IBM

# Workload Partition Management in IBM AIX Version 6.1

Presents updated technical planning information for AIX V6.1 TL2

Covers new partition mobility, isolation, NIM support, and WPAR Manager features

Provides walk-through examples for AIX system administrators

Shane Brandon
Anirban Chatterjee
Henning Gammelmark
Vijayasekhar Mekala
Liviu Rosca
Arindom Sanyal

Redbooks

ibm.com/redbooks

International Technical Support Organization

# Workload Partition Management in IBM AIX Version 6.1

December 2008

SG24-7656-00

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

**First Edition (December 2008)**

This edition applies to AIX 6.1 TL2.

# Contents

# Notices

## Trademarks

# Preface

Workload partition functionality, originally introduced in 2007 with the release of IBM® AIX® Version 6, is a strategic component of the IBM AIX Operating System. With the release of AIX V6.1 TL2 in November 2008, both the core functionality and features related to managing workload partitions have been improved and expanded. This IBM Redbooks® publication provides an updated introduction and “how to” guide for system administrators and architects using workload partitions in AIX V6.1 TL2. It builds on the original concepts and practices described in the first Redbooks publication about this topic, *Introduction to Workload Partition Management in IBM AIX Version 6.1*, SG24-7431, published in 2007.

In AIX 6.2 TL2, significant enhancements to core workload partition functions and new features have been added. Some of the important feature updates provide more flexibility and support for enhanced mobility, improved isolation, and NIM integration. A new and significantly updated version of IBM Workload Partitions Manager™ for AIX (WPAR Manager), the Web browser-based graphical user interface for managing and monitoring WPARs, is also available. WPAR Manager is a platform management solution that provides a centralized point of control for managing WPARs across a collection of managed systems running AIX.

## 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.

**Shane Brandon** is a Senior IT Specialist in IBM Australia, where he works for Integrated Technology Delivery, Server Systems Operations, supporting Federal Government clients. He has 11 years of experience with AIX and POWER™ Systems, p Series and RS/6000® hardware. He has worked at IBM for 4 years, and is currently involved largely with project delivery. Shane is an IBM Certified Advanced Technical Expert System p® 2006 and a Certified Systems Administrator Tivoli® Storage Manager v5. His areas of expertise include operating system provisioning, HACMP™, NIM, Virtualization and Security.

**Anirban Chatterjee** is an IT Specialist at the Executive Briefing Center in Austin, Texas. In this role he develops and delivers demonstrations of POWER and AIX features to external audiences (including potential customers, analysts, press, and investors) both within the Center and around the country, including demonstrations of workload partition technology in general, and WPAR relocation in particular. Prior to his involvement with the Center, Anirban developed firmware for IBM POWER-based servers.

**Henning Gammelmark** is a Systems Programmer for a leading financial services IT provider in Denmark. He has worked in the IT industry for the last 25 years. His experience initially focused on automation and software installations in a z/OS® environment. He then transitioned into his current role, where he provides technical leadership for AIX architecture, and installation and process management in large IBM System p environments. Henning originally started working with HACMP on IBM SP systems, then managed migration to POWER4™-based LPARs on P690. He later managed a similar migration to POWER5™-based and POWER6™-based systems. He participated in the IBM AIX Version 6.1 customer beta program, and is currently focused on implementation methods for using AIX workload partitions.

**Vijayasekhar Mekala** is a Software Engineer with the AIX workload partition (WPAR) functional verification testing team in the IBM India Software Lab. He has more than three years of experience in systems software development. Additionally, he has made significant contributions to IBM Intellectual Property, with 12 published patent disclosures and seven filed patent applications. He has authored technical papers focused on AIX and WPAR technology for IBM developerWorks®. Vijayasekhar is certified in System p administration, and holds a Bachelor of Technology degree in Computer Science from SK University, Ananthapur, India.

**Liviu Rosca** is a Senior IT Specialist working for IBM Global Technology Services, Romania. He has worked for IBM for six years providing support for pSeries®, AIX, and HACMP. His areas of expertise include pSeries, AIX, HACMP, networking, security and telecommunications. Liviu is an IBM Certified AIX 5L™ and HACMP System Administrator, as well as CCNP. He also teaches AIX and HACMP classes and has co-authored other IBM Redbooks.

**Arindom Sanyal** is Advisory IT Specialist and Solution Architect for the IBM Global Solutions Acceleration Initiative, based in IBM India. He works in the System Solution Center in Bangalore, where he focuses on pre-sales technical support and enablement for ISVs and systems integrators using IBM System p and AIX technologies. Arindom has more than 14 years of experience in Information Technology, focusing on architecture, implementation and system administration, using various server platform operating systems. He is an IBM Certified System p specialist, and a Red Hat Certified Engineer.

Production of this book publication was managed by:

**Chris Almond**, who is an ITSO Project Leader and IT Architect based at the ITSO Center in Austin, Texas. In his current role, he specializes in managing technical content development projects focused on Linux® and AIX 5L systems engineering. Chris has a total of 18 years of experience in the IT industry, including the last nine years with IBM.

## Acknowledgements

This book was developed with the generous support of many IBM employees around the world. For their contributions, guidance, patience, and technical feedback, we gratefully acknowledge the following people:

- Maria Diaz-Figueroa
- Thierry Fauck
- Eric Fried
- P I Ganesh
- Chetan L Gaonkar
- Nigel Griffiths
- Bahubali Jain
- Satish Nn Kharat
- Nagesh Nanjundachari
- Ajay Sami
- G Shantala
- David Sheffield
- Edward Shvartsman
- Marc Stephenson
- Philip Warren
- Sungjin Yook
- Kendell D Zahn

A special thank-you to the authors of the first book introducing workload partitions, *Introduction to Workload Partition Management in IBM AIX Version 6.1*, SG24-7431:

- Bruno Blanchard
- Pedro Coelho
- Mary Hazuka
- Jerry Petru
- Theeraphong Thitayanun

The team would also like to acknowledge the support for this project provided by:

- Jay Kruemcke, IBM AIX Offering Manager
- Scott Vetter, ITSO Project Leader

## 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** 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

# Part 1

# Introduction

Part 1 discusses the following workload partition topics:

- Introduction to AIX workload partitions
- Understanding and planning for WPARs

1

# Introduction to AIX workload partitions

AIX workload partitions (WPARs) provide a software-based virtualization solution for creating and managing multiple individual AIX operating system environments *within* a single AIX-based logical partition.

This chapter introduces the concept of workload partitions. We provide context for considering the use of workload partitions relative to other workload management and system partitioning options that IBM has provided for the System p platform running AIX in the past. We also introduce the terminology used to describe WPAR features, and describe system management scenarios in which you could benefit from using WPARs.

The following topics are discussed:

- Workload management and partitioning in AIX systems
- AIX6 workload partitions
- WPAR isolation and security
- Other WPAR features
- When to use workload partitions

## 1.1 Workload management and partitioning in AIX systems

Today's competitive corporate environment requires nimble IT departments with the ability to respond quickly to changes in capacity and usage. Use of innovative methods is necessary to maximize server utilization, control management costs, and reduce deployment time for new applications. Escalating costs of power and raised floor capacity also drive the need to utilize technology in new ways to maximize a company's IT investment.

For this reason, IBM has developed numerous tools to operate within its UNIX® server and operating system products, giving IT administrators new levels of control and flexibility in how they deploy and manage application workloads.

### 1.1.1 AIX Workload Manager

Workload Manager (WLM) was introduced as part of AIX with Version 4.3. It allows multiple workloads to run under one AIX instance. The system administrator builds rules based upon a user, process, or workload. Based upon these rules, shares of CPU and memory can be optimally assigned to the workload with peak demand (see Figure 1-1 on page 5).

If you have used WLM in the past, then refer to 8.6, "Using WPARs instead of WLM classes" on page 254, 8.7, "WPAR resource control changes to WLM" on page 255, and 8.8, "Frequently Asked Questions (FAQs) regarding WLM and WPAR resource control" on page 256, in Chapter 8, "Resource control" on page 241 to learn more about the relationship between Workload Manager and workload partitions.

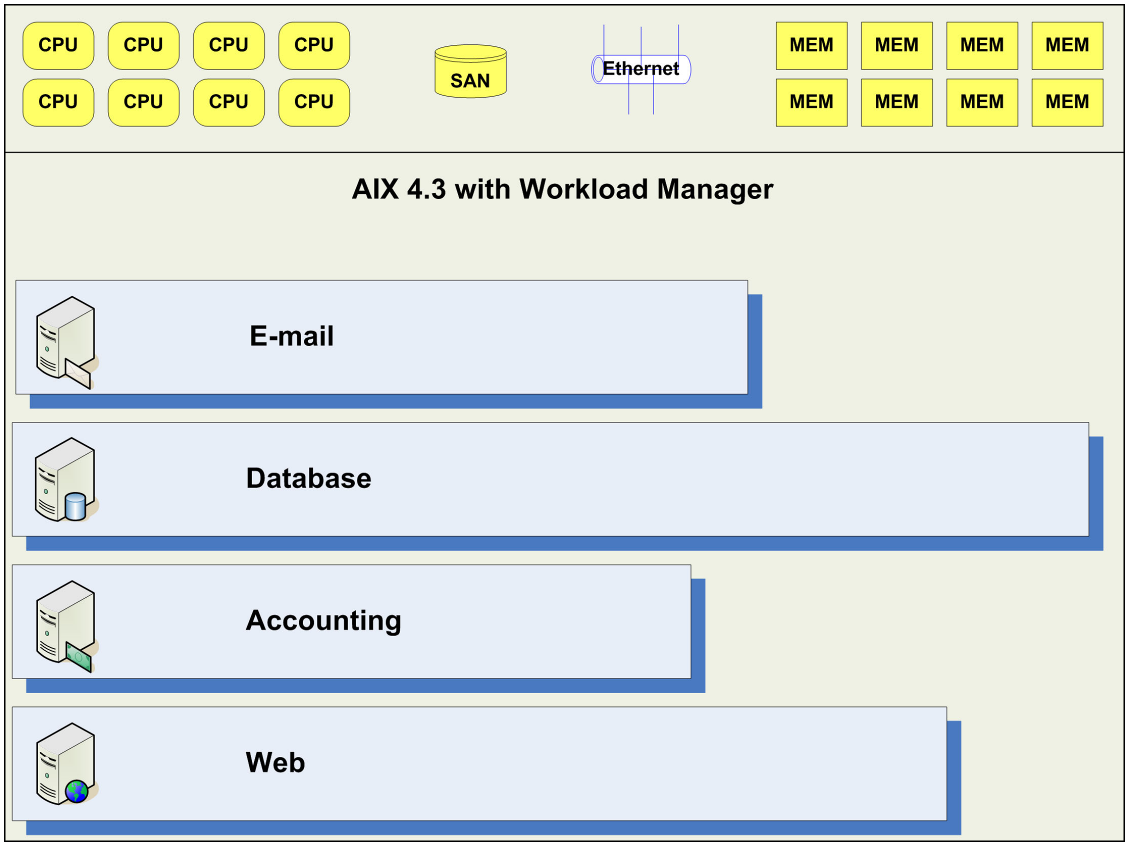


*Figure 1-1 WLM used to manage multiple workloads on a single AIX instance*

## 1.1.2 Logical partitions

With AIX 5.1 and POWER4 technology, IBM introduced logical partitions (LPARs) as a way to provide greater flexibility and better utilization of resources in large systems. With LPARs, systems could run AIX alongside other operating systems in separate partitions starting at a minimum of one CPU, 1 GB of memory, and one Ethernet adapter.

AIX 5.2 added more systems flexibility by being able to move CPU, I/O adapters, and memory dynamically without rebooting the LPARs. This allowed IT environments to become even more flexible in efficiently supporting workload hosting requirements (see Figure 1-2 on page 6).

**Note:** Throughout this book, the term LPAR is used to refer to all types of LPARS, such as a micropartition or dedicated partition of a POWER-based server, or a full physical server that is not partitioned (also known as a *full-system partition* in POWER4 terminology).

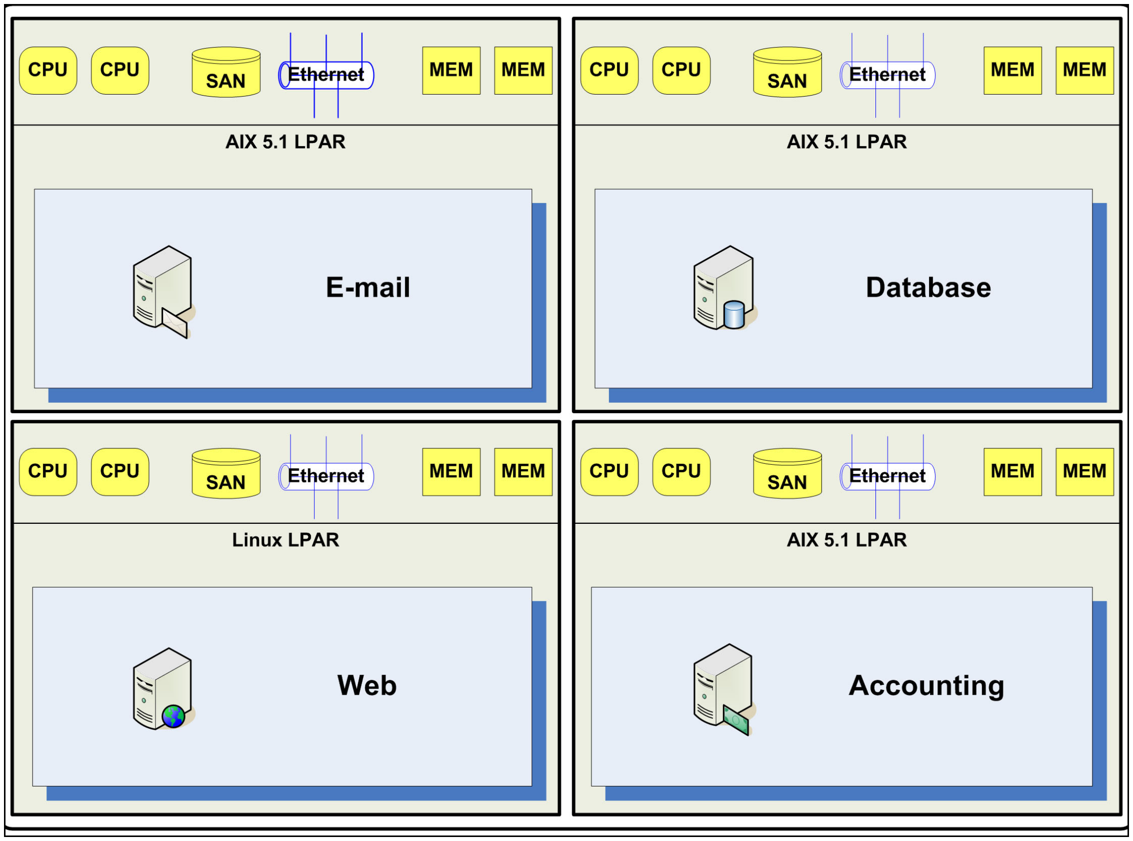


*Figure 1-2 System partitioned into four LPARs, each running a workload*

### 1.1.3 PowerVM (formerly Advanced POWER Virtualization)

The trend toward providing more system resource partitioning flexibility continued with the introduction of AIX 5.3 and the POWER5 processor. IBM System p Advanced POWER Virtualization (APV) offers advanced technology to facilitate server consolidation, reduce costs, provide redundancy, and adapt capacity to quickly meet demand. APV can reduce the need for static adapters, rapidly respond to changing capacity demands, and generally allow companies to utilize their purchasing dollars more effectively.

With the launch of the POWER6 platform, IBM rebranded APV as PowerVM™, and added key features such as the ability to migrate a running LPAR between systems (live partition mobility). See Figure 1-3.

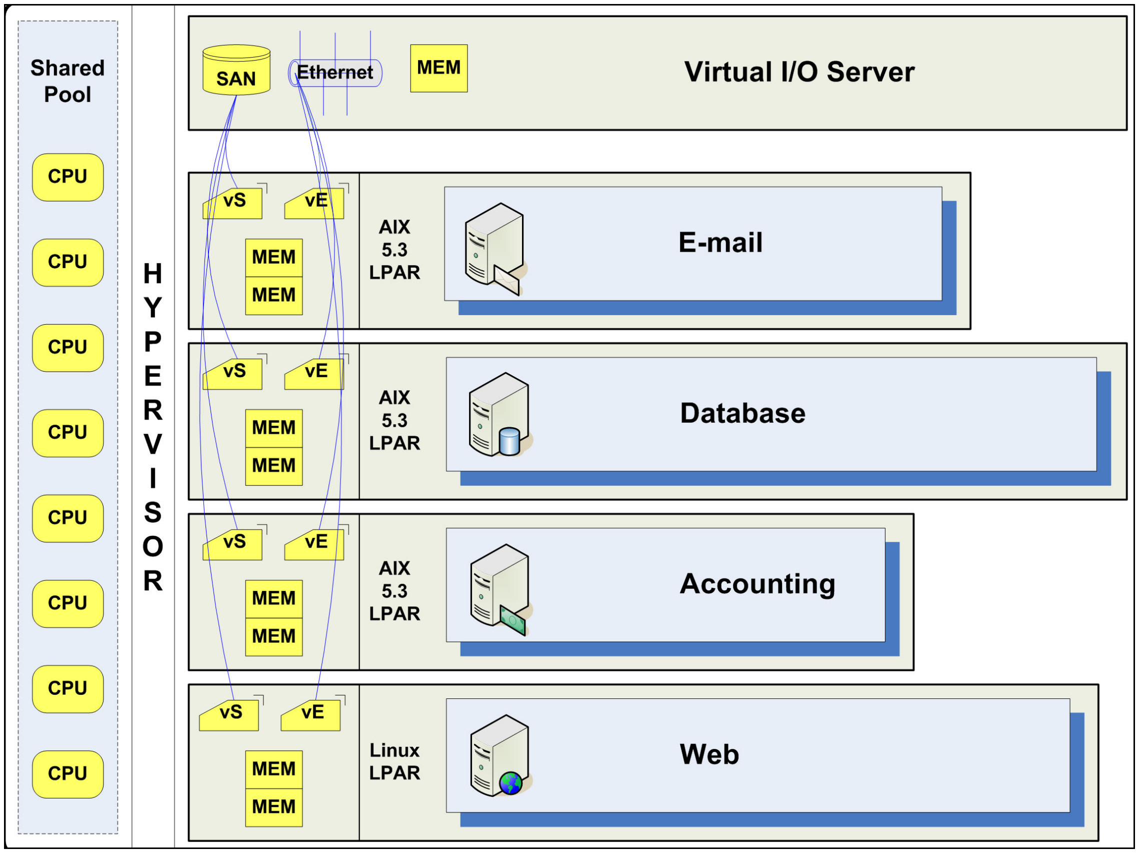


*Figure 1-3 Four LPARs dynamically sharing a pool of resources using VIOS and the PowerVM Hypervisor*

## 1.2 AIX6 Workload Partitions

In AIX 6, workload partitions (WPARs) add an additional operating system software-based layer for virtualization of operating environments. Each workload partition can host applications and isolate them from applications executing within other WPARs. This capability can be leveraged on any server platform capable of running AIX6, including POWER4, POWER5, POWER5+™, and

POWER6. Figure 1-4 shows three application- or service-specific WPARs being hosted within a single LPAR.

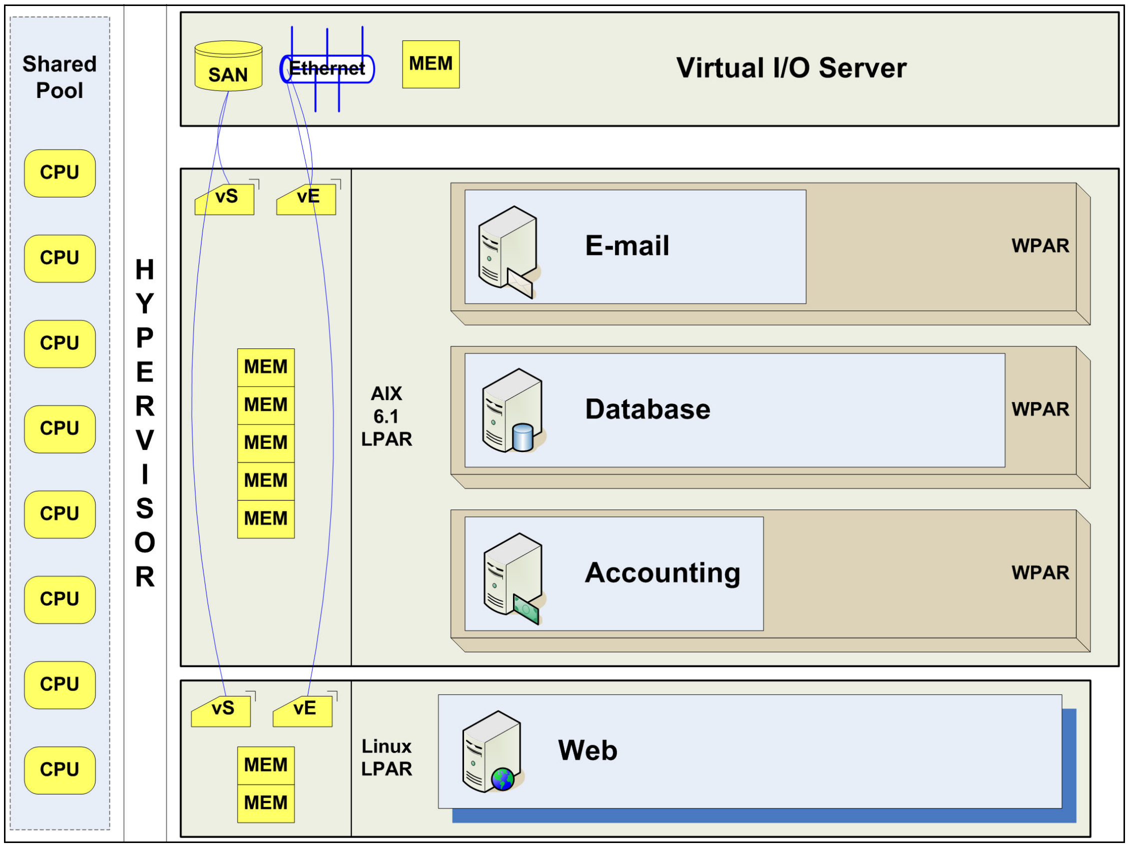


*Figure 1-4 WPARs reduce the number of managed LPARs - still provide workload isolation*

Workload partitions can be created within an AIX6 LPAR. Each workload partition provides an isolated environment for the application it hosts. From the application or service point of view, the WPAR provides a replica of a standard AIX operating system environment. Furthermore, the WPAR runtime environment can be dedicated to only hosting that application (the workload), and can be tuned to optimize performance based on the specific workload characteristics of that application. Logically, WPARs can be considered as an operating system level boundary around a specific set of AIX processes. Inside the WPAR, the applications have the following benefits:

- Private execution environments
- Isolation from other processes outside the WPAR
- Dedicated network addresses and filesystems

- Interprocess communication that is restricted to processes executing only in the same workload partition

The sections that follow introduce new concepts:

- Global environment

  This term refers to the part of the AIX operating system that hosts workload partitions. This is the classical AIX environment, and typically only the AIX6 system's root superuser will have access to it because it should be set up to host WPARs exclusively, not native applications.

- System WPAR

  This term refers to a more flexible WPAR-based instance of AIX. It contains dedicated writeable filesystems and system service daemons.

- Application WPAR

  This term refers to a WPAR that is set up to host only a single application or process. It provides an AIX runtime environment that is suitable for execution of one or more processes that can be started from a single command.

### 1.2.1 Global environment

As mentioned earlier, workload partitions are created within standard AIX V6 instances. The global environment is the part of an AIX V6 instance that does *not* belong to any workload partition. The global environment is therefore similar to the operating system environment of earlier versions of AIX. This global environment can be hosted within a dedicated LPAR or a micropartition.

The global environment owns all physical or virtual resources of the LPAR: network adapters, disk adapters, disks, processors, and memory. It allocates CPU and memory resources to the workload partitions, and provides them access to the network and storage devices.

The global environment has visibility into the workload partitions, and most performance monitoring and tuning activities are performed from this environment. A system administrator must be logged in to the global environment to create, activate, and manage workload partitions. Workload partitions cannot be created within other workload partitions. It is possible from the global environment to see (and control) the processes executing within the WPARs, and to see the file systems used by the WPARs. For this reason, it is recommended that no user accounts other than the system superuser have access to the global environment.

### 1.2.2 System WPAR

A system WPAR is similar to a typical AIX environment. Each system WPAR has dedicated writable file systems, although it can share the global environment /usr and /opt file systems in read only mode. When a system WPAR is started, an init process is created for it, which in turn spawns other processes and daemons. For example, a system WPAR contains an inetd daemon to allow complete networking capacity, making it possible to remotely log in to a system WPAR. It also runs a cron daemon, so that execution of processes can be scheduled.

### 1.2.3 Application WPAR

If an application or group of applications can be started with one command in the AIX command-line interface, it is a candidate to be hosted by an application WPAR. As soon as the command exits, the workload partition is also automatically terminated (or shut down). Using application WPARs is a quick way to leverage the isolation, resource control, and checkpoint features of workload partitions for hosting virtually any application or process.

Note the following points:

- An application WPAR shares the file system of the global environment. It does not own any dedicated storage.
- An application WPAR can run daemons, but it will not run any of the system service daemons such as inetd, cron, or srcmstr.
- It is not possible to remotely log in to an application partition or remotely execute an action into an application WPAR.

## 1.3 WPAR isolation and security

Even though workload partitions all run under the same operating system image, much care has been taken to ensure that applications running within WPARs are isolated from one another. In fact, the features provided with WPARs support levels of isolation that approach those that would be observed if the applications were run in separate LPARs.

These isolation features as they relate to processes, users, and resources are summarized in the following sections. For a comprehensive discussion of WPAR isolation and security, see Chapter 6, “Security in workload partition environments” on page 147.

## 1.3.1 Processes

Great effort has been taken to ensure that processes running in different WPARs cannot affect one another. To start with, a process running inside a workload partition can only see other processes in the WPAR; processes running in other WPARs or the global environment are invisible to it. Signals and other interprocess communications are only possible between processes within the same WPAR.

In addition, such processes can only access resources that are explicitly available inside the WPAR (filesystems mounted by the WPAR, network interfaces bound to the WPAR, and so on). All resources bound to a WPAR are tagged with the WPAR's ID so no other workload partition may access them.

## 1.3.2 Users

Application WPARs inherit their user profiles from the global environment, so they will have the same set of users, with the same privileges, that the global environment does.

System WPARs each maintain a totally independent set of users, complete with potentially unique or overlapping logins and security attributes. They do not inherit any users from the global environment. This is done to further the concept that system WPARs each behave as if they are a unique AIX instance.

## 1.3.3 Resources

In general, resources created or owned by the global environment can only be used by the global environment unless they are explicitly shared with a workload partition. Resources created or owned by a WPAR are visible only to that WPAR and the global environment.

To facilitate isolation of filesystems between system WPARs, a separate directory tree under the /wpars directory is created for each WPAR (for example, /wpars/wpar1, /wpars/wpar2). Inside this directory, each WPAR maintains its own home, tmp, and var directories. A system WPAR will also mount the global environment's /opt and /usr filesystems as read only. Application WPARs do not create their own filesystems, so they are usually allowed access to the filesystems owned by the global environment.

