	Logical Volume Manager
LWI	Light Weight Infrastructure
MAC	Media Access Control
MBps	Megabytes Per Second
MCM	Multichip Module
MIBs	Management Information Bases
ML	Maintenance Level
MLS	Multi Level Security
MP	Multiprocessor
MPIO	Multipath I/O
MPS	Maximum Privilege Set
MTU	Maximum Transmission Unit
Mbps	Megabits Per Second
NDAF	Network Data Administration Facility
NFS	Network File System
NIB	Network Interface Backup
NIM	Network Installation Management
NIMOL	NIM on Linux
NIS	Network Information Server
NVRAM	Non-Volatile Random Access Memory
ODM	Object Data Manager
OSGi	Open Services Gateway Initiative
OSPF	Open Shortest Path First
PCI	Peripheral Component Interconnect
PIC	Pool Idle Count
PID	Process ID
PIT	Point-in-time
PKI	Public Key Infrastructure
PLM	Partition Load Manager
PM	Performance Monitor
POSIX	Portable Operating System Interface
POST	Power-On Self-test
POWER	Performance Optimization with Enhanced RISC (Architecture)
PPC	Physical Processor Consumption
PPFC	Physical Processor Fraction Consumed
PSPA	Page Size Promotion Aggressiveness Factor
PTF	Program Temporary Fix
PTX®	Performance Toolbox
PURR	Processor Utilization Resource Register
PV	Physical Volume
PVID	Physical Volume Identifier

PVID	Port Virtual LAN Identifier
QoS	Quality of Service
RAID	Redundant Array of Independent Disks
RAM	Random Access Memory
RAS	Reliability, Availability, and Serviceability
RBAC	Role Based Access Control
RCP	Remote Copy
RDAC	Redundant Disk Array Controller
RGID	Real Group ID
RIO	Remote I/O
RIP	Routing Information Protocol
RISC	Reduced Instruction-Set Computer
RMC	Resource Monitoring and Control
RPC	Remote Procedure Call
RPL	Remote Program Loader
RPM	Red Hat Package Manager
RSA	Rivet, Shamir, Adelman
RSCT	Reliable Scalable Cluster Technology
RSH	Remote Shell
RTE	Runtime Error
RUID	Real User ID
S	System Scope
SA	System Administrator
SAN	Storage Area Network
SCSI	Small Computer System Interface
SDD	Subsystem Device Driver
SED	Stack Execution Disable
SFDC	Second Failure Data Capture
SLs	Sensitivity Labels
SMI	Structure of Management Information
SMIT	Systems Management Interface Tool
SMP	Symmetric Multiprocessor
SMS	System Management Services
SMT	Simultaneous Muli-threading
SO	System Operator
SP	Service Processor
SPOT	Shared Product Object Tree
SRC	System Resource Controller
SRN	Service Request Number
SSA	Serial Storage Architecture
SSH	Secure Shell
SSL	Secure Socket Layer
SUID	Set User ID
SVC	SAN Virtualization Controller
TCB	Trusted Computing Base
TCP/IP	Transmission Control Protocol/Internet Protocol
TE	Trusted Execution
TEP	Trusted Execution Path
TLP	Trusted Library Path
TLS	Transport Layer Security
TSA	Tivoli System Automation
TSD	Trusted Signature Database
TTL	Time-to-live
UCS	Universal-Coded Character Set
UDF	Universal Disk Format
UDID	Universal Disk Identification
UFST	Universal Font Scaling Technology
UID	User ID
ULM	User Loadable Module
UPS	Used Privilege Set
VG	Volume Group

VGDA	Volume Group Descriptor Area
VGSA	Volume Group Status Area
VIPA	Virtual IP Address
VLAN	Virtual Local Area Network
VMM	Virtual Memory Manager
VP	Virtual Processor
VPA	Visual Performance Analyzer
VPD	Vital Product Data
VPN	Virtual Private Network
VPSS	Variable Page Size Support
VRRP	Virtual Router Redundancy Protocol
VSD	Virtual Shared Disk
WED	WebSphere Everyplace Deployment V6.0
WLM	Workload Manager
WPAR	Workload Partitions
WPS	Workload Partition Privilege Set

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

For information about ordering these publications, see “How to get Redbooks” on page 440. Note that some of the documents referenced here may be available in softcopy only.