Each system WPAR can potentially be assigned its own network address, and applications running inside can only bind to the network address assigned to their WPAR. Communications between WPARs running under the same AIX instance are generally routed via the loopback interface by default. However, the administrator may optionally force traffic between selected WPARs to flow outside the system for network isolation reasons (for example, to monitor traffic levels for individual WPARs, to force all traffic through a firewall).

**Important:** Certain network isolation features are only available with AIX V6.1 TL2.

## 1.4 Other WPAR features

In addition to their isolation benefits, workload partitions provide other capabilities, such as checkpoint/restart and live application mobility. In order to enjoy these benefits, the system must have Workload Partition Manager installed.

### 1.4.1 Checkpoint/restart

Both types of workload partitions, the system WPAR and the application WPAR, have the ability to freeze all execution occurring inside and *checkpoint* the current execution state to a series of state files on disk. You can retrieve this execution state at some later time (perhaps even on a different system) and *restart* it.

The workload partition must be created with a specific option to enable checkpointing, and the checkpoint/restart operations can only be run from the global environment.

### 1.4.2 Live application mobility

Both types of workload partitions, the system WPAR and the application WPAR, are capable of being configured to support mobility, or *relocation*.

**Distinction:** IBM Power Systems™ and AIX V6 have two features that seem similar, but are different: WPAR live application mobility, and live partition mobility.

- WPAR *live application* mobility is a feature of AIX V6 and WPAR Manager. It is available on POWER4, POWER5, and POWER6 systems.
- Live *partition* mobility relies on the POWER6 hardware and hypervisor technology (PowerVM). It is available on POWER6 systems only.

  (This feature is also available to LPARs running AIX 5.3 and other operating systems running on System p.)

The capability to move a WPAR from one LPAR to another, possibly from one physical system to another, currently relies on common NFS-mounted filesystems between the two AIX images. Future support for SAN and GPFS™ filesystems is expected.

The application undergoes active relocation (*hot migration)* without stopping the application. Two modes are available for relocation: a mode that stops execution during the migration, and a different mode that allows for continuous execution of in memory processes during the migration.

**Important:**

- Workload partition mobility is a software solution that is dependent on AIX V6 for execution. When used for the migration of a WPAR from one LPAR to another or between physical systems, hardware *and* software compatibility is required.
- For detailed information that will help you understand and use WPAR mobility, see 4.1.2, “Deployment states and transitions” on page 65, 4.6, “Relocation” on page 88, 4.7, “CLI walkthrough” on page 92, 4.8, “Checkpointing and restarting” on page 97, and 5.4, “Preparing and creating mobile WPARs” on page 118.

The asynchronous mode for live application mobility is only available with AIX V6.1 TL2.

Application mobility is not a replacement for a high availability solution. The premise allows for planned migrations of workloads from one system to another so that the application is uninterrupted, and is intended for use during hardware maintenance, firmware installation, energy conservation, or other planned outages on the server. The workload does not need to be aware of the migration for the most part, but proper planning and testing are always recommended before moving anything into a production environment.

Figure 1-5 depicts the use of WPAR relocation for workload balancing, where two applications are moved between two servers to balance the load of these servers. Relocation of these applications can be done manually by the administrator, or on an automated basis by WPAR Manager using any of a number of resource utilization matrixes. WPAR Manager is described in more detail in Chapter 5, “Managing workload partitions” on page 99.

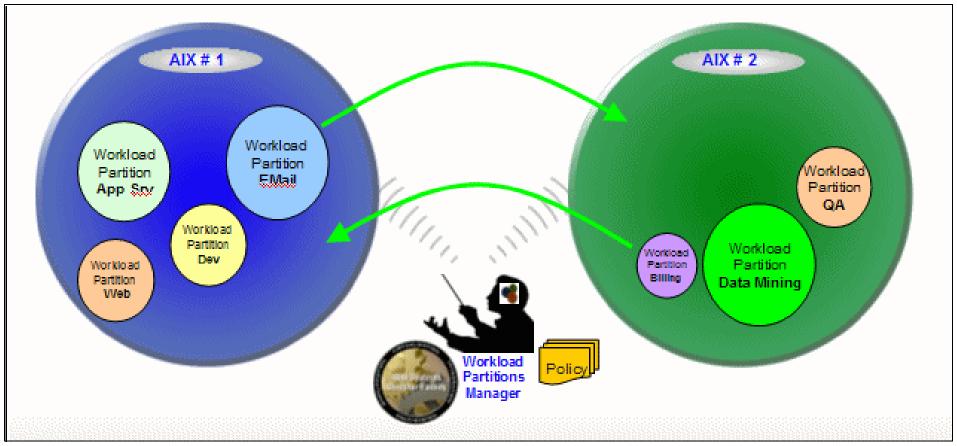

AIX # 1
Workload Partition App Srv
Workload Partition EMail
Workload Partition Dev
Workload Partition Web
AIX # 2
Workload Partition QA
Workload Partition Billing
Workload Partition Data Mining
Workload Partitions Manager
Policy

*Figure 1-5 WPAR migration*

## 1.5 When to use workload partitions

Workload partitions offer new possibilities for managing AIX environments. They complement other virtualization solutions available for System p6 platforms. The following scenarios show the benefit of using WPARs.

### 1.5.1 Improve application reliability and availability

Hardware components of an IT infrastructure might need to undergo maintenance operations requiring the component to be powered off. If an application is not part of a cluster of servers designed to provide continuous availability, then using WPARs to host them can help to reduce interruption of availability. Using the live application mobility feature, the applications that are executing on a physical server can be temporarily moved to another server without an application blackout period during the period of time required to perform the server physical maintenance operations.

Long-running jobs can take advantage of the checkpoint/restart feature of WPARs. It can be used to protect them against a failure or planned outage that requires restarting all computations from the beginning. The checkpoint feature can be used to regularly capture a snapshot of the application runtime environment, without having to instrument the code. In the case where the job needs to be stopped before reaching completion of the computation, the job can be resumed in the state it was when the last checkpoint was saved.

The checkpoint/restart feature can also be used to execute long-lasting batch jobs on a system with limited resources. This job can be run at nighttime, be paused during the daytime (when the computer resources have to be dedicated to other applications, such as transaction handling or Web serving), and then resumed at the beginning of the next night. In this case you need to be aware of any external application connections will be lost due to timeout.

The workload partition technology can also help in an environment where an application needs to be started often, on demand, and quickly. This might apply, for example, in test environments where resources are too scarce to keep multiple applications executing concurrently when not in use. Using WPARs, many applications can be defined on a server, but not activated. Activation of the workload partitions executing each of these applications can be performed only when needed for a test.

### 1.5.2 Simplify operating system and application management

WPAR technology can help system administrators simplify the way that they maintain operating systems and application software stacks.

For a long time, the traditional approach to application deployment has been to dedicate one server to one application. With the advent of virtualization and partitioning technologies, it has been possible to host multiple applications within partitions of a physical server. But this solution still implies that the system administrator needs to maintain one operating system instance for each application.

WPAR technology allows the system administrator to share an AIX instance between multiple applications, while still running each application within its own environment, providing operating system-level isolation between applications. In this case, the more applications that are consolidated within one AIX instance, the less the system administrator has to perform operating system fix applications, backups, migration, and other operating system maintenance tasks. Additionally, memory utilization is optimized because only one running operating system image needs to be resident in memory. However, note that this type of consolidation requires that all applications can run under the same version and maintenance level of the operating system.

In addition to sharing the operating system, the system administrator can take advantage of the WPAR technology to share application code. In a traditional AIX environment, if several Apache Web servers are needed, they each need to be deployed in a dedicated server or LPAR.

In a WPAR environment, it is possible to install Apache in one LPAR and then execute multiple instances of the Apache server within this LPAR, by starting multiple WPARs. Each WPAR runs its own Apache server with its own data in dedicated disk space, but shares the Apache code with all other WPARs. This type of configuration optimizes memory utilization by eliminating duplication of code. It also reduces administration maintenance of the Apache code, which only needs to be updated once for all server instances.

### 1.5.3 Optimize server utilization

The IBM Power Systems family offers many ways to optimize resource utilization through virtualization technologies, such as LPARs, DLPARs, and micropartitions. WPAR technology complements the existing solution offerings, due to its unique characteristics.

WPAR technology gives you additional flexibility in system capacity planning as part of a strategy for maximizing system utilization and provisioning efficiency. Due to the static allocation of partitions in physical servers, in a typical IT environment, each server is sized with spare capacity to allow for resource consumption increase of all applications executing within this server.

Using the mobility feature of WPARs, the server sizing and planning can be based on the overall resources of a group of servers, rather than being performed server by server. It is possible to allocate applications to one server up to 100% of its resources. When an application grows and requires resources that can no longer be provided by the server, the application can be moved to a different server with spare capacity.

The same mobility feature, combined with the policy-based relocation functions of the WPAR Manager, allows you to size a set of servers to handle the peak load, based on the overall resource capacity of the set of servers, and not for each server. In a classical environment, each server must be able to support the peak load of all partitions hosted within that server. With WPAR mobility, it is possible to take advantage of free resources in one physical server to offload another physical server hosting applications that require more resources than are locally available.

AIX V6 provides highly granular control of CPU and memory resource allocation to workload partitions (down to 0.01% increments). WPARs then are very suitable for consolidation of very small workloads. This can be particularly interesting for the replacement of old servers, for which even 10% of one POWER5 or POWER6 processor (the smallest micropartition) can easily handle the workload requirements of the application.

The *theoretical* upper limit on the number of workload partitions that can be executed within one LPAR is 8192. In actual practice, your application environment will probably require far less than 8192 WPARs running within a single LPAR. And in practice, we expect that you will encounter other AIX system limitations preventing you from actually approaching this theoretical limit.

**Note:** In practice, the number of WPARs that can be created and made active in an LPAR depends upon the capacity of the system, the configuration of the WPARs, and the characteristics of the applications being run in those WPARs.

### 1.5.4 Manage application resource utilization

When multiple applications are executing within the same AIX instance, the system administrator might want to control the amount of CPU and memory resources used by each application. One way to perform this control is to set up the Workload Manager (WLM) functions that are part of the standard AIX feature set.

WPAR technology provides a new way to perform this resource control. The WPAR resource control uses the WLM technology, but it in a way that is invisible to the system administrator. There is no need for the system administrator to interact with WLM directly. The resource control is available through options of the WPAR command line and SMIT interfaces.

The WPAR resource control feature allows the system administrator to actively assign resources between applications competing for CPU and memory resources. This guarantees that each application receives a share of the CPU and memory resource available from the global environment. These resources are separate from the requirements of the other applications executing in WPARs within the same operating system instance. It is possible to modify resource allocations even after the WPAR has been defined and started.

### 1.5.5 Scoping administrative privileges by application

In large AIX environments, where a partition hosts many applications, it is not unusual to have multiple people acting as system administrators. However, all of them might not need root or superuser privileges in all domains of system administration. These people can be specialized for activities, such as user administration, network control, storage control, or software maintenance.

WPAR technology supports this specialization of roles, and can help to restrict the privileges given to one person to simply the scope that person needs to control. Each system workload partition has its own user set, independent from the user set defined at the global environment level. An individual who is using root within a system workload partition only has superuser privileges for the resources visible within this WPAR. This user cannot control global environment resources (such as network adapter or physical devices), and cannot act on resources belonging to other workload partitions.

Many applications need the application administrator to use the root user to control the application, even if this person does not need to manage the operating system. WPAR technology allows you to delegate superuser privileges to one individual and limit them to an application environment without jeopardizing the global environment. Refer to 6.5, “RBAC in WPAR environments” on page 177 for more details about how to use RBAC to accomplish this.

The separation of user sets (or security domains) between different system workload partitions also enables system administrators to isolate groups of users logging on in AIX environments according to their application access control requirements. Users defined in one system WPAR are unaware of the applications executing in the global environment or in other WPARs. They cannot see the list of users or processes outside their WPAR.

IBM AIX Version 6.1 provides improvement over the previous AIX 5L Version 5.3 for role-based control of user privileges. This feature is known as Role-Based Access Control (RBAC). For a detailed description of these new features, refer to *AIX V6 Advanced Security Features Introduction and Configuration*, SG24-7430.

WPAR integrates the use of RBAC features for controlling privileges. A default RBAC setting is provided with each WPAR, but the system administrator can also further customize the RBAC configuration used in a WPAR context.

### 1.5.6 Protect the existing hardware investment

Although customers using POWER4 IBM pSeries servers cannot take advantage of physical or hypervisor-based virtualization technology, WPAR technology relies only on IBM AIX Version 6.1 with no dependency on the underlying hardware. It can be used on POWER4, POWER5, and POWER6-based servers.

Clients having many applications, each running on a dedicated POWER-based server or dedicated partition and requiring only a fraction of the available processing power, can consolidate these applications within one LPAR using WPARs. Each application can be executed within one WPAR, providing a dedicated environment isolated from the other applications’ environments, while all WPARs share the physical resource of one LPAR.

### 1.5.7 Easily clone application environments

With workload partitions, it is simple to quickly provision application environments for development, test, or production use.

Prior to AIX6, when an administrator is asked to provision a new server, they would normally need to create a new LPAR, install AIX into it via a boot image, install any fix packs or environment customizations, and finally install any needed applications before the server could be made available. WPAR technology allows the administrator to quickly provision a workload partition for immediate use within minutes. The newly provisioned WPAR would inherit the latest fixpacks, customizations, and applications installed in the global environment by the administrator.

Workload partition configuration information can be stored in human-readable specification files. These specification files can be generated by the operating system from preexisting workload partitions and can be edited, created, or modified manually. In an environment where a system administrator has to manage several application environments, the WPAR technology can help the system administrator quickly clone and define new application environments. These specification files can be used as input to WPAR creation commands, allowing the system administrator to automate, through scripts and programs, the startup and handling of multiple workload partitions. These techniques also facilitate rapid recovery from situations where system users have destabilized their environments beyond the practical point of repair. If the WPAR has a recent checkpoint available, you should be able to reload the checkpoint and resume work with little impact. Otherwise, an identical working environment can quickly be created based on the specification files of the original WPAR.

### 1.5.8 Support “green” computing strategies

Using WPAR relocation features for live application mobility means that you have the flexibility to consolidate workloads during periods of low usage onto smaller numbers of operating server platforms. In this strategy, you still provide continuous application availability, but you do so using a smaller number of powered up servers. As you approach normal high usage periods, you can then power up additional peak demand server resources and relocate cyclical workloads back to those machines during those peak demand periods. For example, if your data center peak workload periods are 12 hours per day, 5 days per week, peak load systems only need to be powered up approximately 35% of the time.

2

# Understanding and planning for workload partitions

This chapter describes the fundamental technical details of workload partition (WPAR) technology. This knowledge is required to help you understand and plan for the implementation of workload partitions into your environment.

The information in this chapter will be useful to solution designers, systems architects and systems administrators who need to understand, plan and implement WPARs in their IT environment.

The following topics are discussed:

- High-level planning information
- General considerations
- Considerations for the global environment
- Application WPARs
- System WPARs
- Mobility
- WPAR and LPAR comparison

## 2.1 High-level planning information

Workload partitioning is a software-based virtualization feature of the AIX 6 Operating System and part of the IBM PowerVM virtualization offering. Therefore, it can be used on any hardware platform that supports the AIX 6 Operating System including:

- IBM p Series POWER4
- IBM Systems p POWER5
- IBM Power Systems p6
- IBM BladeCenter® JS21 PowerPC® 970 processors
- IBM BladeCenter JS22 64-bit POWER6 processors

There are two components (one mandatory and one optional) that make up the WPAR offering:

- *Mandatory* IBM AIX 6.1 contains the base support and tools for the WPAR technology. It is required for WPAR technology to be deployed and implemented.
- *Optional* IBM Workload Partition Manager provides additional features such as a graphical user interface to manage WPARs and WPAR groups including mobility operations. The Workload Partition Manager tool is required to take advantage of all of the capabilities of AIX WPAR technology.

## 2.2 General considerations

Workload partitioning provides for isolation of software services, applications, and administration by utilizing software limitations and boundaries set within a single AIX 6 operating system instance known as the *global environment*. A WPAR in its most simplistic form can be easily created, configured, and started from the command line interface or the smitty interface in a matter of minutes.

Note that this technology presents additional considerations around the planning and configuration of network, file systems, and storage. You must also give care and consideration to the ongoing administration of both the global environment and the WPARs it hosts.

### 2.2.1 Software prerequisites

Having a single AIX operating system image simplifies the installation and general administration of the WPARs and the global environment that hosts them. Software is installed once and used many times in across all WPARs within the global environment. Although totally isolated from each other, these WPARs use the same AIX kernel instance. Therefore, all kernel extensions should be loaded from the global environment. This topic is discussed in greater detail in Chapter 6, “Security in workload partition environments” on page 147.

This means that all WPARs use the exact same level of AIX. When planning for WPARs, you must ensure that all application software products are fully supported for the level of AIX running in the global environment. It is even more important that you plan *meticulously* for future updates and upgrades to the AIX operating system instance in the global environment.

The best practice for global environments is to keep the operating system installs as generic as possible, minimizing the use of unique tuning settings from one AIX host to the next. This will greatly reduce the chance of any unexpected results during mobility operations. Updating or upgrading the AIX operating system image in the global environment means updating or upgrading the inherited AIX images in all hosted WPAR environments. This means special consideration is required for any interdependencies between the application software running in the WPARs and the version of AIX running in the global environment.

If you have an application that falls outside the support matrix for a specific version of AIX and for extenuating reasons it cannot be upgraded, then it should be moved into another LPAR so that its dependencies do not prevent the global environment’s AIX operating system, and therefore, the other remaining WPARs, from being upgraded.

## 2.2.2 File system considerations

When a system WPAR is created with the default options, then it shares (in read-only mode) the /usr and /opt file systems of the global environment. This expedites the creation, installation, and updating of WPARs and also prevents the accidental removal of system software that is shared with other WPARs.

However, having read-only shared /usr and /opt file systems might not suit every application. Certain applications are designed to write into the /usr or /opt file systems. One solution is to define the needed application’s writable directory as a different file system and link it to the mount point that the application requires.

Refer to 7.4, “Advanced file system considerations” on page 213 for an explanation of how a WPAR can have a writable directory under a read-only /usr or /opt.

Another solution is for the application to not use the global environment shared /usr or /opt file systems. This option requires additional disk space, because it duplicates the global environment's /usr or /opt to the WPAR's private and fully writable file systems.

Consolidating several applications within one global environment changes the way the system administrator manages the file systems. The systems administrator now manages an increased number file systems in the one LPAR, as opposed to having the same or a similar number of file systems dispersed across several LPARs.

A *system WPAR* has four dedicated file systems and two shared (read-only) file systems, as well as access to the /proc file system. For example, deploying 100 system WPARs in one global environment will result by default in a global environment with 400 separate file systems and 600 mount points specified in the /proc pseudo-file systems.

Contained within WPAR technology is an option to reduce this number. Instead of using the default creation options, the system administrator can choose to create one single file system per WPAR as described in detail in Chapter 7, "Advanced configuration features" on page 203.

This solution creates only one real file system (the root / file system) for the WPAR. The remaining /var, /tmp, and /home are then simply created as subdirectories of the / file system, instead of individual file systems usually created in AIX instances and as is the default when creating a system WPAR.

File systems of each system WPAR are created in the global environment directory tree and are mounted under the WPAR *base directory*. One base directory is defined per WPAR. The default path of the base directory is /wpars/<*wpar-name*>. When planning to deploy several system partitions, the system administrator might want to consider organizing the base directory in a different manner.

Refer to 2.4, "Application WPARs" on page 28 and 2.6, "Mobility" on page 31 for more detailed information about the file system considerations for application WPARs, system WPARs, and considerations surrounding mobility.

### 2.2.3 Network considerations

When planning and considering network options in support of WPAR deployment, you need to understand how to exploit this technology to its full potential. For instance, using aliases decreases the number of adapters needed for communications but requires careful planning and consideration of bandwidth utilization, because several WPARs can share the same adapter.

The use of the Workload Partition Manager tool requires that access through the network firewalls, for the different components such as LPARs and workstations, be configured to allow traffic on specific ports as listed in Figure 2-1.

**Important:** Figure 2-1 details the default port allocations required for communications between WPAR Manager and WPAR Manager agents.These can be modified when configuring WPAR Manager.

Note the following points:

- Ports 9510, 9511, 9512, and 9513 are used for communication among agents and managers.
- Ports 14080 and 14443 are used for communication between the system administrator’s workstation and WPAR Manager.

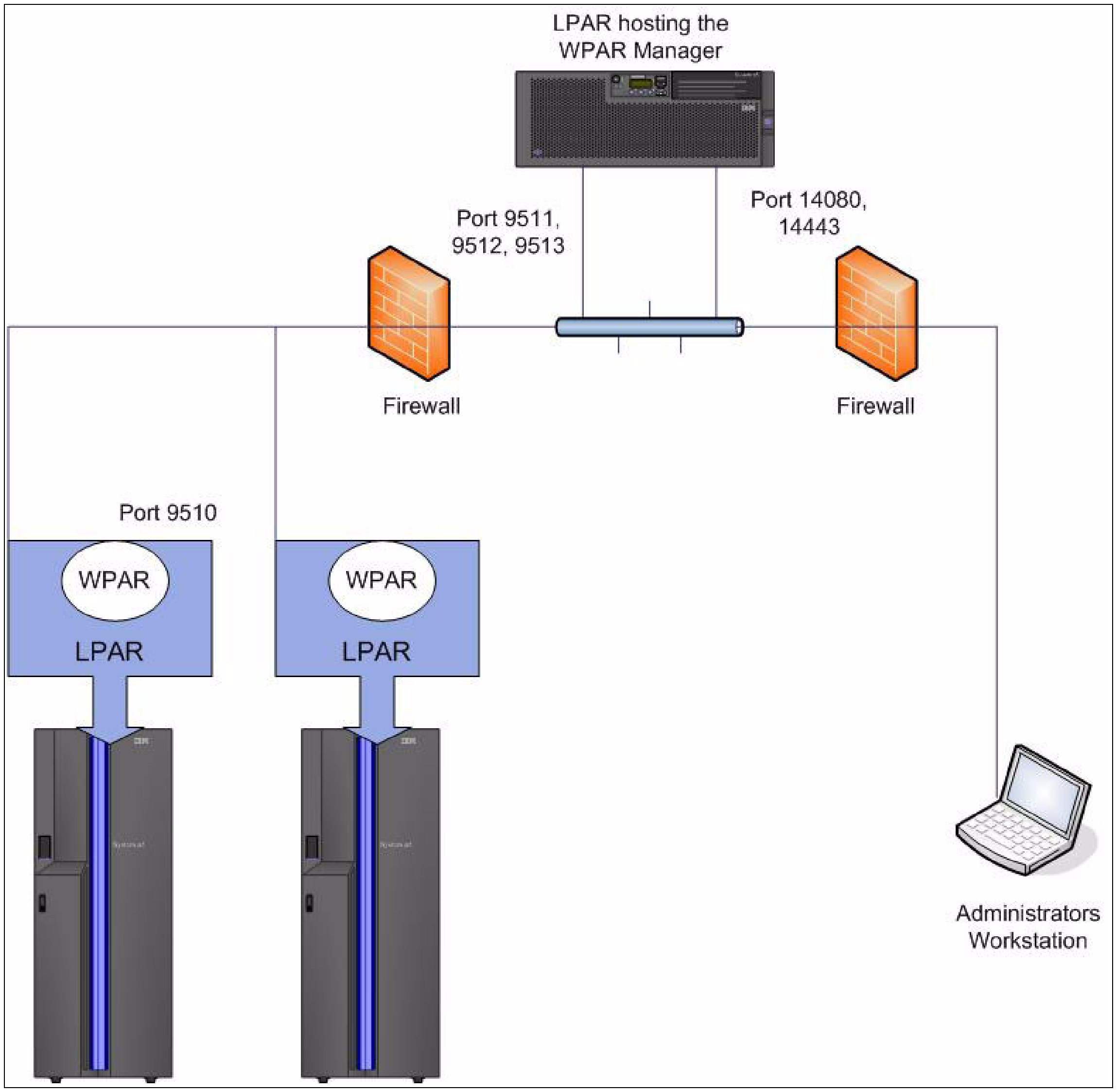


*Figure 2-1 Communication between WPAR Manager and WPAR Manager agents*

NFS is an essential requirement for the WPAR mobility feature. Three components are necessary in order to provide the NFS communications for WPAR mobility:

- The hostname and IP address of the global environment
- The hostname and IP address of the WPAR
- The hostname and IP address of the NFS server

Because they all function in this communication, they all must know each other. It is preferable to have all three components located within the same subnet. The importance of this requirement is explained in greater detail in 10.6, “Network File System client mobility support” on page 319.

## 2.3 Considerations for the global environment

There are several considerations to take into account when you are planning a system running WPARs. The global AIX instance can contain one or a multitude of workload partitions. It is possible to mix application and system WPARs within the same global environment.

If WPAR Manager is used, the global environment also contains a *WPAR Manager agent*.

The global environment, as with any classical AIX instance, has one or more dedicated networks, IP addresses, and disks, along with unique users and groups.

The global environment can use physical or virtual adapters. The hosted WPARs have no control of, nor can they directly access, the hardware devices. The global environment therefore also owns all physical I/O adapters needed by the workload partitions. Note the following points:

- A sufficient number of I/O adapters must be configured on the global environment to support the combined I/O throughput of all hosted WPARs.
- The global environment must have access to all disks that will contain the file systems used by each hosted WPAR.
- If WPARs need to have IP connectivity, they will have an IP address that needs to be configured as an alias on one of the physical network adapters.

It is theoretically possible to create up to 8192 WPARs within the one AIX operating system image. However, the higher the number of WPARs within the global environment, the more crucial it is to provide careful planning and consideration of system resources, particularly devices.

For optimal utilization and consolidation of the system and its resources, a combination of virtualization from the PowerVM package Virtual I/O Server (VIOS) is recommended with WPAR systems. This allows for rapid deployment and redeployment of system resources.

## 2.4 Application WPARs

An *application WPAR* can be viewed as a shell, launched from the global environment, that is spawned by a command and contains an application. This is a lightweight application resource and as such, does not provide remote login capabilities for users, nor does it provide for more than limited customization after the application WPAR is created and running.

It only contains a small number of processes, which are all related to the application, and uses the services and file systems of the global environment. After the application's last process in the application WPAR exits, the WPAR ceases to exist and is removed from the global environment.

Figure 2-2 illustrates the file system relationship between the application WPAR and the global environment.