- *AIX 5L Differences Guide Version 5.3 Addendum*, SG24-7414
- *AIX 5L Differences Guide Version 5.3 Edition*, SG24-7463
- *AIX 5L Practical Performance Tools and Tuning Guide*, SG24-6478
- *AIX V6 Advanced Security Features Introduction and Configuration*, SG24-7430
- *Hardware Management Console V7 Handbook*, SG24-7491*IBM System p Advanced POWER Virtualization (PowerVM) Best Practices*, REDP-4194
- *PowerVM Live Partition Mobility on IBM System p*, SG24-7460
- *Implementing High Availability Cluster Multi-Processing (HACMP) Cookbook*, SG24-6769
- *Integrated Virtual Ethernet Adapter Technical Overview and Introduction*, REDP-4340
- *Integrated Virtualization Manager on IBM System p5*, REDP-4061
- *Introduction to Workload Partition Management in IBM AIX Version 6.1*, SG24-7431
- *Linux Applications on pSeries*, SG24-6033
- *NIM from A to Z in AIX 5L*, SG24-7296
- *Partitioning Implementations for IBM eServer p5 Servers*, SG24-7039
- *A Practical Guide for Resource Monitoring and Control (RMC)*, SG24-6615

Other publications

These publications are also relevant as further information sources:

- The following pSeries and System p references can be found at:

  http://www.ibm.com/servers/eserver/pseries/library

  – User guides
  – System management guides
  – Application programmer guides
  – All commands reference volumes
  – Files reference
  – Technical reference volumes used by application programmers

- Detailed documentation about the PowerVM editions and the Virtual I/O Server can be found at:

  https://www14.software.ibm.com/webapp/set2/sas/f/vios/documentation/home.html

- *AIX 5L V5.3 Partition Load Manager Guide and Reference,* SC23-4883
- *Linux for pSeries installation and administration (SLES 9)*, found at:

  http://www-128.ibm.com/developerworks/linux/library/l-pow-pinstall/

- *Linux virtualization on POWER5: A hands-on setup guide*, found at:

  http://www-128.ibm.com/developerworks/edu/l-dw-linux-pow-virutal.html

- *POWER5 Virtualization: How to set up the SUSE Linux Virtual I/O Server*, found at:

  http://www-128.ibm.com/developerworks/eserver/library/es-susevio/

How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

D

U

V

W

X

IBM

Redbooks

IBM AIX Version 6.1 Differences Guide

(1.0" spine)
0.875"<->1.498"
460 <-> 788 pages

IBM AIX Version 6.1 Differences Guide

Redbooks®

AIX - The industrial strength UNIX operating system

AIX Version 6.1 enhancements explained

An expert's guide to the new release

This IBM Redbooks publication focuses on the differences introduced in IBM AIX Version 6.1 when compared to AIX 5L Version 5.3. It is intended to help system administrators, developers, and users understand these enhancements and evaluate potential benefits in their own environments.

AIX Version 6.1 introduces many new features, including workload partitions, advanced security, continuous availability, and managing and monitoring enhancements. There are many other new features available with AIX Version 6.1, and you can explore them all in this publication.

For clients who are not familiar with the enhancements of AIX through Version 5.3, a companion publication, *AIX 5L Differences Guide Version 5.3 Edition*, SG24-7463 is available, along with an addendum, *AIX 5L Differences Guide Version 5.3 Addendum*, SG24-7414, which includes between release enhancements that are available through applying service updates.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-7559-00 ISBN 0738485799