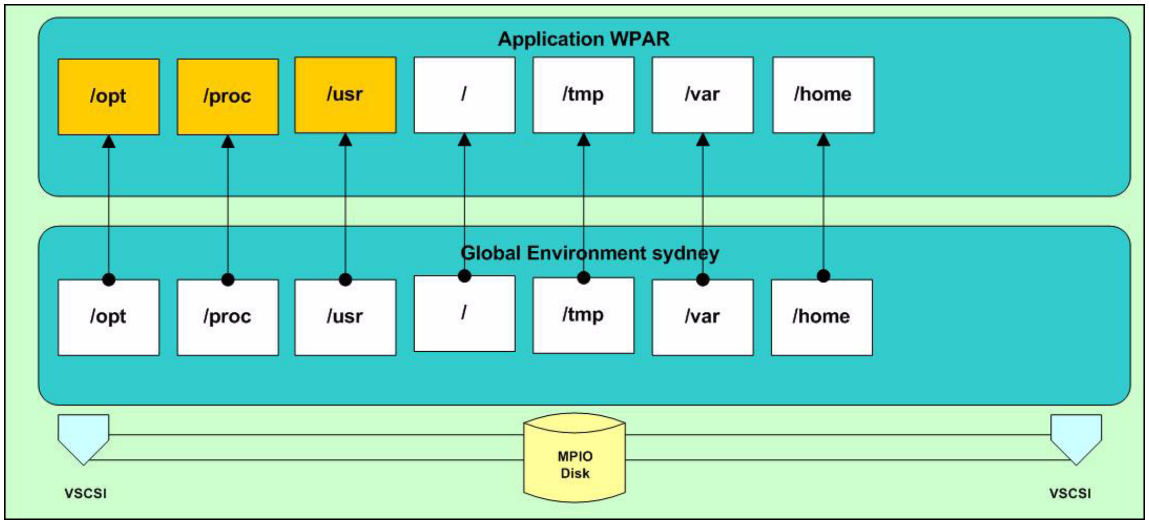


*Figure 2-2 Application WPAR file systems*

## 2.5 System WPARs

When a system WPAR is created, it generates its own file systems and, by default, shares /usr and /opt with the global environment in read-only mode. It has separate users and groups. Users can log into it like any AIX operating system running in a conventional LPAR environment; see Figure 2-3 on page 31.

Example 2-1 shows the /wpars directory created in the global environment that will contain the system WPARs file systems.

*Example 2-1 Files generated in the global environment*

```
root@sydney:/# ls -ald /wpars
drwx------    3 root     system          256 Jun 19 13:30 /wpars
root@sydney:/#
```

Within the /wpars directory in Example 2-2, notice the `skippy` directory. This directory will function as the default mount point for that WPAR's non-shared file systems.

*Example 2-2 Listing the wpars*

```
root@sydney:/wpars# ls -al
drwx------    3 root     system          256 Jun 19 13:30 .
drwxr-xr-x   36 root     system         4096 Jun 19 13:30 ..
drwxr-xr-x   18 root     system         4096 Jun 19 13:32 skippy
drwxr-xr-x   18 root     system         4096 Jun 19 13:32 kenny
```

Example 2-3 lists the WPAR's file systems generated by the creation of the `skippy` system WPAR and made available when the WPAR is started.

*Example 2-3 WPAR file systems created*

```
root@sydney:/wpars# ls -al /wpars/skippy
drwxr-xr-x   18 root     system         4096 Jun 19 13:32 .
drwx------    3 root     system          256 Jun 19 13:30 ..
drwxr-xr-x    3 root     system          256 Jun 06 09:31 admin
drwxr-x---    2 root     audit           256 Jun 06 09:31 audit
lrwxrwxrwx    1 bin      bin               8 Jun 19 13:30 bin -> /usr/bin
drwxrwxr-x    5 root     system         4096 Jun 19 13:32 dev
drwxr-xr-x   28 root     system         8192 Jun 19 13:33 etc
drwxr-xr-x    4 bin      bin             256 Jun 19 13:30 home
lrwxrwxrwx    1 bin      bin               8 Jun 19 13:30 lib -> /usr/lib
drwx------    2 root     system          256 Jun 19 13:30 lost+found
drwxr-xr-x  135 bin      bin           12288 Jun 19 13:32 lpp
drwxr-xr-x    2 bin      bin             256 Jun 06 09:31 mnt
drwxr-xr-x   17 root     system         4096 Jun 18 19:40 opt
dr-xr-xr-x    1 root     system            0 Jun 19 13:45 proc
drwxr-xr-x    3 bin      bin             256 Jun 06 09:31 sbin
drwxrwxr-x    2 root     system          256 Jun 19 13:31 tftpboot
drwxrwxrwt    3 bin      bin            4096 Jun 19 13:33 tmp
lrwxrwxrwx    1 bin      bin               5 Jun 19 13:30 u -> /home
lrwxrwxrwx    1 root     system           21 Jun 19 13:32 unix ->
/usr/lib/boot/unix_64
drwxr-xr-x   42 bin      bin            4096 Jun 19 13:30 usr
drwxr-xr-x   26 bin      bin            4096 Jun 19 13:32 var
drwxr-xr-x    2 root     system          256 Jun 19 13:30 wpars
```

Example 2-4 displays the output of the **df** command executed from the root directory of the `skippy` WPAR. It shows that one system WPAR is hosted within the `sydney` LPAR, with its own independent file systems mounted under the /wpars/skippy base directory.

The example shows that the /, /home/ /tmp, and /var file systems of the system WPAR are created on logical volumes of the global environments. It also shows that the /opt and /usr file systems of the WPAR are namefs mounts over the global environments /opt and /usr file systems.

*Example 2-4   WPAR file systems: df output*

```
root@sydney:/wpars/skippy# df
Filesystem    512-blocks      Free %Used    Iused %Iused Mounted on
/dev/hd4         2621440   1581848   40%    18502    10% /
/dev/hd2         6815744   2764920   60%    39136    12% /usr
/dev/hd9var      1310720   1172208   11%     2617     2% /var
/dev/hd3         1572864   1563408    1%       86     1% /tmp
/dev/hd1          262144    261416    1%        5     1% /home
/dev/hd11admin      262144    261416    1%        5     1% /admin
/proc                  -         -    -         -     -  /proc
/dev/hd10opt      262144     50784   81%     1961    25% /opt
/dev/livedump     524288    523552    1%        4     1% /var/adm/ras/livedump
/dev/fslv00       262144    211536   20%     1623     7% /wpars/skippy
/dev/fslv01       262144    257320    2%        5     1% /wpars/skippy/home
/opt              262144     50784   81%     1961    25% /wpars/skippy/opt
/proc                  -         -    -         -     -  /wpars/skippy/proc
/dev/fslv02       262144    257248    2%        8     1% /wpars/skippy/tmp
/usr             6815744   2764920   60%    39136    12% /wpars/skippy/usr
/dev/fslv03       262144    233272   12%      372     2% /wpars/skippy/var
```

Figure 2-3 on page 31 depicts the default file system layout for a system WPAR and the relationship between the WPAR and the global environment at a file system level.

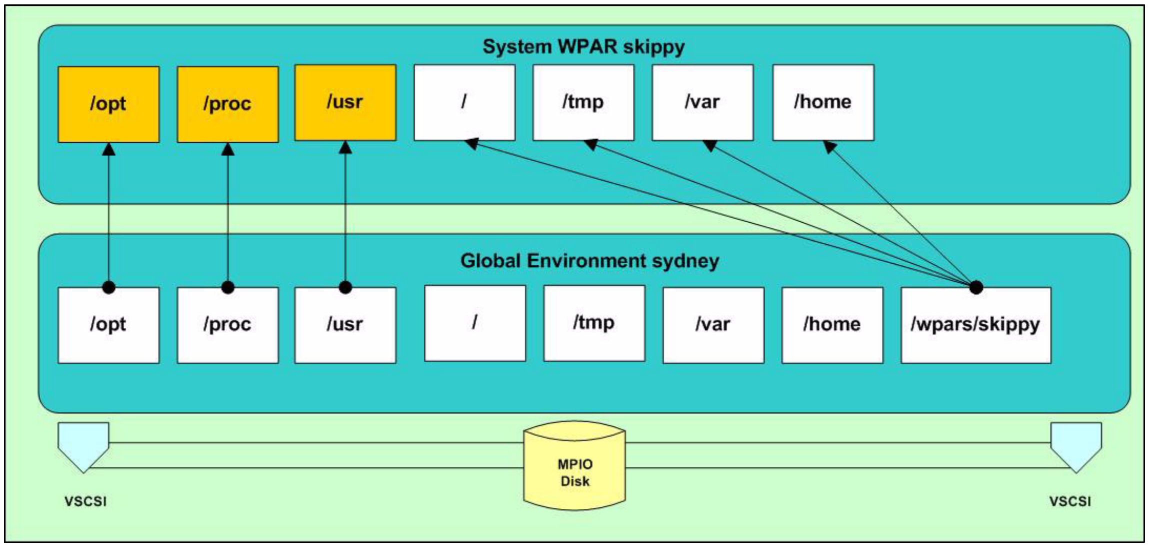


*Figure 2-3 Default file systems relationship between a system WPAR and the global environment*

## 2.6 Mobility

PowerVM technology makes it possible to configure both application and system WPARs to be relocated from one global environment to another global environment. However, mobility is dependent on using the IBM Workload Partitions Manager for AIX (WPAR Manager) offering that is purchased separately.

**Differentiation:** As previously mentioned, there are two distinct features surrounding AIX V6.1 and IBM System p that, on the surface seem quite similar: WPAR mobility and live partition mobility. Note the following explanations:

- WPAR mobility is a key feature of AIX V6.1 and the WPAR Manager. It is available on IBM POWER systems that utilize POWER4, POWER5, and POWER6 processor technology.
- Live partition mobility requires POWER6 hardware and is reliant on the hypervisor technology (PowerVM) to provide its functionality. It is available on POWER6 systems only. This feature is *also* available to AIX 5.3 LPARs.

Live Application Mobility allows for planned migrations of workloads from one system to another without interrupting the application. You can use this technology to perform a planned firmware installation on the server. Most workloads do not need to be aware of the WPAR relocation. However, we always recommend proper planning and thorough testing before moving anything into or around a production environment.

*WPAR mobility* applies to both application and system WPARs. The relocation of a WPAR involves moving its executable code from one LPAR to another LPAR, while keeping the application data on the same storage devices. It is necklaces, therefore, that these storage devices are visible and accessible from both the source and target LPARs hosting the WPAR.

In AIX V6.1, NFS provides this dual access to the storage area. The global environment hides the physical and logical device implementations from the hosted WPARs. The WPARs work with data storage at the file system level. All files that need to be written by the application must be hosted on an NFS file system, which serves the file systems required of the WPARs to the global environments that could host the WPARs. All other files, including the AIX operating system binaries, can be stored in file systems local to the hosting global environment.

Table 2-1 helps you plan the creation of the file systems for an application that requires WPAR mobility, when hosted in either an application or system workload partition, for an application that only writes in file systems dedicated to the application. The table details the high level difference and requirement.

*Table 2-1 Default file system location to enable mobility*

| File system | Application WPAR | System WPAR |
|---|---|---|
| / | Global environment | NFS mounted |
| /usr | Global environment | Global environment |
| /opt | Global environment | Global environment |
| /var | Global environment | NFS mounted |
| /tmp | Global environment | NFS mounted |
| /home | Global environment | NFS mounted |
| Application file systems | NFS mounted | NFS mounted |

Figure 2-4 on page 34 illustrates a complete environment in which there are both LPARs and WPARs on two p595 systems:

- The first global environment is called sydney and is hosted on an LPAR on the first p595. It is a client of the NFS server (which resides on the

Management server; in this instance, a p550), as well as the system WPAR "roy", currently running inside the global environment on sydney.

- The second system is also a p595, but it could be any of the system p range of systems from the p505 up. One of its LPARs hosts a global environment called canberra, which is also a client of the NFS server.

The Management server in this example is a p550. This system contains an NFS server, a NIM server, and a WPAR Manager for AIX to provide the single management point needed for all the WPARs.

The NIM server is needed to store, update, and distribute images and other operating systems patches and updates. It is also needed to provide the capability to install a large number of LPARs concurrently, if required.

The NFS server provides the mechanism to allow you to make the file systems required by the WPAR available to any LPAR that the WPAR may be relocated to, thus allowing them dynamic movement from one system to another without disrupting the application.

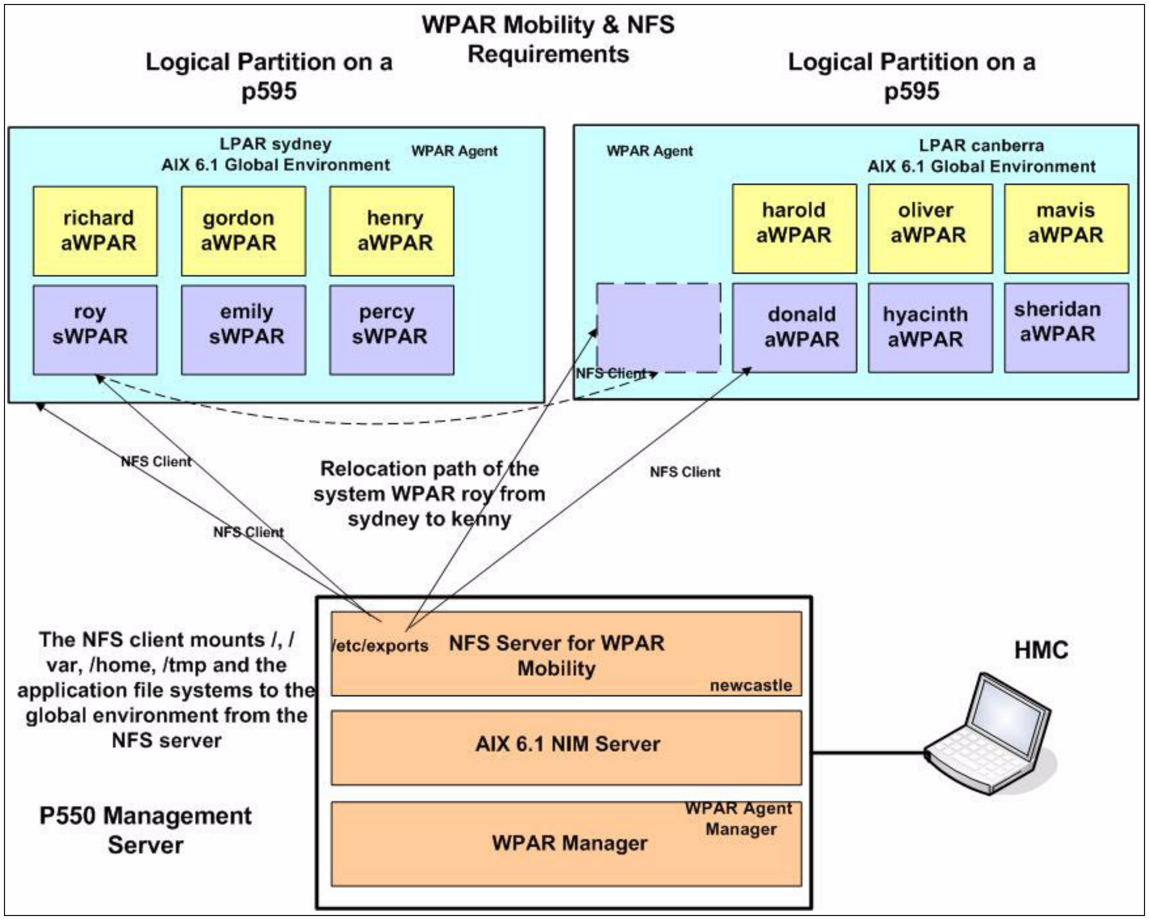

WPAR Mobility & NFS Requirements
Logical Partition on a p595
Logical Partition on a p595
LPAR sydney AIX 6.1 Global Environment
WPAR Agent
richard aWPAR
gordon aWPAR
henry aWPAR
roy sWPAR
emily sWPAR
percy sWPAR
WPAR Agent
LPAR canberra AIX 6.1 Global Environment
harold aWPAR
oliver aWPAR
mavis aWPAR
donald aWPAR
hyacinth aWPAR
sheridan aWPAR
NFS Client
NFS Client
NFS Client
NFS Client
Relocation path of the system WPAR roy from sydney to kenny
The NFS client mounts /, /var, /home, /tmp and the application file systems to the global environment from the NFS server
/etc/exports
NFS Server for WPAR Mobility
newcastle
AIX 6.1 NIM Server
WPAR Agent Manager
WPAR Manager
P550 Management Server
HMC

*Figure 2-4 Environment configured for WPAR mobility*

Figure 2-4 illustrates the file system relationships with NFS incorporated for mobility. The NFS server has a standard configuration and utilizes either NFS Version 3 or Version 4. You can use command line editing or SMIT to configure the exports represents the relationship among the different views of the same file systems as seen.

Figure 2-4 provides an environmental view of the systems configured to provide for WPAR mobility:

- From the NFS server where they are physically located
- From the global environments on which they are NFS-mounted
- From the system WPAR that uses them

In the WPAR, the /opt, /proc, and /usr are set up as namefs with read-only permissions in the global environment, with the exception of /proc that is always read-write. The remainder of the file systems (/, /home, /tmp, and /var) are set up as standard NFS.

It is imperative that the /etc/exports file on the NFS server have permissions set for both the global environment (sydney) and system WPAR (newcastle) for mobility to work.

**Note:** The NFS server must provide access to *both* the global environment and the WPAR for the mobility feature to be utilized. In this instance, access must be provided to the WPAR and *all* global environments to which the WPAR might be relocated.

Figure 2-5 outlines the file system layout for the NFS exports required to allow WPAR mobility for WPAR skippy.

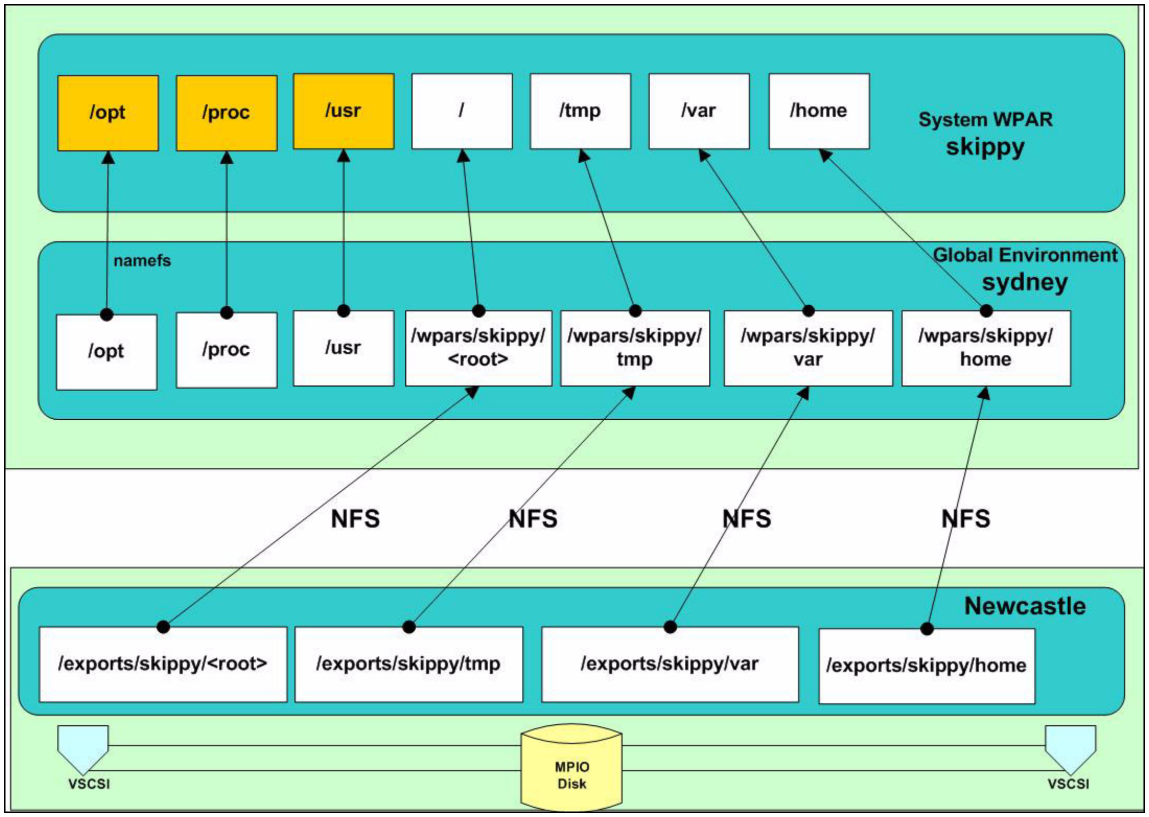


*Figure 2-5 File systems exported from the NFS server (Newcastle) providing for mobility of WPAR skippy*

## 2.7 WPAR and LPAR comparison

Each logical partition (LPAR), regardless of size, runs a complete instance of the AIX Operating System. By contrast, a WPAR can move from the global environment in one AIX instance to the global environment in another AIX instance on a different machine and, at most, run a *condensed* version of AIX.

You can also have, in theory, up to 8192 WPARs in a single AIX instance. Therefore, thorough planning, particularly of the physical adapters and network configuration, is essential.

It is incorrect to think of WPARs as a replacement for LPARs. These technologies are both key components of the IBM PowerVM virtualization strategy and should be seen as complementary components of the same offering. The technologies are best exploited when combined to enhance and extend their individual value.

Combining and utilizing both LPAR and WPAR technology offers a broad range of virtualization options to meet the different and or changing needs of individual IT environments. Table 2-2 compares and contrasts the differences between the two technologies.

*Table 2-2 Comparing WPAR and LPAR*

| **Workload partitions** | **Logical partitions** |
|---|---|
| Shared or minimized operating system image | Complete, individual operating system image |
| Finer granularity in resource management per WPAR | Resource management per LPAR and Capacity on Demand |
| Security isolation at software level provided by the AIX 6 Operating System | Security isolation at hardware level provided by the Hypervisor |
| Operating system tuneable parameters inherited and governed by the global environment | Operating system tuneable for application and system performance |
| Lower overall cost:<br>► Single operating system image to manage<br>► Simple to configure, create, and remove<br>► Integrated management tools | Operating system fault isolation and problem determination |

**Important:** Keep the following guidelines in mind when evaluating the information in Table 2-2 on page 36:

- An LPAR should be seen as the foundation of your virtualization strategy and direction. An LPAR will allow for greater flexibility in supporting the environment and, after being finalized, the LPAR strategy. You can then look to complement that strategy with a WPAR design that will further enhance the overall environment.
- An LPAR also provides a more definitive form of operating system isolation.

Part 2

# Managing workload partitions

This part discusses the following WPAR operations and management topics:

- Functional overview of workload partitions
- Overview of WPAR operations
- Managing WPARs

3

# Functional overview of workload partitions

This chapter contains a functional overview of the WPAR offering as part of the IBM PowerVM family of virtualization technologies available for IBM System p. It introduces the tools and commands used to manage WPARs. It explains the types of WPARs and their functions, and highlights their implementation, administration, and management.

The following topics are discussed:

- Understanding application WPARs and system WPARs
- WPAR tools overview
- WPAR Manager
- AIX command line interface
- WPAR commands
- Modified AIX commands
- WPAR description database

## 3.1 Understanding application WPARs and system WPARs

This section explains the two distinct types of WPAR.

### Application WPAR

The application WPAR shares the file systems of the global environment. It can be set up to receive its application file system resources from disks that reside on the hosting AIX instance or from an NFS server.

If an application WPAR is to utilize file systems mounted via NFS, then the NFS must be mounted in the global environment. The mount point will be the same whether viewed from within the WPAR or viewed from the global environment. The system administrator of the NFS server must configure the /etc/exports file so that file systems are exported to both the global environment and to the application WPAR, and also to any other global environment that the WPAR may be relocated to.

Processes executing within an application WPAR do so in complete isolation from other WPARs. They are only aware of processes that are executing within the same WPAR. Therefore, the use of Inter-process Communication (IPC) by application software is limited to the set of processes isolated within the boundary of the WPAR.

Application WPARs are temporary objects. The complete lifecycle of an application WPAR is the duration of the hosted application's runtime. The application WPAR is created when the application process is initiated and begins its execution. The application WPAR is removed when the last process running within the application partition exits.

An application WPAR is a candidate for mobility. It can be started quickly in one global environment and relocated to another global environment.

### System WPAR

The system WPAR, when created, generates a smaller version of a typical AIX environment for the exclusive purpose of executing applications in an isolated environment that has certain restrictions.

A system WPAR has its own runtime resources. It contains an init process that can spawn daemons. For example, it has its own inetd daemon to provide networking services and its own System Resource Control (SRC).

Every system WPAR has its own unique set of users, groups, and network interface addresses. The users and groups defined within a system WPAR are completely independent from the users and groups defined within the global environment. In particular, the root user of the WPAR only has superuser privileges *within* that particular WPAR, and has no visibility into, or privilege in, the global environment. By contrast, the root user in the global environment has superuser access over all WPARs running within the global environment.

For example, the application or database administrator can be given root privilege within the WPAR without having any privileges in the global environment. The environment provided by a system WPAR to its hosted application is a complete AIX environment with access to all AIX system files that are available in a native AIX environment. The creation of a system WPAR includes the creation of a base directory (see Chapter 5, “Managing workload partitions” on page 99, for more information about this topic).

This base directory is the root of the WPAR environment. By default, the path to this base directory is /wpars/<wparname> in the global environment.

By default, the base directory contains seven file systems:

- The /, /home, /tmp, and /var file systems are real file systems that are dedicated to the system partition’s use.
- The /opt and /usr file systems are read-only namefs mounts over the global environment’s /usr and /opt.
- The /proc pseudo-file system maps to the global environment /proc pseudo-file system (/proc in a WPAR only makes available process information for that WPAR).

## 3.1.1 Comparing application WPARs and system WPARs

Before considering, designing, installing or administering WPARs, it is essential to understand the differences between application WPARs and system WPARs. These differences are listed in Table 3-1 on page 44.

Table 3-1 Workload partition comparison

| **Application WPAR** | **System WPAR** |
|---|---|
| **Used to isolate an individual application** | **Used to create a virtual operating system environment to host multiple applications** |
| - Provides a runtime environment for executing either a single application or a group of related applications.<br>- Does *not* contain running AIX services or daemons normally present in a complete instance of AIX. | - Provides a complete running virtualized instance of AIX, including system init and other AIX services and daemons. |
| - An application or group of related applications are started from the command line or via a script. | - Similar to a conventional AIX environment; applications can be started in a variety of ways (that is, CLI, command from crontab or inittab) |
| - Very lightweight, requiring only one process in addition to the application's processes. | - Consists a large number of processes, similar to a traditional AIX environment running within an LPAR. |
| - Can be created and started in seconds in the same action. | - Created and started in two distinct steps.<br>- Takes longer as it transitions between states. |
| **Transient** | **Persistent** |
| - Created as soon as the application is started, and removed when the final application process exits. | - WPAR can be started, stopped, and restarted, similar to any AIX environment.<br>- WPAR can be running even if the applications it hosts are dormant or quiescent. |
| - If **`startwpar`** fails, WPAR is deleted. | - If **`startwpar`** fails, WPAR remains in the Defined State. |
| - **`lswpar`** will only list active application WPARs. | - **`lswpar`** will list all system WPARs and the state they are in currently. |
| **Global resources** | **Dedicated resources** |
| - Users and groups, including those that are application-specific, are defined in the global environment. | - Separate users and groups that are completely independent of the global environment. |
| - Shares the name space of all required file systems. | - Has its own writable file systems and shares in addition to sharing others with the global environment and other WPARs |

| Application WPAR | System WPAR |
|---|---|
| - Has a vinit process with a PID of 1; sole function is fulfill the role of the parent process to all other processes. | - Has its own init process which spawns other AIX processes and daemons. |
| - Limited user access | - Complete user access |
| - No Remote login access via `ssh`, `telnet` or `clogin`, and so on. | - Supports remote login: `ssh`, `telnet`, `rsh`, and `rlogin`, depending on daemons and services running within the workload partitions. |
| - Does not provide for RBAC within WPARs. | - RBAC can provide granular privileges and security controls within the context of the WPAR. |

## 3.2 WPAR tools overview

The workload partitioning functionality and base tools are packaged as part of the AIX6.1 base operating system. Table 3-2 details which tools are packaged in the operating system and which tools or enhanced features are bundled as part of IBM Workload Partitions Manager v1.2 for AIX offering, an additional separately purchased product.

*Table 3-2 WPAR management tools and features*

| Tool/Function | Packaged within | Usage or purpose |
|---|---|---|
| AIX Command Line Interface | AIX 6.1 Operating System | Create, activate, modify and delete WPARs. |
| SMIT/smiTTY | AIX 6.1 Operating System | Fast path guided usage of the CLI tool. |
| WLM | AIX 6.1 Operating System | WLM is the underlying technology for WPAR Management. Administrators do *not* interact with WLM directly. |
| WPAR Manager | IBM Workload Partitions Manager for AIX | A GUI interface used for all WPAR operations, including mobility and automating mobility. |

With the exception of the WPAR Manager for AIX product, all tools listed in Table 3-2 on page 45 are packaged with AIX 6.1 and can only be utilized to manage and support the WPARs that are running native to that global environment. These tools cannot be used to manage WPARs running remotely in other global environments; this can only be achieved with the licensing and use of the WPAR Manager software.

## 3.3 WPAR Manager

The IBM Workload Partitions Manager for AIX (WPAR Manager) product provides for the management of multiple WPARs executing on a number of LPARs, which can be running on different physical machines. The WPAR built-in features contained in AIX 6.1 are provided by the fileset bos.wpars.

The WPAR Manager additional program consists of the filesets:

| | |
|---|---|
| **mcr.rte** | This is the support for WPAR mobility. |
| **wparmgt.agent.rte** | This is the WPAR agent executing in all LPARs containing managed WPARs. |
| **wparmgt.mgr.rte** | This is the WPAR Manager executing in the management LPAR. |
| **wparmgt.cas.agent** | This is the Common Access Service agent running in all LPARs containing managed WPARs. |
| **wparmgt.cas.agentmgr** | This the Common Access Service agent running in the management LPAR. |
| **Tivoli.tivguid** | This is the graphical user interface (GUI). |
| **lwi.rte** | This is the Eclipse-based LightWeight Infrastructure (LWI) runtime. |

The WPAR Agent Console is a browser-based tool and therefore has no associated fileset. It can be started on any workstation that can connect via the network to the WPAR Manager.

Figure 3-1 on page 47 illustrates the relationship of WPAR Manager, WPAR Agent Manager, LPARs, WPARs and the global environment. It also shows a simplified view of relocating WPARs from one global environment to another.

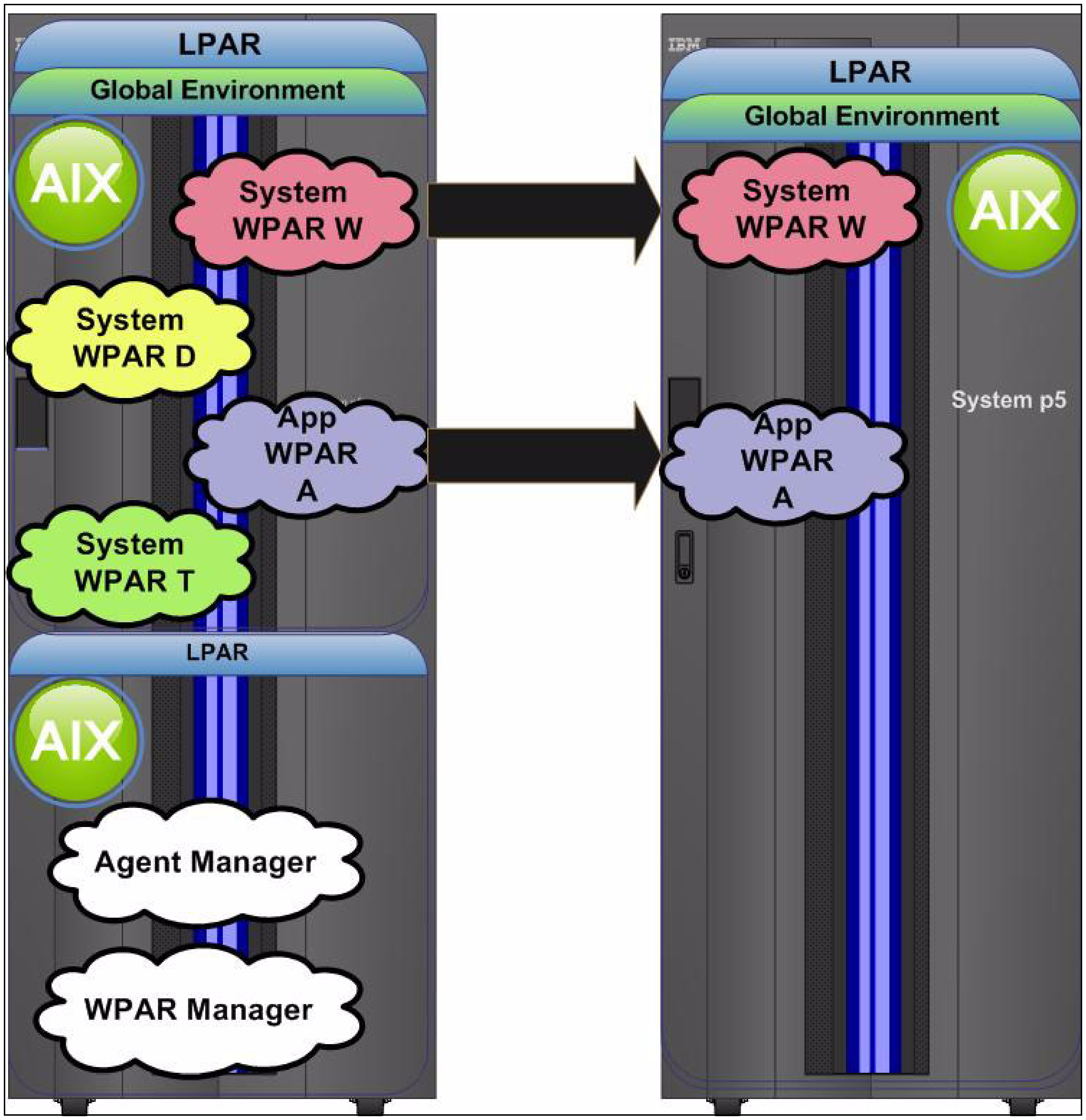


*Figure 3-1 WPAR relationships and mobility example*

## AIX command line interface

To support WPARs, the AIX command line interface (CLI) has been extended:

- New commands have been added.
- Certain existing commands have been modified.

Some of these commands can be used from the global environment hosting the WPARs and some within WPARs themselves. Some commands apply to system WPARs. Other commands apply to application WPARs.

Table 3-3 lists all of the new commands of the AIX CLI that are specific to WPAR management. It also describes the environment in which they can be used. Commands are grouped by the type of system administration activity.

### SMIT

For each of the commands listed in Table 3-3, a SMIT panel exists that provides the same functionality. SMIT will display the command and details of the flags it provides (based on selections and options made) using the F6 key prior to execution of the command. The starting point to access the SMIT panels depends on the type of partition to manage: smitty **`manage_syswpar`** or smitty **`manage_appwpar`**.

In addition, commands that existed in AIX prior to the availability of workload partitions have been enhanced to support this new feature. As a general rule, the at (@) character is used as the new flag of existing commands for requesting WPAR-specific information. For example, vmstat -@ ALL reports a summary of virtual memory information for all workload partitions. These commands are covered in more detail in 3.5, “Modified AIX commands” on page 50.

There are several SMIT panels that will guide and assist you through the creation, customization, management, and administration of WPARs.

## 3.4 WPAR commands

*Table 3-3 WPAR commands*

| | **Applies to** | **Executed from** |
|---|---|---|
| **WPAR instance management** | | |
| lswpar | Application WPARs and system WPARs | Global environment |
| mkwpar | System WPARs only | Global environment |
| startwpar | System WPARs only | Global environment |
| stopwpar | Application WPARs and system WPARs | Global environment |

|  | **Applies to** | **Executed from** |
|---|---|---|
| wparexec | Application WPARs only | Global environment |
| chwpar | Application WPARs and system WPARs | Global environment |
| rebootwpar | System WPARs only | Global environment |
| rmwpar | Application WPARs and system WPARs | Global environment |
| rcwpars | System WPARs only | Global environment |
| **Backup and Restore** |  |  |
| savewpar | System WPARs only | Global environment |
| restwpar | WPAR backup media | Global environment |
| restwparfiles | WPAR backup media | Global environment |
| mkwpardata | WPAR backup media | Global environment |
| lssavewpar | WPAR backup media | Global environment |
| **Software management** | **Only applies to system WPARs** |  |
| syncwpar | System WPARs only | Global environment |
| syncroot | System WPARs only | Within system WPAR only |
| **Partition access** |  |  |
| ssh | System WPARs only | Anywhere in the network with connectivity to the WPAR |
| clogin | System WPARs only | Global environment |
| rsh | System WPARs only | Anywhere in the network with connectivity to the WPAR |
| telnet | System WPARs only | Anywhere in the network with connectivity to the WPAR |

## 3.5 Modified AIX commands

Commands that existed in AIX prior to the introduction of the workload partitioning feature have been modified and enhanced to support WPARs. A general principle is that the command is given an @ character flag to request an action on a WPAR. For example, **`iostat -@ ALL`** will report the I/O for the global environment as well as all WPARs running in the global environment.

These modified commands and their behavior in the global environment and within a WPAR are listed in Table 3-3 on page 48.

*Table 3-4 Modified AIX commands*

| Command | Flags or arguments | Behavior in WPAR | Behavior in the global environment |
|---|---|---|---|
| acctcom | "- @ wparname" | Fails with `workload partition not found` message unless workload partition name is Global. In this case, fails with `cannot open“/var/adm/pacct` message | Executes normally, displaying accounting records for workload partition "wparname". |
| acctctl | "- @ no argument" | Fails with `cannot open /var/adm/pacct` message | Executes normally, displaying accounting records for all workload partitions.<br>A workload partition name is displayed for each record. |
| clogin | All options | Fails with a `not owner` message. | Executes normally if user has correct privilege. |
| df | All flags | Displays only the WPAR mounted file systems. | Displays information on all mounted filesystems with absolute paths. |
| domainname | No arguments | Displays the domain name for the WPAR. | Displays the domain name of the global environment. |
| hostid | No arguments | Displays the hostid of the WPAR. | Displays the hostid of the system. |
| hostname | No arguments | Displays the hostname of the WPAR. | Displays the hostname of the system. |
| ifconfig | All display options (**`-a -l`**) | Displays information about the WPAR. | Displays information about the global environment. |

| Command | Flags or arguments | Behavior in WPAR | Behavior in the global environment |
|---|---|---|---|
| ioo | | Not functional in a WPAR. | no change in behavior. |
| ipcrm | No argument | Removes IPCs associated with the WPAR, | Removes IPCs from the global environment. |
| ipcrm | "-@ wparname" | Invalid unless workload partition name = Global. In this case, removes IPC object within the WPAR, | Removes IPCs from the wparname WPAR. |
| ipcs | No argument | Displays information about IPC objects in the WPAR, | Displays IPC information pertaining to the global environment. |
| ipcs | "-@" | Displays the IPC information for within the WPAR, | Displays information on all IPCs in the system and, for WPAR associated PICs, the name of the WPAR. |
| lparstat | `-w` | Displays information about the workload partition. | Displays information about the workload partition.<br>Run from the global environment the WPAR key value is 0. |
| mkclass | All options | Will only update /etc/wlm directory.<br>Will fail updating the kernel data. | No change. |
| mount | No arguments | Displays the WPAR mounted file systems. | Displays all mounted file systems. |
| mount | With arguments | Only Network File System (NFS) mounts without cachefs allowed.<br>Forced nosuid, nodev. | No change. |
| netstat | All options except `-a` and `-i` | Fails - cannot open /dev/kmem. | Displays information on all sockets and interfaces on the system. |
| nfso | | Not functional in WPAR. | No change. |
| no | all options except `-a` | Fails with an error message. The `-a` option executes normally. | No change. |

| Command | Flags or arguments | Behavior in WPAR | Behavior in the global environment |
|---|---|---|---|
| projctl | All options except qproj(s) | qproj executes normally, else it fails with a `not owner`: error | No change. |
| ps | `-e` | Displays process information within the WPAR. | Displays all process information in the system unless the -@ wparname is specified. |
| ps | `-@ wparname` | Displays process information for process within the WPAR. | Displays process associated with the wparname WPAR. |
| schedo | | Not functional in WPAR. | No change. |
| uname | `-n` | Displays the WPAR hostname. | Displays the system's nodename. |
| uname | `-W` | Displays the static WPAR ID number. | No change. |
| vmo | | Not functional in WPAR. | No change. |
| wlmstat | `-@` | Not functional in WPAR. | Will display the data for meta class in the WPAR. |
| wlmstat | All options | Not functional in WPAR. | No change. |
| wlmtune | All options | Not functional in WPAR. | No change. |
| wlmcntrl | All options | Not functional in WPAR. | No change. |

## 3.6 WPAR description database

When a workload partition is created, a configuration profile for that partition is stored in the workload partition database. This section explains how the AIX system uses the WPAR description database to store information about workload partitions.

WPARs, like most UNIX concepts, are described in *files*. A system administrator needs to be familiar with the three groups of WPAR description files that exist in an AIX instance.

- The definition of the partition is used by the operating system. It is stored in the WPAR database. The system administrator normally accesses the content of this database by using operating system commands, such as `lswpar`.

- The specification of a WPAR can be used by a system administrator to create a partition. Specification files are described in 5.4, “Preparing and creating mobile WPARs” on page 118.
- The definition of the defined partition is stored in the WPAR database. This database consists of files stored in the /etc/wpars directory, as well as an extra stanza added to the /etc/filesystems of the global environment.

**Important:** The implementation of the WPAR database is subject to change in future versions of AIX. This section is intended to provide the system administrator with an understanding of the pieces of information used to describe WPARs. However, we expect system administrators will never need to interact directly with the information contained in the files described here.

A system WPAR that has been defined but is not active or running is represented in:

- An index file (/etc/wpars/index)
- A configuration file (/etc/wpars/wparname.cf)
- Stanzas in /etc/filesystems

The index file contains information regarding the name, the persistent and configured identifiers of all WPARs known to the system, and all defined system partitions, as well as all executing application partitions. It also contains the WPAR type (system or application).

The configuration file and /etc/filesystems contain the information required to start a WPAR. For each system, the configuration file contains the full privilege and device sets for the WPAR and all information provided by the system administrator when creating the WPAR, except the file system-related information. The global environment /etc/filesystem contains stanzas describing how file systems of the global environment are mapped to file systems of the WPAR. Note that /etc/filesystems is only used for information related to system WPARs. Application WPARs do *not* need specific file system information in order to execute.

Because the WPAR database consists of files stored in the global environments rootvg, it is saved during any backup of the global partition (for example, during a mksysb operation) and can be restored just as any other set of files. But if the data and file systems for the WPARs do not match the configuration information contained in the WPAR database, then results can be unexpected.

The best way to recover a WPAR is to use the **`restwpar`** commands to restore a savewpar backup image. The best way to save the definition of an existing WPAR is to create a specification file from the content of the WPAR database.

WPAR backup and restore considerations are further described in Chapter 5, “Managing workload partitions” on page 99. Backup, restore, and cloning require careful planning and special consideration both prior to installation and for ongoing support and maintenance of the environment.

4

# Overview of workload partition operations

This chapter provides an overview of basic WPAR management, including creating and managing simple WPARs. It also covers the considerations you need to keep in mind when using WPARs.

The following topics are discussed:

- WPAR administration
- File system setup
- Software installation
- Backup, restore, and cloning
- Users and groups
- Relocation
- CLI walkthrough
- Checkpoint and restart

## 4.1 WPAR administration

In this section we describe basic methods for creating and administrating WPARs.

In our case, we used the command line interface (CLI). However, you can also use a smit window, as described here:

- To use the main WPAR menu, enter **`smit wpar`**.
- To use the application WPAR menu, enter **`smit manage_appwpar`**.
- To use the system WPAR menu, enter **`smit manage_syswpar`**.

For convenience in the following sections, we do not show all output from all commands. Instead, we simply display the significant output.

### 4.1.1 Creating and managing simple WPARs

Workload partitions are divided into system WPARS and application WPARs. The commands used to create each type of WPAR are different:

- For system WPARs, use **`mkwpar`**, which effectively creates a new virtual AIX.
- For application WPARs, execute the application within the WPAR by issuing **`wparexec`**.

By default, if you do not provide a hostname at the time of the creation of the WPAR, the hostname is the same as the WPAR’s name or the basename of the command.

#### Creating a simple application WPAR

Creating an application WPAR is straightforward, because the only mandatory parameter is the full path of the executable file to run inside the WPAR. As an illustration, this section demonstrates how to create a sample application WPAR called myapp.

The myapp application is a simple script looping for 120 seconds, as shown in Example 4-1. In the real world it would more likely be a script starting WebSphere® Application Server or another, similar workload.

*Example 4-1 Our sample myapp script*

```
#!/bin/ksh
a=0
max=120
sleeptime=1
```

```
   echo "Starting  myapp" > /mnt/wpr02/myapp.log
   while [ "$a" -ne "$max" ]
   do
     let a=a+1
     echo "Killroy was here, for the $a time and he said 42"  >>
   /mnt/wpr02/myapp.log
     sleep $sleeptime
   done
```

You start an instance of an application WPAR by telling **wparexec** to run myapp. Example 4-2 shows the **wparexec** command starting myapp immediately after creation.

*Example 4-2 Creating simple application WPAR myapp*

```
root@lpar00:/hga# wparexec /hga/myapp
Starting workload partition myapp.
Mounting all workload partition file systems.
Loading workload partition.
.
.
Shutting down all workload partition processes.
```

This type of WPAR only exists while the application is running. When the application ends, the WPAR also ends and all of its resources are freed.

If the application WPAR has a dependency on a file system that is not mounted, it will mount the file system automatically, as illustrated in Example 4-3. Note that the WPAR displays first in the Transitional (T) state. Then it changes to the Active (A) state. Finally, it disappears when myapp completes.

*Example 4-3 Application WPAR myapp lifecycle*

```
root@lpar00:/# lswpar
Name    State  Type  Hostname  Directory
---------------------------------------------
myapp   T      A     myapp     /

root@lpar00:/# lswpar
Name    State  Type  Hostname  Directory
---------------------------------------------
myapp   A      A     myapp     /

root@lpar00:/# lswpar
#
```

## Creating a simple system WPAR

Creating a system workload partition is also straightforward. AIX provides a default value for each available option of the **mkwpar** command. To create a new WPAR, you only supply the name of the partition:

```
# mkwpar -n wpar001
```

You will normally want to extend the command syntax to include an IP address for the WPAR:

```
# mkwpar -n wpar001 -N address=9.3.5.182
```

The **-n** flag is not strictly needed, but you may need a IP address for your WPAR to connect to it from outside network.

The process unfolds as follows. First, the operating system creates and mounts the WPAR's file systems. Next, it populates them with the necessary system files. Finally, it synchronizes the root part of the installed software. When the creation of the new WPAR is complete, it is left in the Defined state.

Example 4-4 displays the output of the creation of a simple system WPAR (as mentioned, not all output content is shown).

*Example 4-4 Creating a simple system WPAR*

```
mkwpar -n wpar01 -N address=9.3.5.182
mkwpar: Creating file systems...
 /
 /home
 /opt
 /proc
 /tmp
 /usr
 /var
Mounting all workload partition file systems.
.
.
.
syncroot: Processing root part installation status.
syncroot: Installp root packages are currently synchronized.
syncroot: RPM root packages are currently synchronized.
syncroot: Root part is currently synchronized.
syncroot: Returns Status = SUCCESS
Workload partition wpr00 created successfully.
mkwpar: 0960-390 To start the workload partition, execute the following
as root: startwpar [-v] wpr00
```

After a system WPAR is defined, it can be started and managed similar to a normal AIX logical partition.

### *Starting a WPAR*

A WPAR is started by issuing the **`startwpar`** command, followed by its name. The global environment takes care of the necessary infrastructure, namely, mounting file systems and adding IP addresses (for details, refer to 4.1.3, "Network considerations" on page 70).

First issue **`lswpar`** to list the WPARs. Example 4-4 on page 58 shows the newly created LPAR in the Defined (`D`) state.

*Example 4-5 List WPARs*

```
# lswpar
Name   State  Type  Hostname  Directory
-------------------------------------------
wpr00  D      S     wpr00     /wpars/wpr00
#
```

Now issue **`startwpar`** to start the WPAR, as shown in Example 4-6.

*Example 4-6 Start WPAR and check state*

```
# startwpar wpr00
Starting workload partition wpr00.
Mounting all workload partition file systems.
Loading workload partition.
Exporting workload partition devices.
Starting workload partition subsystem cor_wpr00.
0513-059 The cor_wpr00 Subsystem has been started. Subsystem PID is
340192.
Verifying workload partition startup.
#
# lswpar
Name   State  Type  Hostname  Directory
-------------------------------------------
wpr00  A      S     wpr00     /wpars/wpr00
```

At this point you can to logon to the WPAR either using **`clogin`** from the global environment or telnet.

Note that **`clogin`** does not depend on a TCP/IP connection. This is shown in Example 4-7 on page 60.

*Example 4-7   Login using clogin or telnet*

```
# clogin wpr00
*******************************************************************************
*                                                                             *
*                                                                             *
*  Welcome to AIX Version 6.1!                                                *
*                                                                             *
*                                                                             *
*  Please see the README file in /usr/lpp/bos for information pertinent to    *
*  this release of the AIX Operating System.                                  *
*                                                                             *
*                                                                             *
*******************************************************************************
Last login: Wed Jun 18 11:58:37 CDT 2008 on /dev/Global from lpar00

# uname -W
5
# hostname
wpr00
```

### *Stopping a WPAR*

When you no longer need to work with this WPAR, you can stop it either from inside the WPAR or from the global environment.

**Tip**: To determine whether you are in the WPAR or inside the global environment, execute the **`uname -W`** command. This returns zero (0) if you are in the global environment, and a value other than 0 if you are inside a WPAR. You can also check the hostname or the mounted file systems.

Example 4-8 on page 61 illustrates how to stop a WPAR from inside the WPAR.

*Example 4-8 Stopping the WPAR from inside the WPAR*

```
# uname -W
5
# hostname
wpr00
# shutdown -F

SHUTDOWN PROGRAM
Wed Jun 18 12:03:38 CDT 2008

Wait for '....Halt completed....' before stopping.
Error reporting has stopped.
Advanced Accounting has stopped...
Process accounting has stopped.
nfs_clean: Stopping NFS/NIS Daemons
0513-004 The Subsystem or Group, nfsd, is currently inoperative.
0513-044 The biod Subsystem was requested to stop.
0513-044 The rpc.lockd Subsystem was requested to stop.
.
.
.....Halt completed....
```

Example 4-9 illustrates how to stop a WPAR from inside the global environment.

*Example 4-9 Stopping the WPAR from the global environment*

```
# uname -W
0
# lswpar
Name   State  Type  Hostname  Directory
-------------------------------------------
wpr00  A      S     wpr00     /wpars/wpr00
# stopwpar wpr00
Stopping workload partition wpr00.
Stopping workload partition subsystem cor_wpr00.
0513-044 The cor_wpr00 Subsystem was requested to stop.
stopwpar: 0960-261 Waiting up to 600 seconds for workload partition to
halt.
Shutting down all workload partition processes.
Unmounting all workload partition file systems.
#
```

When stopping a WPAR from the global environment, you can add the **-F** flag to the command if you want to force the WPAR to shut down. You can add the **-N** flag if you want it to shutdown immediately (that is, no delay). However, exercise care when using the force option, keeping in mind the nature of the applications running in the WPAR.

### *Rebooting a WPAR*

You can reboot a WPAR in the normal AIX way from inside the WPAR by using **`shutdown -Fr`**. Example 4-10 shows the output of this command. During the reboot, the state of the WPAR will be Transitional (`T`), which means that it is moving from one state to another.

*Example 4-10   Rebooting WPAR from inside the WPAR*

```
# uname -W
5
# hostname
wpr00
# shutdown -Fr

SHUTDOWN PROGRAM
Wed Jun 18 12:13:28 CDT 2008

Wait for 'Rebooting...' before stopping.
Error reporting has stopped.
Advanced Accounting has stopped...
Process accounting has stopped.
nfs_clean: Stopping NFS/NIS Daemons
0513-004 The Subsystem or Group, nfsd, is currently inoperative.
0513-044 The biod Subsystem was requested to stop.
0513-044 The rpc.lockd Subsystem was requested to stop.
.
.
.
Rebooting . . .
#
# uname -W
0
# hostname
lpar00
```

It is also possible to reboot a WPAR from the global environment, as shown in Example 4-11 on page 63.

*Example 4-11 Rebooting a WPAR from the global environment*

```
# rebootwpar wpr00
Stopping workload partition wpr00.
Stopping workload partition subsystem cor_wpr00.
0513-044 The cor_wpr00 Subsystem was requested to stop.
stopwpar: 0960-261 Waiting up to 600 seconds for workload partition to
halt.
Shutting down all workload partition processes.
Unmounting all workload partition file systems.
Starting workload partition wpr00.
Mounting all workload partition file systems.
Loading workload partition.
Exporting workload partition devices.
Starting workload partition subsystem cor_wpr00.
0513-059 The cor_wpr00 Subsystem has been started. Subsystem PID is
466962.
Verifying workload partition startup.
#
```

## Changing a WPAR

After a WPAR has been created, you might need to change some of its attributes. To change a WPAR's attributes, enter **chwpar** on the command line.

As an example, we created a WPAR and named it wpr00. Now we will change its name by using the command **chwpar**.

However, be aware that you can change a WPAR's name only when the WPAR is in the Defined (D) state. Notice that in Example 4-12, the WPAR is in the Active (A) state and therefore **chwpar** fails. After stopping the WPAR, **chwpar** changes the WPAR name and we check the status using the **lswpar** command.

*Example 4-12 Changing WPAR's hostname*

```
# lswpar
Name   State  Type  Hostname  Directory
-------------------------------------------
wpr00  A      S     wpr00     /wpars/wpr00
#
root@lpar00:/# chwpar -n mynewname wpr00
chwpar: 0960-053 Cannot change the name (-n) of a running workload
partition.
#
#stopwpar wpr00
.
```

```
.
root@lpar00:/# lswpar
Name   State  Type  Hostname  Directory
-------------------------------------------
wpr00  D      S     wpr00     /wpars/wpr00

root@lpar00:/# chwpar -n mynewname wpr00
root@lpar00:/# lswpar
Name       State  Type  Hostname  Directory
-----------------------------------------------
mynewname  D      S     wpr00     /wpars/wpr00
```

## Recovering from Broken state

If a WPAR gets into an undefined state it is called a Broken state, as explained in 4.1.2, “Deployment states and transitions” on page 65.

Example  displays a WPAR in the Broken (B) state.

*Example 4-13   WPAR in Broken state*

```
root@lpar00:/# lswpar
Name   State  Type  Hostname  Directory
-------------------------------------------
wpr00  B      S     wpr00     /wpars/wpr00
```

To investigate the reason for the Broken state, check the following:

- The logs at /var/adm/ras.
- The logs at /var/adm/WPARs.
- WPAR’s running processes (use **`ps -@`** to show processes by WPAR).
- Problems with WPAR mounted file systems.

Before you can start the WPAR again, it must go to the Defined state, so enter this command:

```
# stopwpar -F wpr00
```

## Removing a WPAR

To remove a system WPAR from the system, perform the following tasks:

1. Verify that the WPAR is in the Defined (D) state.
2. Make sure that you do not need the WPAR or any data inside it, or that you have a working backup of it.
3. Issue the command **`rmwpar <name of the WPAR>`**, as shown in Example 4-14 on page 65.

*Example 4-14 Removing a WPAR*

```
# rmwpar wpr00
rmwpar: Removing file system /wpars/wpr00/var.
rmlv: Logical volume fslv03 is removed.
rmwpar: Removing file system /wpars/wpr00/usr.
rmwpar: Removing file system /wpars/wpr00/tmp.
rmlv: Logical volume fslv02 is removed.
rmwpar: Removing file system /wpars/wpr00/proc.
rmwpar: Removing file system /wpars/wpr00/opt.
rmwpar: Removing file system /wpars/wpr00/home.
rmlv: Logical volume fslv01 is removed.
rmwpar: Removing file system /wpars/wpr00.
rmlv: Logical volume fslv00 is removed.
#
# lswpar
#
```

## 4.1.2 Deployment states and transitions

WPARs can be created as static, or as candidates for mobility and relocation, depending on whether the **`-c`** flag is used in the **`mkwpar`** or **`wparexec`** command when creating the partition. The number of states that a partition can reach depends on whether it was created as being capable of the checkpoint state.

### States

There are eight states that a WPAR can reach, as listed and described in Table 4-1 on page 66.

*Table 4-1 WPAR states*

| State | Display as | Description |
|---|---|---|
| Defined | D | Applies mostly to system WPARs. This is the state of a system WPAR that has been created but has not yet started the execution of any process.<br>This state also exists for application partitions, but as an internal state that does not display in normal conditions. |
| Active | A | This is the normal state of either an application partition or a system partition while executing the workload. |
| Broken | B | This state is not visible to the system administrator under normal circumstances. A partition only reaches this state when a failure occurs and the recovery process fails. |
| Transitional | T | Transitional is not a “state” in the strictest sense of the word. The **`lswpar`** command displays `T` while a WPAR is in the process of changing from one state to another. |
| Paused | P | This state only applies to mobile partitions (created with the **`-c`** flag). This state is reached when a WPAR has had a successful checkpoint or restore state, at which time it is ready to be either resumed or ended. |
| Frozen | F | This state is mostly used in the High Performance Computing (HPC) or distributed type of application. It allows you to check that all parts of the application are ready to be checkpointed without effectively performing the checkpoint. This state is rarely viewed by system administrators of non-HPC applications. |
| Loaded | L | This state applies to mobile partitions and is visible on the target global environment after a WPAR relocation. This state is mostly internal to the life cycle of a partition. It is rarely viewed by system administrators under normal circumstances. |
| Moving | M | This state applies to a WPAR created with the **`-c`** flag while memory data is moved from the departing node to the arriving node. |

### Transitions

When a WPAR is created or moved, the WPAR goes through a series of states before you are able to use the newly created WPAR. Figure 4-1 on page 67 shows the states and transitions for a **`mkwpar`** (or **`wparexec`**) command.

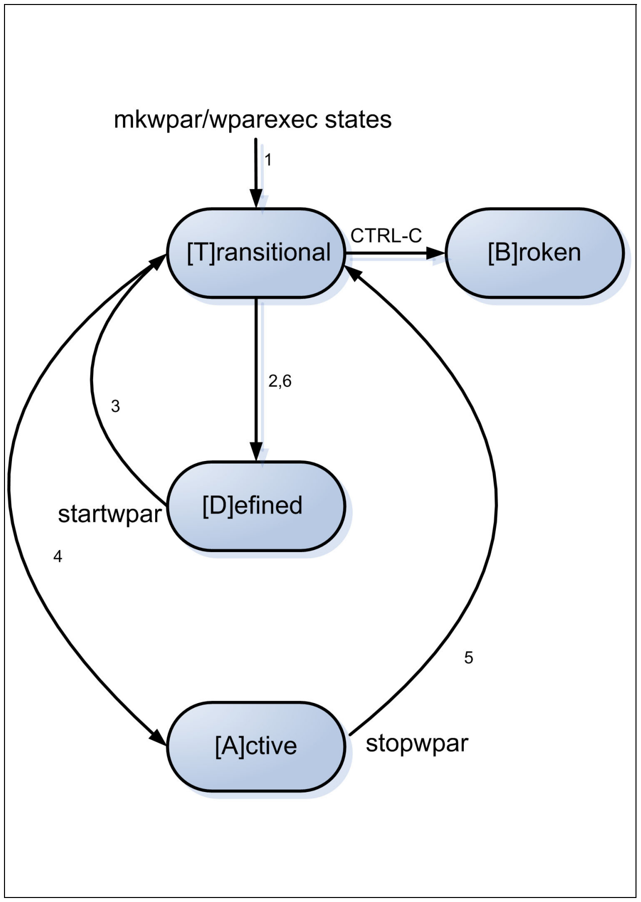


*Figure 4-1 States and transitions for mkwpar*

As shown in Figure 4-1, the flow proceeds as follows:

1. We issue the **`mkwpar`** command; our WPAR is now in the Transitional (`T`) state.

   While the WPAR is in this state, the global environment creates and mounts the WPAR's filesystems. Then it populates them with the necessary system files. Finally, it synchronizes (install) the root part of the installed software.

2. When this process is finished, the filesystems are unmounted and we change state to Defined (`D`).

3. When we issue **`startwpar`**, we again change state from `D` to `T`.

   In this phase, the global environment takes care of the necessary infrastructure, mounts file systems, adds IP address, and starts the process in the global environment.

4. When done, we change state from `T` to Active (`A`). Our WPAR are now ready for work.
5. When our WPAR has finished the work it was expected to do, we issue **`stopwpar`**, and again change state to `T`.

   Our WPAR closes all its processes. Our global environment stops the WPAR process in the global environment, removes routes and IP address, and unmounts the file systems
6. At the end, we are back to state `D`.

   Note that CTRL-C is an acronym for any interrupt that may occur while the WPAR is in the transitional stage.

For an application WPAR, it is almost the same scenario. However, an application WPAR is never in state D. It goes from T to A to- T. When finished, it is removed.

Figure 4-2 on page 69 displays the states as we move a WPAR from the Departing system to the Arriving system. For clarity we show our NFS server in the state diagram, although the NFS server is not a "state."

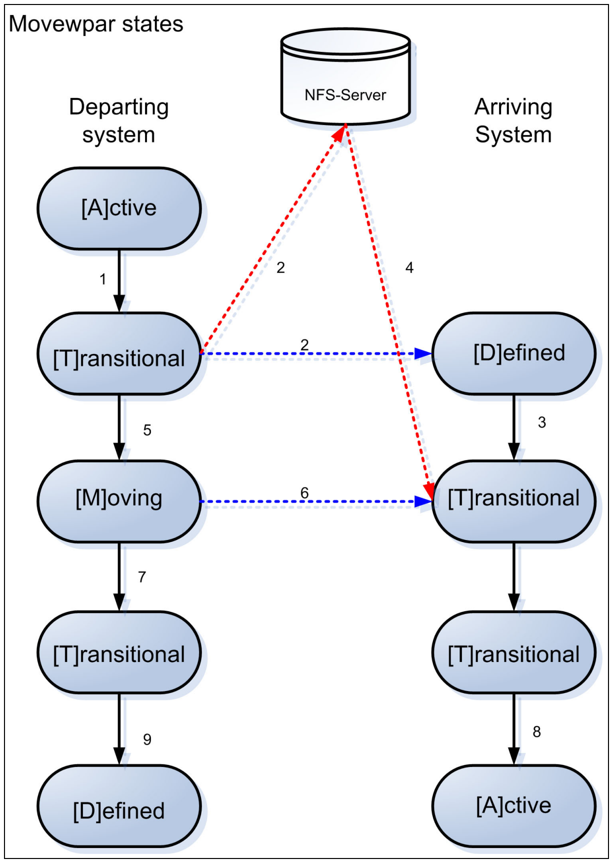


*Figure 4-2   States and transitions during movewpar*

As shown in Figure 4-2, the flow proceeds as follows:

1. Our WPAR is active, and we issue **`movewpar`**.

   The WPAR changes state to `T`, and starts the move processes.

2. We send our WPAR spec file to the arriving system. At the same time, we start saving page and segment table information on our NFS server.
3. When our Arriving system receives the spec file, it creates the WPAR. When this is done we change state to `T`.
4. We are ready for receiving memory data from departing system, and we can get page and segment table information from the NFS server.
5. While this happens on the Arriving system, our Departing system changes state from `T` to Moving (`M`).

6. The `M` state is only shown while our memory data is transmitted to the Arriving system.
7. As soon as this is finished, we change state to `T`.
8. When our Arriving system has received all needed data, it starts and changes states from `T` to `A`.
9. At the same time, our Departing system changes state to `D`.

### 4.1.3 Network considerations

A system WPAR works in the same manner as a full LPAR. You might want to have a service IP address and be able to log in to it remotely. To do so, you must consider the networks that you want to use, because you can only add IPs to the existing adapters of the global environment as aliases. Also, the networks must be routable. That is, the WPAR's IP address must be in the same subnet of the global environment's IP address. If adding an IP address from a different subnet, remember to explicitly define the interface where you want to place that IP.

**Tips:**

- To use WPAR mobility, your network adapters must have the same interface ID (en0) on each of the global environments where you want to be able to start the WPAR.
- With WPAR mobility, use fully qualified domain names (FQDNs), particularly if your systems (WPAR's global environment and NFS servers) are on different subnets or domains.
- If the WPAR name is in /etc/hosts or the DNS, then **`mkwpar`** uses this address as the default address for the WPAR. You do not need to specify the **`-n`** option while creating the WPAR.
- Use the **`-r`** option when creating your WPAR to make the WPAR adopt network name resolution from the global environment.

## 4.2 File system setup

At creation time, a WPAR can have several types of file systems: namefs, jfs, jfs2, or nfs. By default, the system creates /, /home, /tmp, and /var as jfs2 and /usr, /opt, and /proc as namefs. If you want more file systems for your applications, you need to create them. Inside the WPAR, you can mount nfs but you cannot create, change, or delete file systems. You need to do it from the global environment.

**Note:** If you are considering the use of encrypted file systems (EFS), refer to 6.3, “Encrypted File Systems” on page 156.

## 4.2.1 Creating a new filesystem

The best way to add a file system to an existing WPAR is from the CLI. To make it permanent to a WPAR, specify the mount group with the name of that WPAR. Example 4-15 shows the commands to create and mount a file system. The filesystem will be mounted on /newfs of WPAR wpr00.

*Example 4-15 Creating a filesystem for a running WPAR*

```
crfs -v jfs2 -g datavg -m /wpars/wpr00/newfs -u wpr00 -a logname=INLINE -a
size=1G

File system created successfully.
1040148 kilobytes total disk space.
New File System size is 2097152
#
# mount /wpars/wpr00/newfs
#
# clogin wpr00
*******************************************************************************
*                                                                             *
*                                                                             *
*  Welcome to AIX Version 6.1!                                                *
*                                                                             *
*                                                                             *
*  Please see the README file in /usr/lpp/bos for information pertinent to    *
*  this release of the AIX Operating System.                                  *
*                                                                             *
*                                                                             *
*******************************************************************************
# hostname
wpr00
# df -k
Filesystem    1024-blocks      Free %Used    Iused %Iused Mounted on
/dev/fslv00         98304     74000   25%     1620     9% /
/dev/fslv01         32768     31904    3%        5     1% /home
/opt               131072     23208   83%     1960    26% /opt
/proc                   -         -    -         -     -  /proc
/dev/fslv02         98304     96888    2%        7     1% /tmp
/usr              3342336   1450596   57%    38165    11% /usr
/dev/fslv03        131072    118640   10%      362     2% /var
/dev/fslv09       1048576   1039896    1%        4     1% /newfs
#
```

## 4.2.2 Changing an existing file system

To change a file system, you must be in the global environment. The changes you make are effective immediately to the WPAR.

Example 4-16 illustrates the use of the CLI to decrease the size of a WPAR's file system, but you can use smit as well.

*Example 4-16 Changing a filesystem on a running WPAR*

```
# hostname
lpar00
# chfs -a size=512M /wpars/wpr00/newfs
Filesystem size changed to 1048576
Inlinelog size changed to 4 MB.
#
#
# clogin wpr00
*******************************************************************************
*                                                                             *
*                                                                             *
*  Welcome to AIX Version 6.1!                                                *
*                                                                             *
*                                                                             *
*  Please see the README file in /usr/lpp/bos for information pertinent to    *
*  this release of the AIX Operating System.                                  *
*                                                                             *
*                                                                             *
*******************************************************************************
Last login: Wed Jun 18 13:41:49 CDT 2008 on /dev/Global from lpar00

# hostname
wpr00
# df -k
Filesystem    1024-blocks      Free %Used    Iused %Iused Mounted on
/dev/fslv00         98304     74000   25%     1618     9% /
/dev/fslv01         32768     31904    3%        5     1% /home
/opt               131072     23208   83%     1960    26% /opt
/proc                   -         -    -         -     -  /proc
/dev/fslv02         98304     96888    2%        7     1% /tmp
/usr              3342336   1450596   57%    38165    11% /usr
/dev/fslv03        131072    118632   10%      362     2% /var
/dev/fslv09        524288    519784    1%        4     1% /newfs
```

## 4.2.3 Shared /usr with writable subdirectory (filesystem)

Default system WPARs use a shared /usr and /opt. As we have seen, they are read-only. If you must write to a local directory under /usr (for example, /usr/local), you need to create a new filesystem for that.

To have a writable filesystem under /usr, create a new file system in the global environment and mount it under the WPAR's structure. Then, go to the global environment's /usr and make a soft link to that file system. Example 4-17 demonstrates this task.

This scenario involves two running WPARs, wpr00 and wpr01. Each WPAR needs a local, private, and writable file system called /usr/WebSphere.

Example 4-17 illustrates these steps:

1. Start WPAR wpr00 and wpr01.
2. Create the file system for wpr00.
3. Create the file system for wpr01.
4. Mount both file systems.
5. Create a link in the global partition.
6. Create a file in each WPAR's /usr/WebSphere.

*Example 4-17 Creating a writable subdirectory in read-only /usr*

```
# startwpar wpr00
..
# startwpar wpr01
..
# lswpar
Name   State  Type  Hostname  Directory
-------------------------------------------
wpr00  A      S     wpr00     /wpars/wpr00
wpr01  A      S     wpr01     /wpars/wpr01
#
#crfs -v jfs2 -g datavg -m /wpars/wpr00/WebSphere -u wpr00 -a
logname=INLINE -a size=1G
File system created successfully.
1040148 kilobytes total disk space.
New File System size is 2097152
#
#crfs -v jfs2 -g datavg -m /wpars/wpr01/WebSphere -u wpr01 -a
logname=INLINE -a size=1G
File system created successfully.
1040148 kilobytes total disk space.
New File System size is 2097152
```

```
#
# mount /wpars/wpr00/WebSphere
# mount /wpars/wpr01/WebSphere
#
#cd /usr
ln -s /WebSphere /usr/WebSphere

clogin wpr00
# hostname
wpr00
cd /usr/WebSphere
# touch wpr00_file_in_writable_usr_WebSphere
# ls -l
total 0
drwxr-xr-x    2 root     system          256 Jun 18 14:15 lost+found
-rw-r--r--    1 root     system            0 Jun 18 14:25
wpr00_file_in_writable_usr?WebSphere
#
# hostname
wpr01
#cd /usr/WebSphere
# touch wpr01_file_in_writable_usr_WebSphere
# ls -l
total 0
drwxr-xr-x    2 root     system          256 Jun 18 14:14 lost+found
-rw-r--r--    1 root     system            0 Jun 18 14:27
wpr01_file_in_writable_usr_WebSphere
```

## 4.3 Software installation

By default, system WPARs have two read-only file systems: /usr and /opt. Although this saves space on disk and saves time during WPAR creation, it prevents normal software installation inside the WPAR. Also, the WPAR's root user cannot delete software from /usr or /opt directories.

Therefore, installing software on a system WPAR requires a different approach. There are usually two ways to install software:

- In the global environment, using normal installation procedures
- Inside the WPAR, using read/write file systems

Application WPARs do not pose a problem regarding software installation, because they run only a single application that is already present in the global environment.

The global environment is regular AIX running in an LPAR. Accessing the CD-ROM or DVD-ROM, an NFS mount or NIM is the same as usual. However, for WPARs, there are differences.

## 4.3.1 Software availability

To have software available inside the WPAR, you need to have it in a file system form, the software repository, a mounted CD or DVD, or an NFS mount.

If the software you want to install is in a CD format, copy it to a regular file system. You can use **`smit bffcreate`** to do that. Just mounting the CD might be enough for the operation that you want. In this case, pass the mount point as a namefs parameter to the WPAR at creation time: **`-M directory=/cdrom dev=/cdrom`**.

If your WPAR is already running, the best way to temporarily mount file systems is under /WPAR’s/<name of the WPAR>/<my mount point> in the global environment, as illustrated in 4.2, “File system setup” on page 70. After creating the new (temporary) file system, copy the software from your CD to that file system.

## 4.3.2 Installing software

Software can be installed in the global environment, or directly in the WPAR. To install software directly in the WPAR, either create the WPAR with a writable /usr or install the software in a writable file system like /home.

If the software should be generally available to other WPARS, install it in the global environment. If the software is only to be used in one WPAR, install it in this WPAR. The choice you make depends on your needs and your application.

### Installing software in the global environment

Example 4-18 on page 76 displays global environment lpar00 and WPAR wpr00, which has a read-only /usr. The example first shows the software is installed, then the WPAR is synchronized.

The software was copied to the /source directory in the global environment and then installed from the command line with the **`installp`** command:

```
installp -ac -d /source X11.Dt
```

After installation, the WPAR was synchronized from the global environment using the command **syncwpar wpr00**. Synchronization can also be performed from inside the WPAR by using the command **syncroot**.

**Note:** Synchronize the WPAR *after* the software installation in the global environment to prevent inconsistencies between the root and user parts of the WPARs.

*Example 4-18 Installing X11.Dt in global environment and synchronizing wpr00*

```
root@lpar00:/source# hostname
lpar00
root@lpar00:/source# installp -av -d /source X11.Dt

FILESET STATISTICS
------------------
    7  Selected to be installed, of which:
        7  Passed pre-installation verification
  ----
    7  Total to be installed

+-----------------------------------------------------------------------------+
                         Installing Software...
+-----------------------------------------------------------------------------+

installp:  APPLYING software for:
        X11.Dt.rte 6.1.0.0
        X11.Dt.lib 6.1.0.1
        X11.Dt.helprun 6.1.0.0
        X11.Dt.helpmin 6.1.0.0
        X11.Dt.helpinfo 6.1.0.0
        X11.Dt.bitmaps 6.1.0.0
        X11.Dt.ToolTalk 6.1.0.0
.
.
.
Installation Summary
--------------------
Name                        Level           Part        Event       Result
-------------------------------------------------------------------------------
X11.Dt.rte                  6.1.0.0         USR         APPLY       SUCCESS
X11.Dt.lib                  6.1.0.1         USR         APPLY       SUCCESS
X11.Dt.helprun              6.1.0.0         USR         APPLY       SUCCESS
X11.Dt.helpmin              6.1.0.0         USR         APPLY       SUCCESS
X11.Dt.helpinfo             6.1.0.0         USR         APPLY       SUCCESS
X11.Dt.bitmaps              6.1.0.0         USR         APPLY       SUCCESS
X11.Dt.ToolTalk             6.1.0.0         USR         APPLY       SUCCESS
```

```
X11.Dt.rte                 6.1.0.0         ROOT        APPLY       SUCCESS
X11.Dt.helpmin             6.1.0.0         ROOT        APPLY       SUCCESS
X11.Dt.helpinfo            6.1.0.0         ROOT        APPLY       SUCCESS
X11.Dt.bitmaps             6.1.0.0         ROOT        APPLY       SUCCESS
X11.Dt.ToolTalk            6.1.0.0         ROOT        APPLY       SUCCESS

# hostname
wpr00
# lslpp -l | grep -i x11.dt
  X11.Dt.ToolTalk            6.1.0.0  COMMITTED  AIX CDE ToolTalk Support
  X11.Dt.bitmaps             6.1.0.0  COMMITTED  AIX CDE Bitmaps
  X11.Dt.helpinfo            6.1.0.0  COMMITTED  AIX CDE Help Files and Volumes
  X11.Dt.helpmin             6.1.0.0  COMMITTED  AIX CDE Minimum Help Files
  X11.Dt.helprun             6.1.0.0  COMMITTED  AIX CDE Runtime Help
  X11.Dt.lib                 6.1.0.1  COMMITTED  AIX CDE Runtime Libraries
  X11.Dt.rte                 6.1.0.0  COMMITTED  AIX Common Desktop Environment
#
#
#
#
root@lpar00:/source# syncwpar wpr00
*******************************************************************************
Synchronizing workload partition wpr00 (1 of 1).
*******************************************************************************
Executing /usr/sbin/syncroot in workload partition wpr00.
syncroot: Processing root part installation status.
syncroot: Synchronizing installp software.
+-----------------------------------------------------------------------------+
                    Pre-installation Verification...
+-----------------------------------------------------------------------------+
Verifying selections...done
Verifying requisites...done
Results...

SUCCESSES
---------
  Filesets listed in this section passed pre-installation verification
  and will be installed.

  Selected Filesets
  -----------------
  X11.Dt.ToolTalk 6.1.0.0                     # AIX CDE ToolTalk Support
  X11.Dt.bitmaps 6.1.0.0                      # AIX CDE Bitmaps
  X11.Dt.helpinfo 6.1.0.0                     # AIX CDE Help Files and Volumes
  X11.Dt.helpmin 6.1.0.0                      # AIX CDE Minimum Help Files
  X11.Dt.rte 6.1.0.0                          # AIX Common Desktop Environme...

  << End of Success Section >>
Installation Summary
```

```
--------------------
Name                        Level           Part        Event       Result
-------------------------------------------------------------------------------
X11.Dt.bitmaps              6.1.0.0         ROOT        APPLY       SUCCESS
X11.Dt.helpinfo             6.1.0.0         ROOT        APPLY       SUCCESS
X11.Dt.helpmin              6.1.0.0         ROOT        APPLY       SUCCESS
X11.Dt.ToolTalk             6.1.0.0         ROOT        APPLY       SUCCESS
X11.Dt.rte                  6.1.0.0         ROOT        APPLY       SUCCESS
syncroot: Processing root part installation status.
syncroot: Installp root packages are currently synchronized.
syncroot: RPM root packages are currently synchronized.
syncroot: Root part is currently synchronized.
syncroot: Returns Status = SUCCESS
Workload partition wpr00 synchronized successfully.

Return Status = SUCCESS.
```

### Installing software in the system WPAR

To install software inside the system WPAR, you must have writable file systems. These file systems can be private /usr and /opt, or a required file system for an application installation. The process for installing in the system WPAR is exactly the same as in the global environment, but you do not need to synchronize. Simply install using installp, smit, or your application's scripts.

**Note:** Certain application installation scripts search for particular devices. If the your application's scripts fail inside the WPAR, consider installing in the global environment.

## 4.4 Backing up, restoring, and cloning WPARs

This section describes common methods of backing up and restoring WPARs and the global environment. It also provides examples that show methods for cloning WPARs.

### 4.4.1 Backup considerations

When backing up WPARs and the global environment in which the WPAR runs, there are certain considerations to be aware of.

To back up the global environment, use the regular tools.

To back up rootvg, use **mksysb**. To back up other vgs, **savevg** is normally used. There are other methods which can be used (including **alt_disk_install**, **mkdvd** and others). All these standard tools conform to the following considerations.

Note that, by default, **mksysb** and **savevg** will only back up file systems that are mounted. This means that if you have created ten WPARs in your global environment, but only one is running, then only the running WPAR's file systems will be included the backup. This is because inactive WPAR file systems are not mounted, and unmounted file systems by convention are not backed up.

The backup created by using the standard **mksysb** command will contain all ten WPAR definitions but the filesystems for the nine inactive WPARs will be missing.

**Tip:** The new flag **-N** has been added to **mksysb**, so you are able to back up unmounted WPAR file systems. The behavior of **smitty mksysb** is changed to always back up active or inactive WPAR file systems.

The same flag exists for **mkdvd, mkcd** and **mkszfile**. For **alt_disk_install**, the new flag is **-u**.

Also be aware that, if you back up your global environment with running WPARs and the application inside the WPARs is running when you perform the backup, then you risk ending up with an unusable backup because your application inside the WPAR can hold or update files. Therefore, you face the risk of making WPAR application data in the backup inconsistent.

So make sure all applications inside all your WPARs are stopped to ensure that your backups are consistent.

### 4.4.2 WPAR considerations

If a relocatable WPAR is created, you must ensure you synchronize backups of the WPAR and its NFS mounted filesystems. Note the following points:

- For application WPARs there is really nothing to do, because all data and definitions are backed up while backing up the global environment.
- For system WPARs, the **savewpar** and **restwpar** commands are used to back up and restore a single WPAR, in the same way as **mksysb** and **savevg** are used to back up AIX. Both **savewpar** and **restwpar** have to be run from the global environment.

When backing up a WPAR, employ the same considerations as when backing up AIX. For example, stop your application inside the WPAR before running the

**savewpar** command. Also ensure that all filesystems for a given WPAR are mounted.

## 4.4.3 Backing up the global environment

The simplest way to back up the global environment is to stop all WPARs. Then run a **mksysb/mkdvd,mkcd** command with the **-N** flag. Although it is possible to run this command while WPARs are running, make sure your applications inside the WPAR ares stopped. Otherwise, your backup might not be consistent.

**Tip:** If you have a WPAR filesystem in volume groups other than rootvg, then you must mount all filesystems for the inactive WPAR before using **savevg**. This way you can be sure that all data is consistent, and that your mksysb/savevg file can be used to recover your global system and all your WPARs in one operation.

Another possibility is to use **savewpar** on every created WPAR, one WPAR at a time. Remember that for **savewpar** it is also important that applications inside the WPAR be stopped.

Depending on how many WPARs you have created this can be time-consuming, both when you perform the **savepar** and in a disaster recovery situation.

In the real world, a combination of the **mksysb** solution and **savewpar** is often used. The **mksysb** solution described allows you to restore the global environment and all WPARs in one operation, which is useful in a disaster recovery situation. But in daily operation, you might want to restore (or recreate) a given WPAR without touching the rest of the global environment. You choice of solution must conform to your company's backup and restore policy, as well as its disaster recovery policy.

**Important:** If your NFS client systems are running AIX 6.1 prior to TL1 and you are using audit, you will experience dramatically slow performance using NFS. To avoid this, either update to the latest technology level (TL2, at the time of writing) or apply Fix IZ22120 PTF U814482 to TL1.

## 4.4.4 Backing up and restoring a system WPAR

This section explains how to back up and restore a system WPAR.

## Backing up

Use the **savewpar** command to back up a system WPAR. The **savewpar** command works in a similar way to the **savevg/mksysb** commands used in the global environment. It saves the files and the configuration of the WPAR.

When you have a system WPAR with shared /usr and /opt, the backup is very small, because it does not save those file systems. Note that you must be in the global environment to execute this backup.

The following examples illustrate a backup of the WPAR used in 4.2.3, "Shared /usr with writable subdirectory (filesystem)" on page 73. This WPAR includes the writable /usr/WebSphere directory. Example 4-19 shows how to back up a system WPAR.

*Example 4-19 Backing up a system WPAR*

```
# hostname
lpar00
# savewpar -f /backup/wpr00.save wpr00

Creating list of files to back up.
Backing up 2160 files
2160 of 2160 files (100%)
0512-038 savewpar: Backup Completed Successfully.
```

## Restoring a WPAR

If you accept the defaults, no flags are needed to restore a WPAR. In most cases, you have a file as a backup and want to restore that file. In that case, use the **-f** flag as shown in Example 4-20. In that example, wpr00 is removed and then restored and started. After the WPAR is started, the file is checked in /usr/WebSphere.

The **restwpar** command has three major steps:

1. Create the necessary file systems according to the image.data file that is created with the **savewpar** command, and mount them.
2. Restore the files in the backup to their proper places. This might include /usr and /opt, depending on the type of WPAR.
3. Synchronize the WPAR with the global environment.

*Example 4-20 Restoring a system WPAR*

```
# rmwpar wpr00
rmwpar: Removing file system /wpars/wpr00/var.
```

```
rmlv: Logical volume fslv03 is removed.
rmwpar: Removing file system /wpars/wpr00/usr.
rmwpar: Removing file system /wpars/wpr00/tmp.
rmlv: Logical volume fslv02 is removed.
rmwpar: Removing file system /wpars/wpr00/proc.
rmwpar: Removing file system /wpars/wpr00/opt.
rmwpar: Removing file system /wpars/wpr00/home.
rmlv: Logical volume fslv01 is removed.
rmwpar: Removing file system /wpars/wpr00/WebSphere.
rmlv: Logical volume fslv09 is removed.
rmwpar: Removing file system /wpars/wpr00.
rmlv: Logical volume fslv00 is removed.
#
# restwpar -f /backup/wpr00.save
New volume on /backup/wpr00.save:
Cluster size is 51200 bytes (100 blocks).
The volume number is 1.
The backup date is: Wed Jun 18 15:28:20 CDT 2008
Files are backed up by name.
The user is root.
x         2889 ./.savewpar_dir/wpar.spec
x         6075 ./.savewpar_dir/image.data
x       149187 ./.savewpar_dir/backup.data
The total size is 158151 bytes.
The number of restored files is 3.
mkwpar: Creating file systems...
 /
Creating logical volume 'fslv00' specified in image.data
.
.
.The number of restored files is 2160.
syncroot: Processing root part installation status.
syncroot: Installp root packages are currently synchronized.
syncroot: RPM root packages are currently synchronized.
syncroot: Root part is currently synchronized.
syncroot: Returns Status = SUCCESS
Workload partition wpr00 created successfully.
mkwpar: 0960-390 To start the workload partition, execute the following
as root: startwpar [-v] wpr00
#
#clogin wpr00
# hostname
wpr00
#cd /usr/WebSphere
# touch wpr00_file_in_writable_usr_WebSphere
# ls -l
total 0
drwxr-xr-x    2 root     system          256 Jun 18 14:15 lost+found
-rw-r--r--    1 root     system            0 Jun 18 14:25
wpr00_file_in_writable_usr_WebSphere
```

## 4.4.5 Cloning a system WPAR

To clone a system in AIX, often a mksysb-image is simply created. This mksysb-image is often called the “golden image.” It is then used to restore (clone) a system on a new location. In the same way, you can create a golden WPAR image and use it for cloning.

To clone a WPAR, use the **savewpar** and **restwpar** commands as shown in Example 4-21. As shown, one working WPAR named wpr00 is cloned in the same global environment to new WPAR named wpr01. (The **savewpar** command is the same as used in Example 4-19 on page 81.)

Follow these steps to clone a WPAR:

1. Select a new name for the cloned WPAR (in our case, wpr01).
2. Obtain the IP address for the cloned WPAR name that was chosen (in our case, the cloned WPAR’s IP was 9.3.5.183).
3. Because the WPAR is in the same global environment, you must choose a different mount point: /wpars/wpr01. The **restwpar** command tries to restore the file systems to logical volumes with the name specified in the image.data of the backup file. In the example, these LVs already exist. The restore detects and corrects this using different, unique LV names.
4. After collecting the selected information, issue the **restwpar** command with extra flags. (For a detailed description of the flags used here, refer to **restwpar** documentation.)

*Example 4-21 Cloning a system WPAR*

```
restwpar -n wpr01 -h wpr01 -d /wpars/wpr01 -M ‘-N address=9.3.5.182’ -U
-f /backup/wpr00.data
New volume on /backup/wpr00.save:
Cluster size is 51200 bytes (100 blocks).
The volume number is 1.
The backup date is: Wed Jun 18 15:28:20 CDT 2008
Files are backed up by name.
The user is root.
x         2889 ./.savewpar_dir/wpar.spec
x         6075 ./.savewpar_dir/image.data
```

```
x       149187 ./.savewpar_dir/backup.data
The total size is 158151 bytes.
The number of restored files is 3.
mkwpar: Creating file systems...
 /
The total size is 30766971 bytes.
The number of restored files is 2160.
syncroot: Processing root part installation status.
syncroot: Installp root packages are currently synchronized.
syncroot: RPM root packages are currently synchronized.
syncroot: Root part is currently synchronized.
syncroot: Returns Status = SUCCESS
Workload partition wpr01 created successfully.
mkwpar: 0960-390 To start the workload partition, execute the following
as root: startwpar [-v] wpr01
```

This cloning was straightforward because one WPAR was cloned from rootvg to the same global environment. Also in rootvg, only the name and IP address were changed. In the real world, however, cloning is more complicated because you probably want to restore on other systems and maybe also in other vgs.

Next is a more realistic scenario. It involves restoring the golden WPAR image in another LPAR, and on this LPAR restoring the WPAR in a vg other than rootvg. Perform these tasks:

1. Create the **`savewpar`** image (the gold image), just as shown in Example 4-19 on page 81.
2. Running **`savewpar`** also creates a image.data file in /tmp/wpardata/wparname. This file contains general information about where filesystems reside. Copy this file and store it together with the golden WPAR image file. Then copy this image.data file and the golden WPARimage file to the new LPAR.
3. Edit the image.data file so it reflects the disk and vg setup on the target LPAR.

   On the source (lpar00) system, the WPAR was installed in rootvg, and rootvg was created on hdisk0.

   On the target system (lpar01), restore in datavg, and datavg is created on hdisk1.

   In our case we used **`vi`** and substitute commands to edit the image.data file.

**Tip:** We used the following **`vi`** substitute commands to change hdisk0 to hdisk1 and rootvg to datavg:

```
:1,$/rootvg/datavg/g
:1,$s/rootvg/datavg/g
```

Although you can use **`vi`**, we recommend that you create a script for this task.

4. Start the **`restwpar`** command, as shown in Example 4-22.

   The example refers to the edited image.data file with the **`-i`** option. After the restore, we are in lpar01, and have restored the WPAR in datavg.

*Example 4-22 Cloning a WPAR on another WPAR using the golden image*

```
# restwpar -n wpr02 -h wpr02 -d /wpars/wpr02 -i /backup/image.data.rootvg
-M '-N address=9.3.5.184' -U -f /backup/wpr00.save

New volume on /backup/wpr00.save:
Cluster size is 51200 bytes (100 blocks).
The volume number is 1.
The backup date is: Wed Jun 18 15:28:20 CDT 2008
Files are backed up by name.
The user is root.
x         2889 ./.savewpar_dir/wpar.spec
x         6075 ./.savewpar_dir/image.data
x       149187 ./.savewpar_dir/backup.data
The total size is 158151 bytes.
The number of restored files is 3.
mkwpar: Creating file systems...
 /
ATTENTION: Logical volume 'fslv00' is not unique. Renaming to 'wlv0'.
Creating logical volume 'wlv0' specified in image.data
Creating file system '/' specified in image.data
 /WebSphere
.
.
syncroot: Processing root part installation status.
syncroot: Installp root packages are currently synchronized.
syncroot: RPM root packages are currently synchronized.
syncroot: Root part is currently synchronized.
syncroot: Returns Status = SUCCESS
Workload partition wpr01 created successfully.
mkwpar: 0960-390 To start the workload partition, execute the following as
root: startwpar [-v] wpr01

# hostname
```

```
lpar01
# uname -W
0
# lswpar
Name   State  Type  Hostname  Directory
-------------------------------------------
wpr01  D      S     wpr01     /wpars/wpr01

# lsvg -l datavg
datavg:
LV NAME             TYPE       LPs     PPs     PVs  LV STATE      MOUNT
POINT
loglv00             jfs2log    1       1       1    open/syncd    N/A
fslv00              jfs2       16      16      1    open/syncd    /backup
wlv0                jfs2       1       1       1    closed/syncd
/wpars/wpr01
fslv09              jfs2       8       8       1    closed/syncd
/wpars/wpr01/WebSphere
fslv01              jfs2       1       1       1    closed/syncd
/wpars/wpr01/home
fslv02              jfs2       1       1       1    closed/syncd
/wpars/wpr01/tmp
fslv03              jfs2       1       1       1    closed/syncd
/wpars/wpr01/var
#
```

## 4.5 User and group management

In a WPAR environment, user management depends on whether you have a system or an application WPAR. Because application WPARs use the same file systems as the global environment, they have the same authentication files and methods. Managing users and groups in the global environment is the same as with regular AIX. This topic covered in greater detail in 6.4, “Users and groups” on page 165.

### 4.5.1 Defaults access: Users and root

In system WPARs, you can manage users and groups inside the WPAR by using smit or the CLI. Users inside a system WPAR are not related to users of the global environment. There is a root user for the global environment, and another root user for each WPAR.

In the same way, if there is a user “pac” with ID 204 inside wparA, and another user “pac” with the same ID in wparB in the same global environment, they cannot interfere with each other’s processes and files.

The root user inside a WPAR only has access to the virtual operating system. In a default system, the WPAR /usr and /opt are read-only. Thus, the root user inside the WPAR is unable to delete files from these directories.

If, by mistake, the root user inside wparA removes all files that the user is allowed to remove, the user is only removing what is under /wpars/wparA of the global environment. At this point, the user must restore a previous backup of that WPAR or recreate the WPAR—but the user does not affect the global environment and the other WPARs. Management of users and groups inside a system WPAR is essentially the same as in the global environment.

### 4.5.2 Recovering a password for users inside WPARs

If the owner of a system WPAR loses the root password, the owner can ask the global environment’s system administrator to reset it. To reset the root password:

1. Log in to the global environment as root and chroot to the proper WPAR:

   ```
   # ; /wpars/wpr00 /bin/ksh
   ```

2. Use the **passwd** command to change the password:

   ```
   # passwd
   Changing password for “root”
   root's New password:
   Enter the new password again:
   # exit
   ```

The same method can be used if the global environment administrator needs to reset a user inside a WPAR:

```
# chroot /wpars/wpr00 /bin/ksh
# passwd wparuser
Changing password for "wparuser"
wparuser's New password:
Enter the new password again:
# exit
```

For the global environment’s recovery procedures, use standard AIX recovery procedures such as booting from CD-ROM or NIM.

## 4.6 Relocation

A brief introduction to mobility is given in 1.4.2, “Live application mobility” on page 12.

At this point you should be aware that relocating is only possible *if* you have installed mcr.rte, which is part of WPAR Manager. IBM Workload Partitions Manager for AIX is a optional, separately installable licensed program product that supports more advanced features (for example, if you want to use the WPAR mobility feature (manual /automated) and so on).

WPAR Manager is described in detail in Chapter 5, “Managing workload partitions” on page 99.

The first version of WPAR Manager offered only one method for relocating WPARs. This method required the remote (shared) data to be on a NFS server. In this way the WPAR could access remote data from different locations. Also, the WPAR had to be in Active state before relocation could take place.

WPAR Manager Version 1.2 introduces a new feature called *static relocation*, or *non-live relocation*. This feature can relocate WPARs from Active or Defined state. Also, static relocation can only be performed on WPARs that do *not* use remote file systems. For remote data (shared file systems), you must use *asynchronous relocation*, which is the enhanced version of the relocation mechanism that was introduced in the first WPAR Manager versions.

Both application WPARs and system WPARs can use static relocation. There is a slight difference in relocation methods between these types of WPARs.

### Using static relocation

To perform a static relocation for system WPARs, issue **savewpar** on the departing system and **restwpar** on the arriving system. Table 4-2 lists the stages for static relocation of a system WPAR.

*Table 4-2 Stages for static relocation of a system WPAR*

| Departure | Arrival | Notes |
|---|---|---|
| Verify WPAR is either Active or Defined | | |
| Lock WPAR | | |
| | Verify WPAR does not exist | |
| | Lock WPAR | |

| Departure | Arrival | Notes |
|---|---|---|
| Update WPAR spec if modified | | |
| | | Set last known valid ID to source<br>Set managed server ID to target<br>Set state to Transitional |
| Stop WPAR if ACTIVE | | |
| Save WPAR | | Execute **`savewpar`** with necessary flags |
| | Restore WPAR | Execute **`restwpar`** with necessary flags |
| | Start WPAR that was originally Active | |
| | Verify WPAR Active if it was originally Active | Mark WPAR Active in database |
| | Verify WPAR Services if it was originally Active | |
| | Update WPAR Spec | |
| Remove WPAR | | |
| | Unlock WPAR | |
| Unlock WPAR | | |
| Remove image | | |

For application WPARs, non-live relocation is performed using “offline” relocation. Table 4-3 lists the stages for static relocation of an application WPAR.

*Table 4-3 Stages for static relocation of an application WPAR*

| Departure | Arrival | Notes |
|---|---|---|
| Verify WPAR Active | | |
| Lock WPAR | | |
| | Verify WPAR does not exist | |

| Departure | Arrival | Notes |
|---|---|---|
| | Lock WPAR | |
| Update WPAR spec if modified | | |
| | | Set last known good ID to source<br>Set managed server ID to target<br>Set state to Transitional |
| Stop WPAR | | |
| | Start WPAR | |
| | Verify WPAR Active | Mark WPAR Active in database |
| | Verify WPAR services | |
| | Update WPAR spec | |
| | Unlock WPAR | |
| Unlock WPAR | | |

### Asynchronous relocation

In asynchronous relocation, you checkpoint data to the NFS filesystem. Then you give control to the arriving LPAR, restarting the checkpointed data from NFS share. The path is indicated by the blue solid lines in Figure 4-3 on page 91.

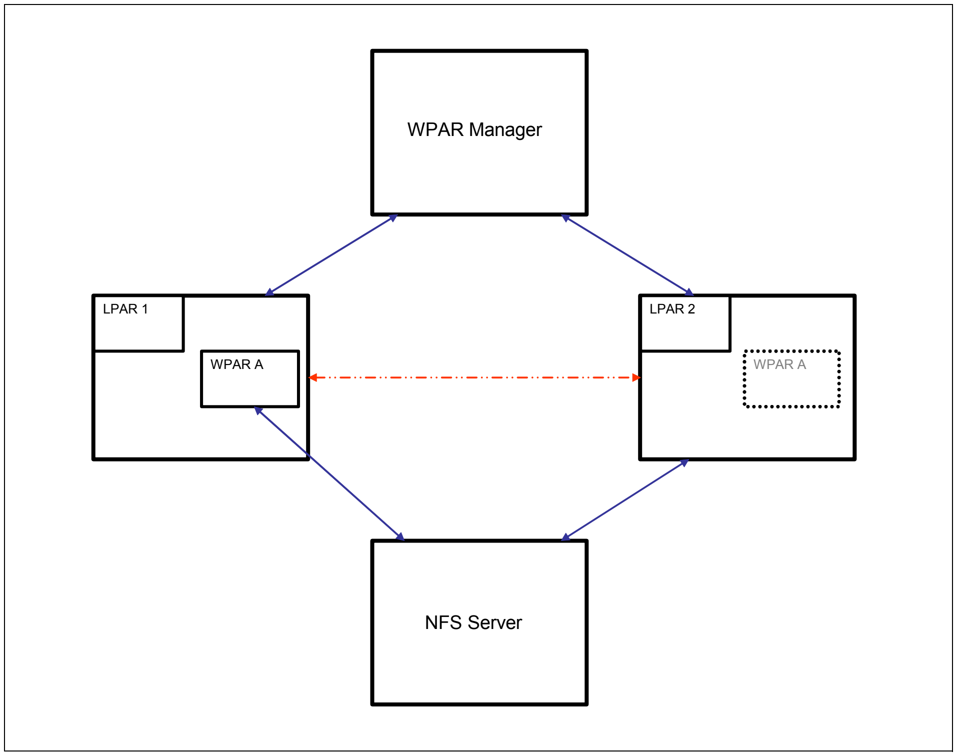


*Figure 4-3   Asynchronous relocation*

Only page table information and segment table information is stored on the NFS server. Paging data is transmitted over the TCP/IP connection from the Departing LPAR to the Arriving LPAR (indicated by the red dotted line in Figure 4-3) at the same time. Therefore, there are “parallel functions running,” and the WPAR is moved to the new location faster.

## 4.6.1 Administrative locking

Administrative locking is a new feature. As shown in Table 4-2 on page 88, we lock our WPAR as a part of the static relocation process.

A Workload Partition Administrative Lock is an advisory lock associated with a particular workload partition, process tree, and operation. Only one lock per workload partition may exist at any time.

For example, base workload partition commands such as **`mkwpar`**, **`startwpar`**, and **`chwpar`** manage locks; **`lswpar`** does not manage locks.

The same method is available for administrators, so administrators can lock a WPAR and prevent it from being managed by other users.

In Example 4-23, we create a process (385230) and set a lock to that process on wpr00. After the lock is set, we try to stop wpar00.

*Example 4-23 Locking a WPAR*

```
/usr/lib/wpars/wparlockctl  -l -p 385230 -o 6 wpr00
root@lpar00:/source# stopwpar wpr00
stopwpar: 0960-246 Workload partition wpr00 is currently locked by
stopwpar (PID = 385230).
root@lpar00:/source#
```

Next, we unlock from our process and were again able to stop our WPAR.

*Example 4-24 Unlocking a WPAR*

```
/usr/lib/wpars/wparlockctl  -u -p 385230 -o 6 wpr00
root@lpar00:/source# stopwpar wpr00
stopwpar: 0960-246 Workload partition wpr00 is currently locked by
stopwpar (PID = 385230).
root@lpar00:/source# /usr/lib/wpars/wparlockctl  -u -p 385230 -o 6
wpr00
root@lpar00:/source# stopwpar wpr00
Stopping workload partition wpr00.
Stopping workload partition subsystem cor_wpr00.
0513-044 The cor_wpr00 Subsystem was requested to stop.
stopwpar: 0960-261 Waiting up to 600 seconds for workload partition to
halt.
Shutting down all workload partition processes.
Unmounting all workload partition file systems.
```

For further details about **`wparlockctl`** commands, refer to the command documentation.

## 4.7 CLI walkthrough

Chapter 5, "Managing workload partitions" on page 99, describes how to manage WPARs using WPAR Manager. Even though WPAR Manager provides a flexible WPAR management environment, you may still want to execute mobility functions from the CLI, without using WPAR Manager. This section describes how this can be done.

### 4.7.1 Static mobility

For system WPARs, static mobility is accomplished using the **savewpar** and **restwpar** commands. For application WPARs, use **stopwpar** and then move the application.

If you are planning to move a WPAR that is eligible for static mobility to another system, you can consider it a cloning, with an extra **rmwpar** as the last command you run on the initial host. Refer to 4.4.4, "Backing up and restoring a system WPAR" on page 80 and 4.4.5, "Cloning a system WPAR" on page 83, for more detailed information about this topic.

### 4.7.2 Creating WPARs eligible for Live mobility from CLI

Be aware that Live mobility is only possible if you have installed mcr.rte, which is part of WPAR Manager. IBM Workload Partitions Manager for AIX is an optional, separately instable licensed program product that supports more advanced features. This section explains how to create a WPAR that is eligible for Live mobility from the command line.

**Tip:** The **movewpar** command only works on WPARs that are eligible for Live mobility. This means that the WPAR must have file systems on an NFS server, and that **Enable checkpoint** must be selected when the WPAR is created.

#### Creating a mobile application WPAR

Example 4-25 illustrates how to create an application WPAR by the name of wpr01, which should be eligible for Live mobility. In our case, we use NFS server lpar02, which has exported the share /wpars_share. The local mountpoint is /mnt/wpr01. The application is myapp.

*Example 4-25 Creating a mobile application WPAR*

```
wparexec -c -n wpr01 -M host=lpar02 directory=/mnt/wpr01 dev=/wpar_share /hga/myapp
```

To save the configuration specification of the WPAR in a file, add the flag **-o** and an output filename. You can later use this file for cloning or recreating the WPAR.

### Creating a mobile system WPAR

Example 4-26 illustrates how to create a system WPAR by the name of wpr01, which should be eligible for Live mobility. In our case we use NFS server lpar02, which has exported the share /wpars_share. On this share we created directory wpr01, and we want all file systems for wpr01 to be in this directory.

*Example 4-26 Creating a mobile system WPAR*

```
mkwpar -n wpr01 -h wpr01 -c -p -M dev=/wpar_share/wpr01 directory=/ host=lpar02
vfs=nfs -M dev=/usr directory=/usr mountopts=ro vfs=namefs -M
dev=/wpar_share/wpr01/tmp directory=/tmp host=lpar02 vfs=nfs -M
dev=/wpar_share/wpr01/var directory=/var host=lpar02 vfs=nfs -M dev=/opt
directory=/opt mountopts=ro vfs=namefs -M dev=/wpar_share/wpr01/home directory=/home
host=lpar02 vfs=nfs
```

## 4.7.3 Live mobility prerequisites

Before you move WPARs, it is good practice to verify that the departing system and arriving system have the necessary resources. Also verify that the WPAR you want to move is in a moveable state.

Because the API described in 5.5, “Compatibility API” on page 137, is not available from the CLI, in our case we must perform a similar test manually. Note the following points:

- Time on all systems must be synchronized; refer to Example 5-1 on page 107. Additionally, use **setclock** just before executing **movewpar** command.
- A System WPAR has to be Active on the departing system, and Defined on the arriving system.
- For further tests, refer to 5.5, “Compatibility API” on page 137.

### Prerequisite for system WPARS only

For system WPARS only, you must have the *same* definitions on both the Departing and Arriving systems. This is done by creating a WPAR spec-file. You can create this file using the method described in 4.7.2, “Creating WPARs eligible for Live mobility from CLI” on page 93. Use the following command:

```
mkwpar -w -o wpar.spec -e wpr01
```

Next, move the spec-file to the Arrival system (using ftp, rcp, ssh or other methods).

At the Arriving system, create the same WPAR with exactly the same definitions. Use the following command:

```
mkwpar -p -f wpar.spec
```

Now the system WPAR is ready for Live mobility.

## 4.7.4 Live mobility

If you have an application or system WPAR that is eligible for Live mobility, use the **movewpar** command as follows:

- At the Departing system, issue **movewpar** with the **-s** flag and the WPAR name. This command returns a key value.
- At the Arriving system, issue **movewpar** with the **-k** flag, the key value from command issued at the departing node, the WPAR name, and departing system IP address.

a

**Tip:** Even if you are using NTP, it is good practice to make sure time is absolutely equal on both systems prior to starting the move operation by using the **setclock** command immediately before issuing tne **move** command at the Arriving system.

Next, we show how to move WPARs that were created using the commands discussed in 4.7.2, "Creating WPARs eligible for Live mobility from CLI" on page 93.

Example 4-27 illustrates a move WPAR scenario. We move wpr01 from lpar00 (9.3.5.127) to lpar01 (9.3.5.121).

*Example 4-27 Moving a WPAR with a movewpar command*

```
hostname
lpar00
/opt/mcr/bin/movewpar -s -o /tmp/hga.log -l debug wpr01
Connection key:
48693b810000db09
mcr: Migration server started successfully for wpar wpr01
root@lpar00:/# lswpar
Name   State  Type  Hostname  Directory
-------------------------------------------
wpr01  T      S     wpr01     /wpars/wpr01
wpr03  A      S     wpr03     /wpars/wpr03
#
hostname
```

```
lpar01
setclock 9.3.5.127
/opt/mcr/bin/movewpar -k 48693b810000db09 -o /tmp/hga.log -l debug
wpr01 9.3.5.127
lswpar
Name   State  Type  Hostname  Directory
-------------------------------------------
wpr01  T      S     wpr01     /wpars/wpr01
#
lswpar
Name   State  Type  Hostname  Directory
-------------------------------------------
wpr01  A      S     wpr01     /wpars/wpr01
```

As you can see in the example, we use more switches than necessary (we added **`-o`** and **`-l`**). The **`-o`** gives a path to a logfile, and **`-l`** sets the level in the logfile. Using these extra switches, you are more likely to discover why your **`movewpar`** failed if there is an error during the process.

If you start a **`movewpar`** command and then need to stop the move process for some reason, you must use the **`movewpar`** command with the **`-x`** flag followed by the WPAR name. This command must be executed both at the departing and arriving node, as shown in Example 4-28.

*Example 4-28 Stopping the movewpar process*

```
# hostname
lpar00
# /opt/mcr/bin/movewpar -s -o /tmp/hga.log -l debug wpr01
Connection key:
48693b810000db09
mcr: Migration server started successfully for wpar wpr01
root@lpar00:/# lswpar
Name   State  Type  Hostname  Directory
-------------------------------------------
wpr01  T      S     wpr01     /wpars/wpr01
wpr03  A      S     wpr03     /wpars/wpr03
#
#
# /opt/mcr/bin/movewpar -x wpr01
mcr: Migration server for wpar wpr01 stopped
#
lswpar
```

```
Name   State  Type  Hostname  Directory
-------------------------------------------
wpr01  A      S     wpr01     /wpars/wpr01
wpr03  A      S     wpr03     /wpars/wpr03
#
```

## 4.8 Checkpointing and restarting

The checkpoint and restart features of WPARs allow the administrator to pause and resume workloads at will. This ability grants increased levels of flexibility when executing long-running jobs.

- A workload can be checkpointed at regular intervals, so that it does not need to be restarted from the beginning if there is an unanticipated failure.
- A workload can be checkpointed, thereby releasing its resources to other workloads when higher-priority applications need to run. The job can be resumed later without losing work.

The checkpointing mechanism operates at a binary level on any 32-bit or 64-bit application. No recompiling or relinking is required. When invoked, it captures the instantaneous running state (memory image) of the application and stores it to disk. All aspects of the running application, including open files, IPC pipes, shared memory, sockets, and other resources are preserved.

To be able to checkpoint a running WPAR, the WPAR must have been created with the checkpointable (**-c**) flag. This means that all writable filesystems in use by the WPAR must be stored in NFS. For more information about this topic, refer to 4.7.2, “Creating WPARs eligible for Live mobility from CLI” on page 93.

To checkpoint and kill a WPAR, run the following command from the global environment:

```
/opt/mcr/bin/chkptwpar -k -d /checkpoint_dir mywpar
```

The current state of the WPAR is dumped into the checkpoint_dir directory specified by the caller. If this is a system WPAR, this can be in the /wpars/mywpar directory.

To restart a WPAR, run the following command from the global environment:

```
/opt/mcr/bin/restartwpar -d /checkpoint_dir mywpar
```

Example 4-29 on page 98, shows how to checkpoint and restart an application WPAR called chkpt_test.

*Example 4-29 Checkpoint and restart an application WPAR*

```
root@lpar03:/# /opt/mcr/bin/chkptwpar -k -d /tmp/checkpt chkpt_test
mcr: WPAR chkpt_test was checkpointed in /tmp/checkpt.
Stopping workload partition chkpt_test.
Shutting down all workload partition processes.
Unmounting all workload partition file systems.
Checkpoint command succeeded.

root@lpar03:/# /opt/mcr/bin/restartwpar -d /tmp/checkpt chkpt_test
Starting workload partition chkpt_test.
Mounting all workload partition file systems.
Loading workload partition.
mcr: WPAR chkpt_test was restarted from /tmp/checkpt.
```

5

# Managing workload partitions

IBM Workload Partition Manager for AIX (WPAR Manager) is a tool for monitoring and managing WPARs. It will also gather information about CPU, memory, and other useful statistics. This chapter introduces and explains how to use the Workload Partition Manager for AIX.

Although you can manage your WPARs using the command line interface (CLI), this can get complicated when you want to use WPAR mobility. For information about using the CLI, refer to 4.7, “CLI walkthrough” on page 92.

**Tip:** IBM Workload Partitions Manager for AIX is an optional, separately installable licensed program product that supports more advanced features which can assist your WPAR mobility feature usage.

This chapter contains the following sections:

- WPAR Manager components and functions
- WPAR Manager installation
- WPAR agent installation
- Prepare and create mobile WPARs
- Compatibility APIs

- Performance monitoring
- WPAR groups and load management

## 5.1 WPAR Manager components and functions

WPAR Manager is a Web-based Java™ application that uses Common Agent Services (CAS) and a database. CAS is installed automatically when you install WPAR Manager.

WPAR Manager can use either Apache Derby or DB2® as a database. During WPAR Manager installation, you select which database to use. If DB2 is selected you also have to install and configure DB2, which is a separate installation process.

After the basic installation, CAS, WPAR Manager, and the database must be configured. When this is done, WPAR Manager is ready to orchestrate mobility events among servers in its purview.

For an overview explaining how WPAR Manager works and how the different components interact, refer to 5.1.1, "Common Agent Service" on page 101. To install WPAR Manager, skip 5.2, "WPAR Manager installation" on page 106.

### 5.1.1 Common Agent Service

To help you understand how WPAR Manager components work, this section provides a brief overview of the Common Agent Services (CAS) framework. Figure 5-1 on page 102 shows WPAR Manager components, including the management server and two managed servers.

In this scenario:

- WPAR Manager is the application and user interface. It is in charge of communicating with the user and updating the database.
- CAS Agent Manager is in charge of communication between the WPAR agent and the WPAR Manager in the LPAR.
- The database is the repository for all WPAR definition data.

There are WPARs running on the managed servers. B and G are set up as system WPARs, and W is set up as an application WPAR.

In this setup, the WPAR Manager should be able to move WPARs W and G between the two nodes. For a detailed explanation, refer to 5.4, "Preparing and creating mobile WPARs" on page 118.

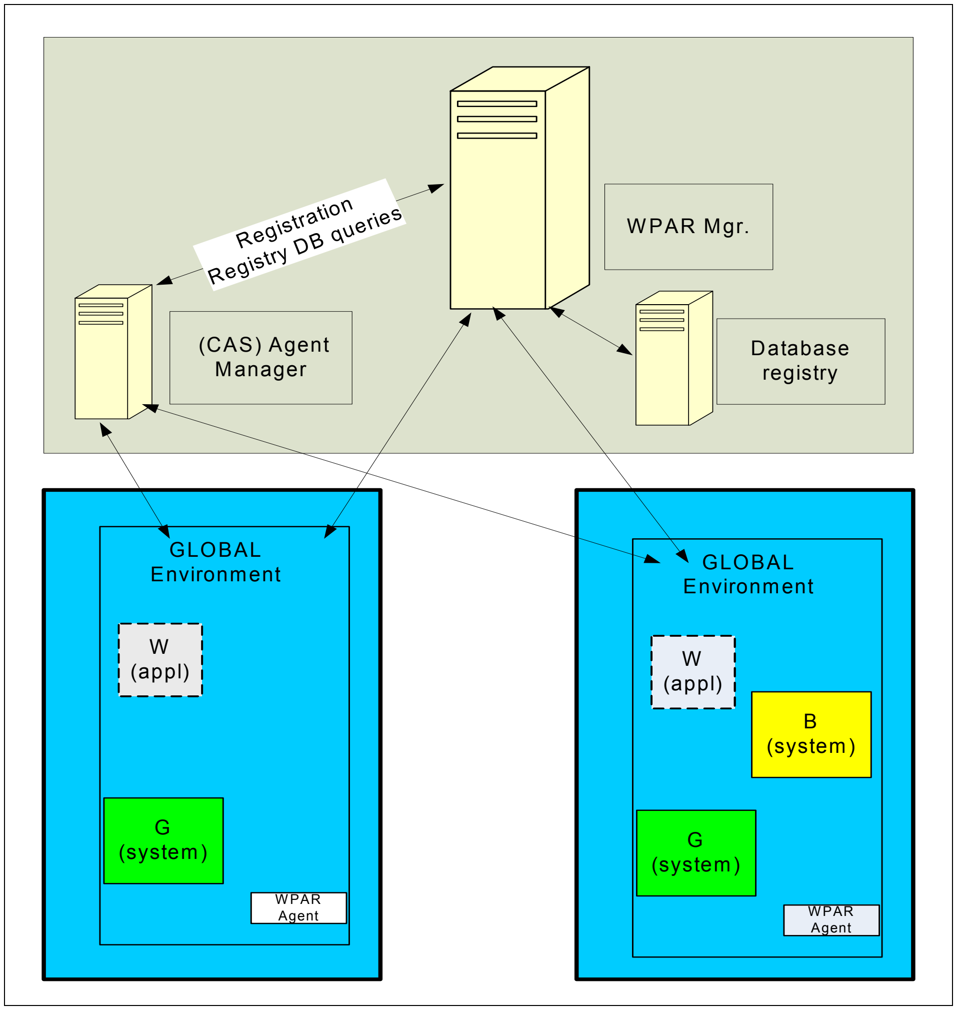


*Figure 5-1 Components of WPAR Manager*

Deploying WPAR Management software usually requires a server that hosts the management software and an agent that has to be installed on each server to be managed, as shown in Figure 5-1.

The effort required to install, configure, maintain, and monitor different agent implementations can become a very time-consuming management task. In addition, the number of redundant functions provided by each agent implementation (for example, listening to ports, the runtime daemon, and memory consumption) leads to ineffective usage of the system resources.

The goal of Common Agent Services is to minimize the complexity of the software management deployment by reducing the effort needed for deployment and utilizing system resources more effectively.

This is accomplished by providing a common framework that can be reused by various management applications that are deployed across the enterprise.

This Common Agent Services framework consists of the following components:

- The Agent Manager
- The Resource Manager
- The Common Agent

## 5.1.2 Agent Manager

**Note:** The Agent Manager component implements the CAS services in the WPAR Manager application. This agent talks to the Common Agent on the managed systems.

The Agent Manager is the server component of the Common Agent services. It provides authentication and authorization services, enables secure connections between managed systems in your deployment, and maintains the registry about the managed systems and the software running on those systems. It also handles queries from the resource managers against the database.

The Agent Manager has the following components:

- The Agent Manager service

  The Agent Manager service serves as a certificate and registration authority to provide authentication and authorization using X.590 digital certificates and the Secure Socket Layer (SSL) protocol. It also handles requests for registry information from Common Agents and resource managers.

  Resource managers and Common Agents must register with the Agent Manager service before they can use its services to communicate with each other. This registration is password-protected and there are separate passwords for the Common Agents and resource managers

  **Note:** For WPAR Manager, you only need to specify the registration password for the Common Agents. The password for the resource manager is automatically generated during the configuration of WPAR Manager.

- The registry

  The registry is the database that contains the current configuration of all known Common Agents and resource managers.

  Information contained in the registry includes:

  - The identity, digital certificates, and communication information for each resource manager
  - The identity, digital certificates, and communication information for each Common Agent
  - Basic configuration information for each Common Agent (for example, hardware type and operating system version)
  - The status of each Common Agent
  - The last error or, optionally, a considerable number of errors, reported by each Common Agent

  The registry information is updated by events, such as the registration of Common Agents and resource managers, and by periodic updates from the Common Agents.

  **Note:** For WPAR Manager, the Agent Manager uses an Apache Derby database to implement the registry. This Derby database is included within the lightweight runtime fileset that is delivered with WPAR Manager.

- The agent recovery service

  The agent recovery service provides error logging for Common Agents that cannot communicate with other Agent Manager services.

  WPAR Manager does not make use of this feature.

  **Note:** The agent manager functionality for WPAR Manager is in the fileset wparmgt.cas.agentmgr.

  CAS Agent Manager listens to communication from WPAR Agent and WPAR Manager on ports 9511, 9512, and 9513. These are default ports that can be overridden by the user during the configuration of WPAR Manager.

## 5.1.3 Resource Manager

A *resource manager* is management software that uses the services of the agent manager to communicate securely with, and obtain information about, the managed systems running the Common Agent software. The resource manager uses the services of the Common Agent to deploy and run its software on the managed systems.

Note that in our case, the WPAR Manager component of WPAR Manager is the resource manager of the CAS framework.

**Note:** The resource manager functionality for WPAR Manager is in the fileset wparmgt.mgr.rte. This fileset also includes the functionality of the WPAR Management console.

WPAR Manager listens to ports 14080 and 14443 and communicates to port 9510 on the WPAR Agent. WPAR Manager listens to ports 9511, 9512, and 9513 on the CAS Agent Manager. These are default ports that can be overridden by the user during the configuration of WPAR Manager.

## 5.1.4 Common Agent

Each managed system in your deployment runs a *Common Agent*, which provides a single implementation of common services that can be used by all management products.

The Common Agent consists of:

- Common Agent services
- Product-specific subagent (for example, for WPAR Manager this is the piece of code for WPAR Agent)

For instance, if two management products (A and B) manage the same system, the Common Agent is installed only once on that system and it will run two product-specific subagents: one for product A and one for product B.

The Common Agent provides the following functions:

- A single set of security credentials and a common security infrastructure for all management applications
- Automated management of security credentials

  The Common Agent certificate is automatically renewed when it is near the expiration date.

- Common agent health monitoring and configuration monitoring services

  The Common Agent has a “heartbeat” function that sends periodic status and configuration reports to the agent manager. It also allows any subagent to participate and to provide application-specific status.

  The frequency of this update can be set, or it can be turned off. Management applications can register to the agent manager to receive these updates.

The registry contains only the most recent configuration update. By default, only the most recent status information is saved for each Common Agent. The retention period is configurable.

**Note:** The Common Agent functionality for WPAR Manager is in the fileset wparmgt.cas.agent.

The product-specific subagent, WPAR Agent, functionality for WPAR Manager is in the fileset wparmgt.agent.rte.

WPAR Agent listens to port 9510 and communicates to port 9511, 9512, and 9513 on the CAS Agent Manager. These are default ports that can be overridden by the user during the configuration of WPAR Manager.

## 5.2 WPAR Manager installation

WPAR Manager can be installed in any LPAR. The requirement is shown in Table 5-1 on page 107.

Before you start installation, you have to decide if you want to use the default Apache Derby database or a DB2 database. A rule of thumb says if you have 25 or fewer WPARs to administer, then select the Apache Derby database.

If you start with the default Apache Derby database, you can later migrate to DB2. However, if you start with DB2, no migration to the Apache Derby database exists.

**Note:** Using DB/2 will require extra system resources and planning.

If your choose DB2, it does not need to be on the same LPAR as the WPAR Manager. You can use an existing DB2, or install DB2 in a separate LPAR.

In our installation we used the default Apache Derby database. We also show the commands to install and configure DB2.

Table 5-1lists the disk and memory requirements for WPAR Manager.

*Table 5-1 Disk and memory requirements for WPAR Manager*

| **Application** | **Memory requirement** | **Disk Space requirement** |
|---|---|---|
| WPAR Manager including Apache Dearby database | 125Mb | /home 800 MB<br>/opt 700 MB<br>/tmp 175MB<br>/var 200 MB |
| Optional<br>DB2 server | 256Mb | 500 Mb minimum<br>2 Gb recommended<br>/home, 200 -1500 Mb<br>//opt, 500 Mb |
| Agent Manager | 50 Mb when idle | 30 Mb |
| WPAR agent | 45 Mb when idle | 27 Mb |

At this point you can install your WPAR Manager and WPAR agents.

To complete WPAR Manager installation, follow these steps:

1. Check prerequisites.
2. Install WPAR Manager file sets.
3. Configure WPAR Manager.
4. Install WPAR agent in all nodes that you want to manage.
5. Configure WPAR agents.
6. Verify installation.

**Tip:** If you plan to use mobility, you have to make sure that the time on all involved systems are in sync. This is normally accomplished by using an NTP server.

Example 5-1 shows how to set up an NTP server.

*Example 5-1 Setting up NTP server*

```
9.3.5.1.99 is the server we want to be our NTP sever. On the LPAR you
want to act as NTP server update /etc/ntp.conf with content like this.
driftfile /etc/ntp.drift
tracefile /etc/ntp.trace
restrict default nomodify notrust
restrict 9.3.5.199
restrict 127.0.0.1

On all the Client update /etc/ntp.conf
server 9.3.5.199
broadcastclient
driftfile /etc/ntp.drift
tracefile /etc/ntp.trace
ntpdate 9.3.5.199

Then start ntp daemon on all systems using command:
startsrc -s xntpd

Edit /etc/rc.tcpip to allow ntp start at boot time, remove “#” so the
line looks like this:
# Start up Network Time Protocol (NTP) daemon
start /usr/sbin/xntpd "$src_running"
```

Now you are able to install and exploit WPAR Manager.

### 5.2.1 Check WPAR Manager prerequisites

The prerequisites for WPAR Manager are listed here. Verify them before you start installing WPAR Manager and agents.

- IBM AIX 6.1
  - Use oslevel -r to check.
- Java Version 5
  - Use **`lslpp -lq ‘Java5*’`** to verify.
- Space requirement
  - This requirement depends on the database you want to use; refer to Table 5-1 on page 107 for guidance.

### 5.2.2 WPAR Manager installation and configuration

Before you start installing WPAR Manager, ensure that you do not have any old versions installed on your system. This can be done with the following command:

```
installp -ug wparmgt.mgr wparmgt.db wparmgt.cas.agentmgr  wparmgt.lwi
```

This chapter assumes that you have already copied your software from WPAR Manager CD/DVD to a local directory “/source,” and that this directory is your working directory. Note that you do not need to copy the files to the LPAR. Installation can also be performed from a CD, DVD. or NFS mount.

At this point, you must decide whether to use Apache Derby database or DB2.

To use Apache Derby, install:

```
# installp -acXYgd . wparmgt.mgr
```

To use DB2, use:

```
# installp -acXYgd . wparmgt.mgr wparmgt.db
```

This will install these filesets on your WPAR Manager machine:

- wparmgt.mgr.rte 1.2.0.0
- wparmgt.cas.agentmgr 1.4.1.0
- tivoli.tivguid 1.3.0.0
- wparmgt.db.db2 1.2.0.0 (For DB2)
- wparmgt.lwi  1.2.0.0

And, if you select DB2:

- wparmgt.db.db 1.2.0.02

You have to be at or above these fileset levels.

If you select DB2, you must install and configure DB2 before continuing to WPAR Manager configuration. DB2 is embedded on the WPAR Manager CD/DVD. Use this command to install it:

```
# /opt/IBM/WPAR/manager/db/bin/DBInstall.sh -dbinstallerdir /source/db2
-dbpassword <db2_instance_password>
```

Replace <db2_instance_password> with the 8-character or less password for the DB2 instance.

Providing more detailed information about DB2 installation and configuration is beyond the scope of this book.

You are now ready to configure your WPAR Manager. The configuration can be done in console mode (txt) or graphical mode.

**Tip:** If you select graphical mode (GUI), configure your workstation to use a graphic console or any Xdisplay client tool (for example, Xwindow, Hummingbird, and so on) to set the necessary environment variables (export DISPLAY).

For graphical mode, enter:

```
# /opt/IBM/WPAR/manager/bin/WPMConfig.sh
```

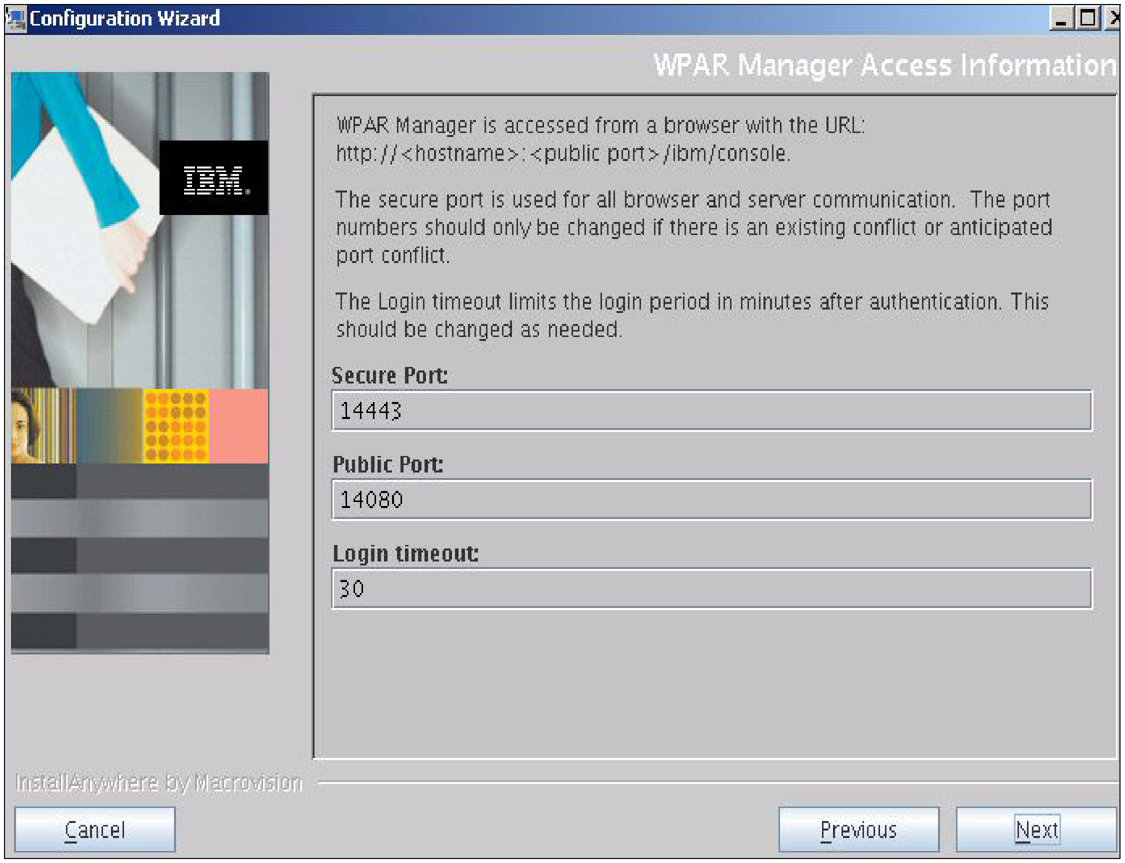
For console (txt) mode, enter:

```
# /opt/IBM/WPAR/manager/bin/WPMConfig.sh -i
```

We selected graphical mode for our configuration.

During configuration, you must make some choices. Some of these choices will be used when installing the configuring the agent, and others will be used when logging on to WPAR Manager.

The first important choices to make are shown in Figure 5-2. Here you select the port numbers to use when later logging on to WPAR Manager. If you are comfortable with the default values, click **Next**.



WPAR Manager Access Information

WPAR Manager is accessed from a browser with the URL: http://<hostname>:<public port>/ibm/console.

The secure port is used for all browser and server communication. The port numbers should only be changed if there is an existing conflict or anticipated port conflict.

The Login timeout limits the login period in minutes after authentication. This should be changed as needed.

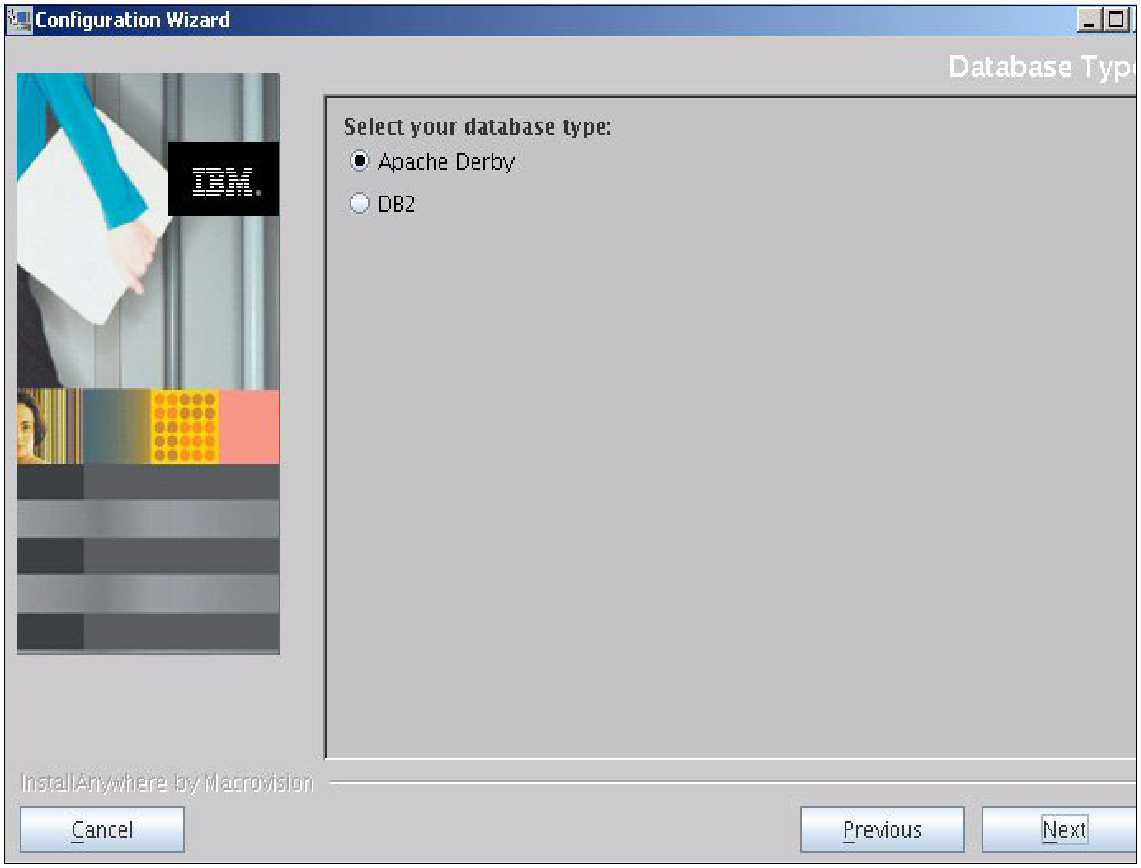
Secure Port: 14443

Public Port: 14080

Login timeout: 30

*Figure 5-2 Port selection*

Now you make your database choice; see Figure 5-3.



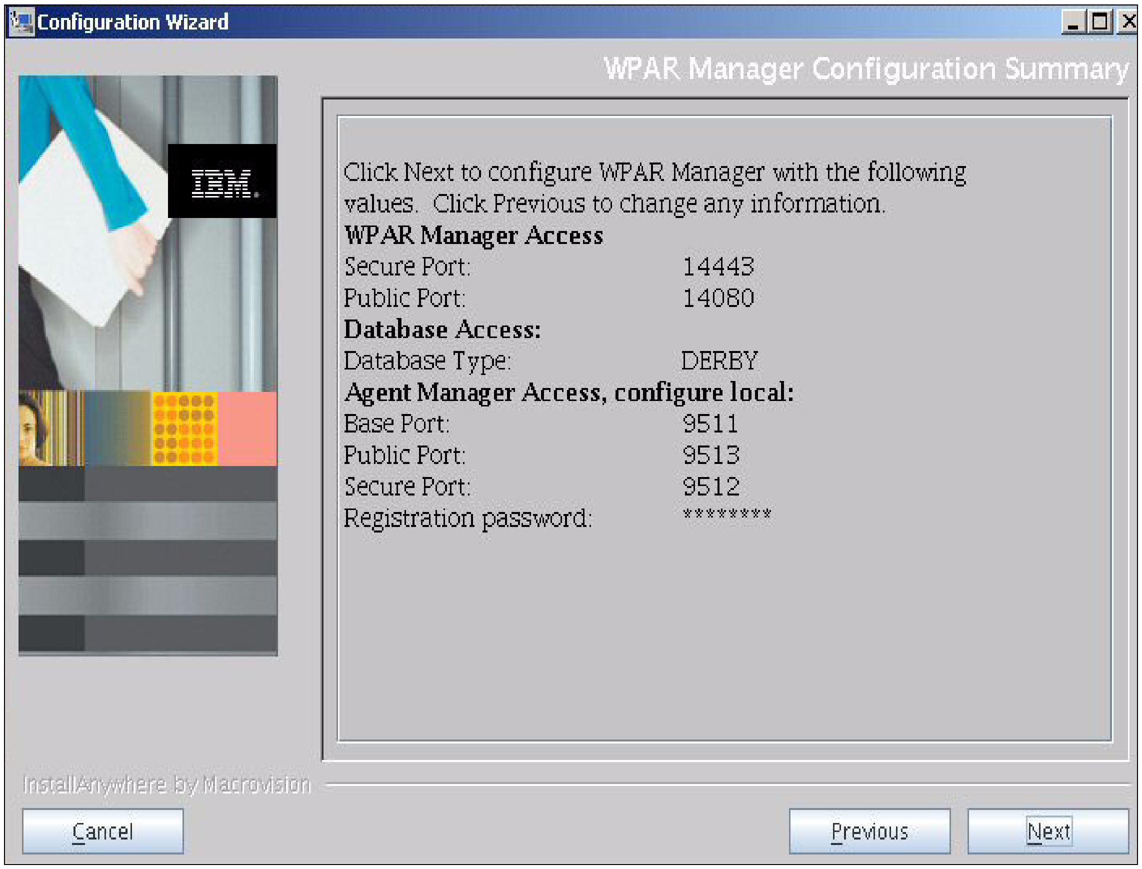
Select your database type:

(●) Apache Derby

( ) DB2

*Figure 5-3   Database selection*

There are other selections to make, such as port number and password for agents, which are not shown here. When you are finished making your selections, you will see a summary window that displays the choices you made (default values can be an ideal choice). The WPAR Manager Configuration Summary is shown in Figure 5-4 on page 112.


Configuration Wizard
WPAR Manager Configuration Summary
InstallAnywhere by Macrovision
Cancel
Previous
Next

Click Next to configure WPAR Manager with the following values. Click Previous to change any information.

**WPAR Manager Access**

| | |
|---|---|
| Secure Port: | 14443 |
| Public Port: | 14080 |

**Database Access:**

| | |
|---|---|
| Database Type: | DERBY |

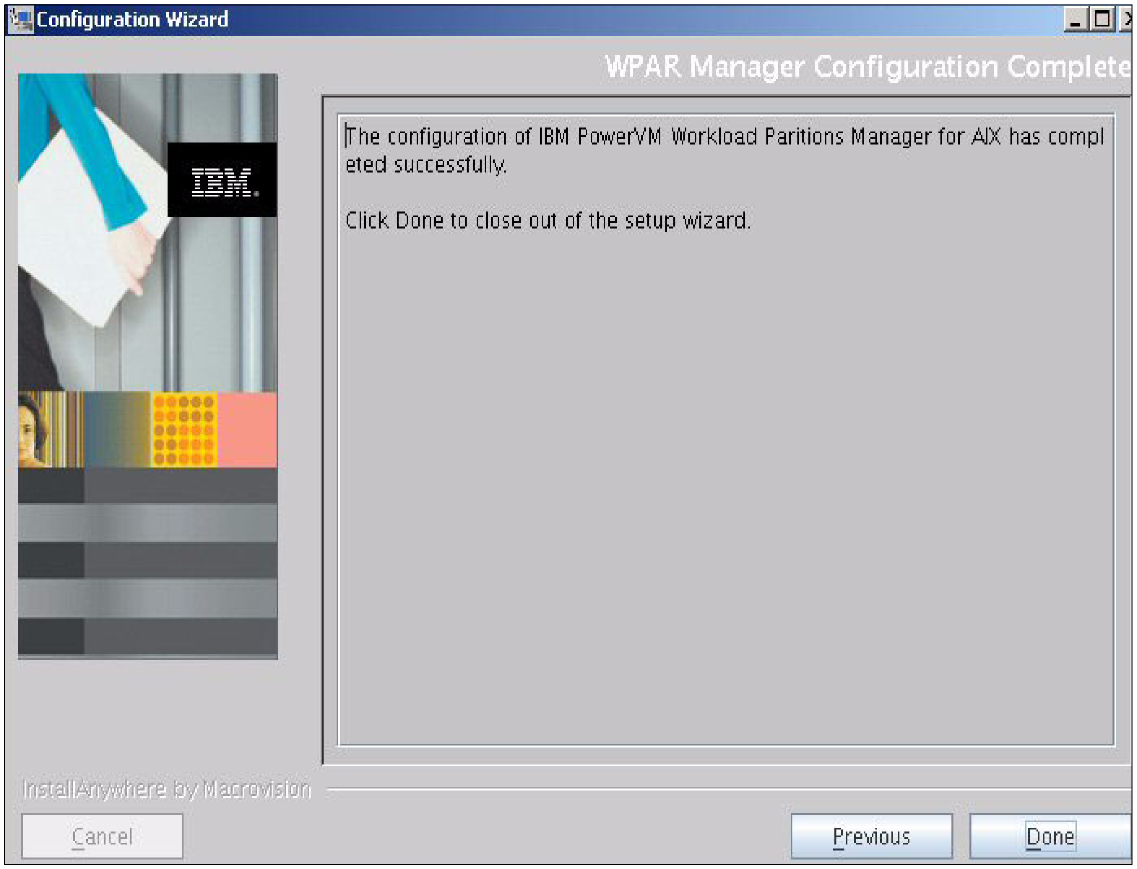
**Agent Manager Access, configure local:**

| | |
|---|---|
| Base Port: | 9511 |
| Public Port: | 9513 |
| Secure Port: | 9512 |
| Registration password: | ******** |

*Figure 5-4 WPAR Manager summary, before installation start*

**Tip:** If you select port numbers other than the default ports suggested by the configuration tool, it is useful to record what they are because you will need them again when you install the agents. Also retain the agent password you selected.

After your installation starts, you will see the WPAR Manager Configuration Complete window showing that your configuration completed successfully; see Figure 5-5 on page 113.


Configuration Wizard
WPAR Manager Configuration Complete
InstallAnywhere by Macrovision
Cancel
Previous
Done

The configuration of IBM PowerVM Workload Paritions Manager for AIX has compl eted successfully.

Click Done to close out of the setup wizard.

*Figure 5-5 WPAR Manager installation is completed*

If you had selected console (text) mode installation, the look would have been different, but you would have to enter the same information.

## 5.2.3 Verifying WPAR Manager installation

Before installing your agents, verify that WPAR Manager is installed properly and is working.

To check that the WPAR Manager installation is completed:

- ▶ Verify that WPAR Manager daemon is active.
- ▶ Verify that CAS Agent Manager daemon is active.
- ▶ Make sure it is possible for a Web browser to connect to both of them.

For the WPAR Manager daemon:

- To verify, use **`/opt/IBM/WPAR/manager/bin/wparmgr status`**
- To start, use **`/opt/IBM/WPAR/manager/bin/wparmgr start`**
- To stop, use **`/opt/IBM/WPAR/manager/bin/wparmgr stop`**

For the CAS Agent Manager daemon:

- To verify, use **`/opt/IBM/WPAR/manager/bin/agentmgr status`**
- To start, use **`/opt/IBM/WPAR/manager/bin/agentmgr start`**
- To stop, use **`/opt/IBM/WPAR/manager/bin/agentmgr stop`**

**Note:** Both WPAR Manager and CAS Agent Manager are configured to autostart from inittab, so normally you do not need to start them after rebooting the system.

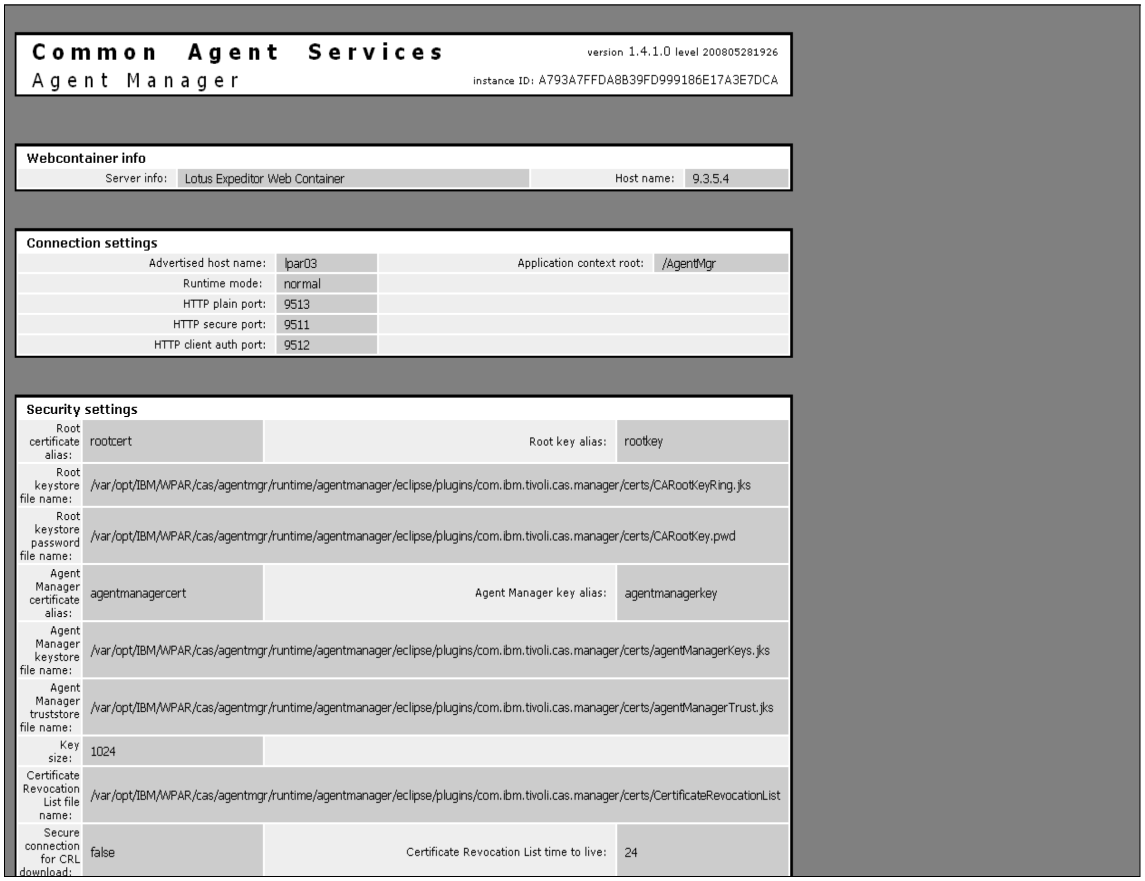
Use the Web browser to verify the installation by testing that it can connect to both CAS Agent Manager and WPAR Manager.

To verify the connection to the CAS Agent Manager, go to:

`http://<WPARManagerhostname>:9513/AgentMgr/Info`

This URL verifies that the CAS Agent Manager is running. If it is not running, you will get a message similar: to `Cannot establish a connection to the server at 9.3.5.4:9513`.

When it is working, you will see a display similar to Figure 5-6 on page 115.



**Common Agent Services**
Agent Manager

version 1.4.1.0 level 200805281926
instance ID: A793A7FFDA8B39FD999186E17A3E7DCA

**Webcontainer info**

| | | | |
|---|---|---|---|
| Server info: | Lotus Expeditor Web Container | Host name: | 9.3.5.4 |

**Connection settings**

| | | | |
|---|---|---|---|
| Advertised host name: | lpar03 | Application context root: | /AgentMgr |
| Runtime mode: | normal | | |
| HTTP plain port: | 9513 | | |
| HTTP secure port: | 9511 | | |
| HTTP client auth port: | 9512 | | |

**Security settings**

| | | | |
|---|---|---|---|
| Root certificate alias: | rootcert | Root key alias: | rootkey |
| Root keystore file name: | /var/opt/IBM/WPAR/cas/agentmgr/runtime/agentmanager/eclipse/plugins/com.ibm.tivoli.cas.manager/certs/CARootKeyRing.jks | | |
| Root keystore password file name: | /var/opt/IBM/WPAR/cas/agentmgr/runtime/agentmanager/eclipse/plugins/com.ibm.tivoli.cas.manager/certs/CARootKey.pwd | | |
| Agent Manager certificate alias: | agentmanagercert | Agent Manager key alias: | agentmanagerkey |
| Agent Manager keystore file name: | /var/opt/IBM/WPAR/cas/agentmgr/runtime/agentmanager/eclipse/plugins/com.ibm.tivoli.cas.manager/certs/agentManagerKeys.jks | | |
| Agent Manager truststore file name: | /var/opt/IBM/WPAR/cas/agentmgr/runtime/agentmanager/eclipse/plugins/com.ibm.tivoli.cas.manager/certs/agentManagerTrust.jks | | |
| Key size: | 1024 | | |
| Certificate Revocation List file name: | /var/opt/IBM/WPAR/cas/agentmgr/runtime/agentmanager/eclipse/plugins/com.ibm.tivoli.cas.manager/certs/CertificateRevocationList | | |
| Secure connection for CRL download: | false | Certificate Revocation List time to live: | 24 |

*Figure 5-6 CAS Manager installed and running*

To verify the connection to the WPAR Manager, go to:

`http://<WPARManagerhostname>:14080/ibm/console`

Or go to:

`https://<WPARManagerhostname>:14443/ibm/console`

These URLs verify that the WPAR Manager is running. If it is not running, you will get a message similar to: `Cannot establish a connection to the server at` <*WPAR Manager hostname*>`:14443.`

If WPAR Manager is working, you will reach the WPAR Manager login screen.

## 5.3 WPAR agent installation

On every node where you want the WPAR Manager to manage the WPARs, you have to install the WPAR agent.

Assume, in all examples, that you have already copied your WPAR agent files to /source, and that /source is your working directory. Only these filesets are needed on your managed nodes

- wparmgt.agent
- cas.agent
- tivoli.tivguid
- wparmgt.license

You do not need to copy the files to the LPAR. Installation can also be performed from CD/ DVD or NFS mount.

### 5.3.1 Checking WPAR agent prerequisites

Verify these WPAR are installed.

#### WPAR agent prerequisites

- IBM AIX 6.1
- bos.wpars 6.1.0.0
- Java5.sdk 1.5.0.0
- perfagent.tools 6.1.0.0

The three filesets are generally installed by default.

### 5.3.2 Installing and configuring WPAR agent

Before you install and configure WPAR agent, ensure that you do not have any old software installed by entering:

```
# installp -ug cas.agent wparmgt.agent
# installp -acXYgd . wparmgt.agent
```

Install all necessary agent filesets:

```
# installp -acXYgd . wparmgt.agent
```

This installs the following filesets on your system:

- wparmgt.agent.rte 1.2.0.0
- cas.agent 1.4.1.0

- tivoli.tivguid 1.3.0.0
- mcr.rte 4.2.0.0

You must be at or above the fileset levels shown.

Now that you have installed the agent you must configure it so that it will register to WPAR Manager. Run the post-install configuration tool:

```
# /opt/IBM/WPAR/agent/bin/configure-agent -amhost
<agent_manager_hostname> -passwd <agent_registration_password>
```

Note the following:

- agent_manager_hostname is the hostname (FQDN) of the LPAR where you installed WPAR Manager.
- agent_registration_password is the password you selected during WPAR Manager installation; see 5.2, "WPAR Manager installation" on page 106.

> **Tip:** The hostname of both the manager and the client should be resolvable through DNS, or should have entry in the /etc/hosts file on all machines.

Example 5-2 illustrates how to configure the WPAR agent.

*Example 5-2 Configuring WPAR agent*

```
/opt/IBM/WPAR/agent/bin/configure-agent -amhost lpar03.austin.ibm.com -passwd abc1234
```

### 5.3.3 Verifying agent installation

After a successful registration, you should have these files in the /opt/IBM/WPAR/agent/cas/runtime/agent/cert directory:

- agentKeys.jks
- agentTrust.jks
- amRevocationList.crl
- pwd

If they do not exist (after allowing a few minutes for registration to complete), examine the log files on the Agent Manager in the /var/opt/tivoli/ep/logs directory to see if any registration errors have been logged.

To check that WPAR Agent installation is completed:

- Verify that WPAR Agent daemon is active.
- Verify that the WPAR Management console can discover the server on which the WPAR Agent has been installed.

For WPAR Agent daemon:

- To verify, use **`/opt/IBM/WPAR/agent/bin/wparagent status`**
- To start, use **`/opt/IBM/WPAR/agent/bin/wparagent start`**

To stop, use **`/opt/IBM/WPAR/agent/bin/wparagent stop`**

Finally, you must reboot the agent LPARs.

**Note:** WPAR Agent is started through the lwi subsystem. The lwi subsystem is started from inittab (entry pconsole), so normally you do not need to start it after rebooting the system.

After registration, from the WPAR Management console you can start to *discover* the managed system (that is, that its agent has been installed and configured) and start to manage it from the WPAR Management console.

To discover the newly registered managed system:

- Point your browser to `http://lpar03.austin.ibm.com:14080/ibm/console`
- Enter the user name and password.
- Select **Resource Views** -> **Managed Systems** -> **Discover**.

Figure 5-8 on page 122 shows the result after Discover was run. Notice that all tree clients that we installed the WPAR Agent are now visible to the WPAR Manager. Therefore, it is possible for the WPAR Manager to manage the WPAR on these systems from now on.

## 5.4 Preparing and creating mobile WPARs

After your WPAR Manager is installed, you can start to create your WPARs. The following sections explain how you can use WPAR Manager for:

- Creating an application WPAR eligible for static mobility
- Creating an application WPAR eligible for live mobility
- Creating a system WPAR eligible for static mobility
- Creating a system WPAR eligible for live mobility

Note that the application used here is the same application as used in 4.1.1, “Creating and managing simple WPARs” on page 56.

## 5.4.1 Preparing your NFS server

In addition to your WPAR Manager software, note that you must have data on a shared environment in order to make a WPAR eligible for live mobility. This entails using an NFS server; refer to 2.6, “Mobility” on page 31 for detailed information about that topic.

When setting up shares on the NFS server, you can choose to create one share, and then make directories in this share one directory for each WPAR you need. Alternatively, you can create a share for each WPAR you want to create. Which method to use is up to you, but you might want to keep security considerations in mind when making that decision. Refer to 6.2, “File system isolation and security” on page 151 for more information about that topic.

**Tip:** When creating shares on the NFS server, keep in mind that all LPARs where you want your WPAR to run, as well as the WPAR itself, must have root access to the actual share.

## 5.4.2 The mobility process

This section explains the terminology and conditions used when discussing mobile WPARs.

- Departure system - this is the system where your WPAR is running before you start moving it.
- Arrival system - this is the managed server to which you want to move your system.

The mobility modes available are static and live:

- Static mobility is stop, backup, and restore (for application WPARs, it is stop and redeploy).
- Live mobility is where your application is suspended (but not stopped) during the move process, and where you have data on a remote server (NFS-server). This is the most advanced scenario and is also the most complex to set up.

During live mobility, your WPAR can be seen in the following states:

- Paused
- Frozen
- Loaded
- Moving

The Paused, Frozen and Loaded states are rarely used. They generally appear only in High Performance Computing (HPC) WPARs where your WPAR is managed by checkpoint/restart.

Frozen and Loaded states are used in the original relocation method based on checkpoint/restart. This method is still available to be used, but it is slower than the asynchronous mobility feature. We expect there will only be a need to use this method in HPC, PowerHA™, or other specific environments with unique operational requirements.

Moving is the state when a WPAR is changing location from one managed system to another. As a part of the moving process, the WPAR Manager verifies both whether your WPAR is eligible for moving, and whether your arrival system is capable of receiving the WPAR.

The following server types are used: WPAR Manager, NFS server, and Managed System, as explained here.

- WPAR Manager is the LPAR where your WPAR Manager is installed.
- NFS server is the server where the WPAR that is eligible for live mobility must have its file systems; this is also referred to as your “remote server”.
- Managed System refers to the LPARs where you create your WPARs. The managed server has the WPAR agent installed, so it is “controllable” from WPAR Manager. The managed system is often referred to as the “global environment.

## 5.4.3 Logging on to WPAR Manager

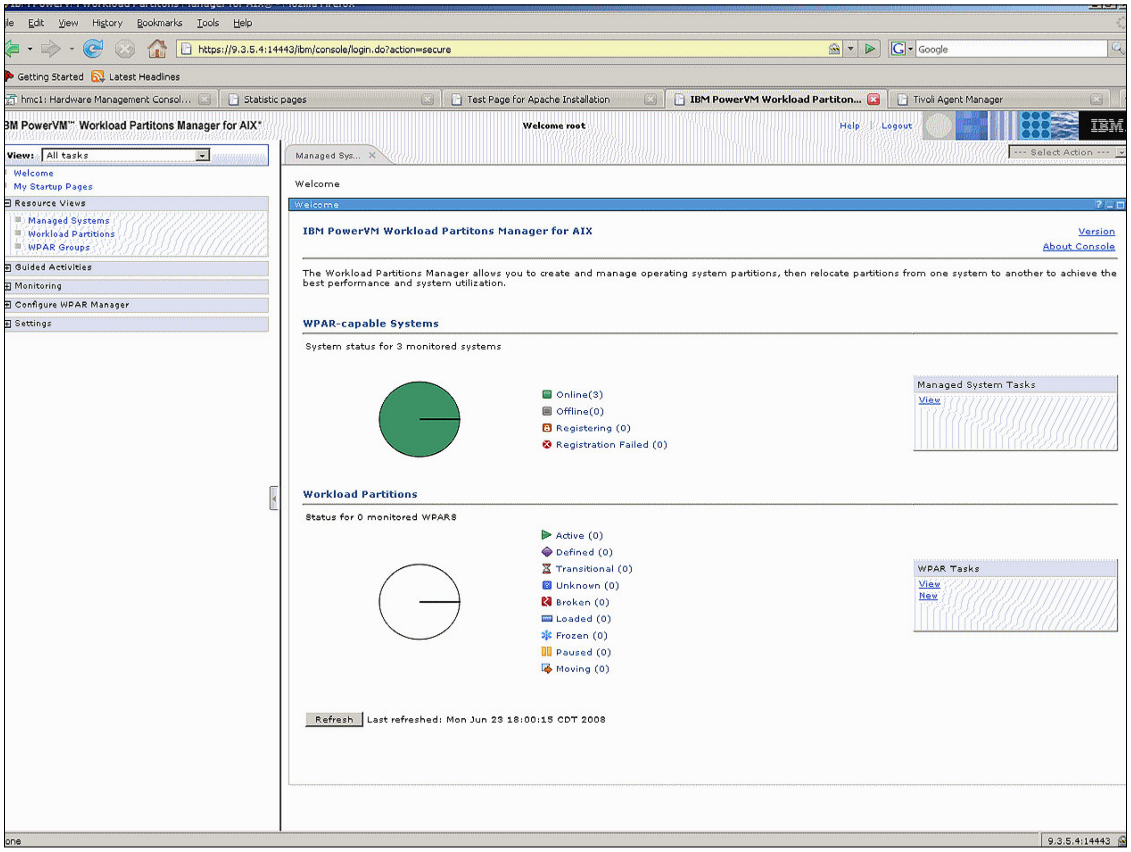
The next step is to logon to your WPAR Manager. As you recall, during installation you selected port numbers for the WPAR Manager, so for login start a browser and fill out the address and port you selected earlier:

```
https://9.3.5.4:14443/ibm/console/logon.jsp
```

Note that you can use hostname instead of IP address; just ensure that the name can be resolved by DNS or /etc/hosts.

At the logon screen, logon with the same user and password that installed your WPAR Manager.

Figure 5-7 on page 121 shows the first window you will see after logging in. The left part is the navigation window. The right part is where you do the work and obtain information.


File Edit View History Bookmarks Tools Help
https://9.3.5.4:14443/ibm/console/login.do?action=secure
Getting Started Latest Headlines
hmc1: Hardware Management Consol... Statistic pages Test Page for Apache Installation IBM PowerVM Workload Partiton... Tivoli Agent Manager
IBM PowerVM™ Workload Partitons Manager for AIX®
Welcome root
Help Logout
View: All tasks
Welcome
My Startup Pages
Resource Views
Managed Systems
Workload Partitions
WPAR Groups
Guided Activities
Monitoring
Configure WPAR Manager
Settings
Managed Sys...
--- Select Action ---
Welcome
Version
About Console
Managed System Tasks
View
WPAR Tasks
View
New
Refresh

IBM PowerVM Workload Partitons Manager for AIX

The Workload Partitions Manager allows you to create and manage operating system partitions, then relocate partitions from one system to another to achieve the best performance and system utilization.

WPAR-capable Systems

System status for 3 monitored systems

- 🟩 Online(3)
- ▨ Offline(0)
- Registering (0)
- Registration Failed (0)

Workload Partitions

Status for 0 monitored WPARS

- Active (0)
- Defined (0)
- Transitional (0)
- Unknown (0)
- Broken (0)
- Loaded (0)
- Frozen (0)
- Paused (0)
- Moving (0)

Last refreshed: Mon Jun 23 18:00:15 CDT 2008

Figure 5-7   Welcome window in WPAR Manager

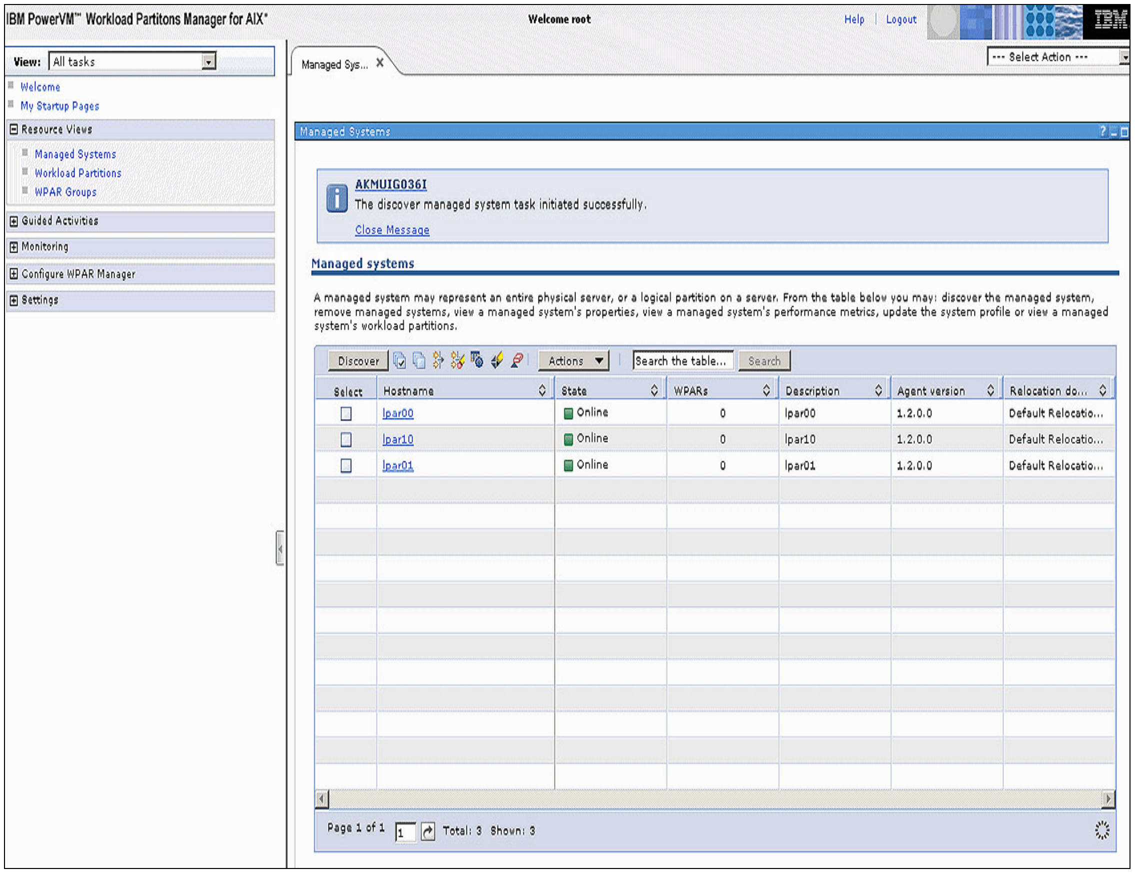
WPAR Manager works like any other browser or Windows® application. Clicking the plus (+) sign opens a menu. Clicking the minus (-) sign closes a menu. After you open the menus, simply click the one you want to use.

First, create other WPAR Manager users. Select **Settings** -> **Console User Authority**, and click **Add**. Next, select the role for this user and click **OK**.

**Tip:** Before you create a user in WPAR Manager, that user must first exist as an AIX user. No superuser or admin privilege is needed for the user at the AIX level.

Figure 5-8 on page 122 shows how many managed systems exist. In our case, we selected Resource Views and then Managed Systems. At this point our system had three managed systems: lpar00, lpar01, and lpar10. They represented the systems where we installed our WPAR agent, and the systems where our WPAR Manager can control WPARs.

Our NFS server in this test was lpar02, and we did not install WPAR agent software in this LPAR.


IBM PowerVM™ Workload Partitons Manager for AIX® — Welcome root — Help | Logout

View: All tasks

- Welcome
- My Startup Pages
- Resource Views
  - Managed Systems
  - Workload Partitions
  - WPAR Groups
- Guided Activities
- Monitoring
- Configure WPAR Manager
- Settings

Managed Sys... ×    --- Select Action ---

Managed Systems

AKMUIG036I
The discover managed system task initiated successfully.
Close Message

Managed systems

A managed system may represent an entire physical server, or a logical partition on a server. From the table below you may: discover the managed system, remove managed systems, view a managed system's properties, view a managed system's performance metrics, update the system profile or view a managed system's workload partitions.

Discover | Actions | Search the table... | Search

| Select | Hostname | State | WPARs | Description | Agent version | Relocation do... |
|---|---|---|---|---|---|---|
| ☐ | lpar00 | 🟩 Online | 0 | lpar00 | 1.2.0.0 | Default Relocatio... |
| ☐ | lpar10 | 🟩 Online | 0 | lpar10 | 1.2.0.0 | Default Relocatio... |
| ☐ | lpar01 | 🟩 Online | 0 | lpar01 | 1.2.0.0 | Default Relocatio... |

Page 1 of 1 | 1 | Total: 3 Shown: 3

*Figure 5-8 Managed System overview*

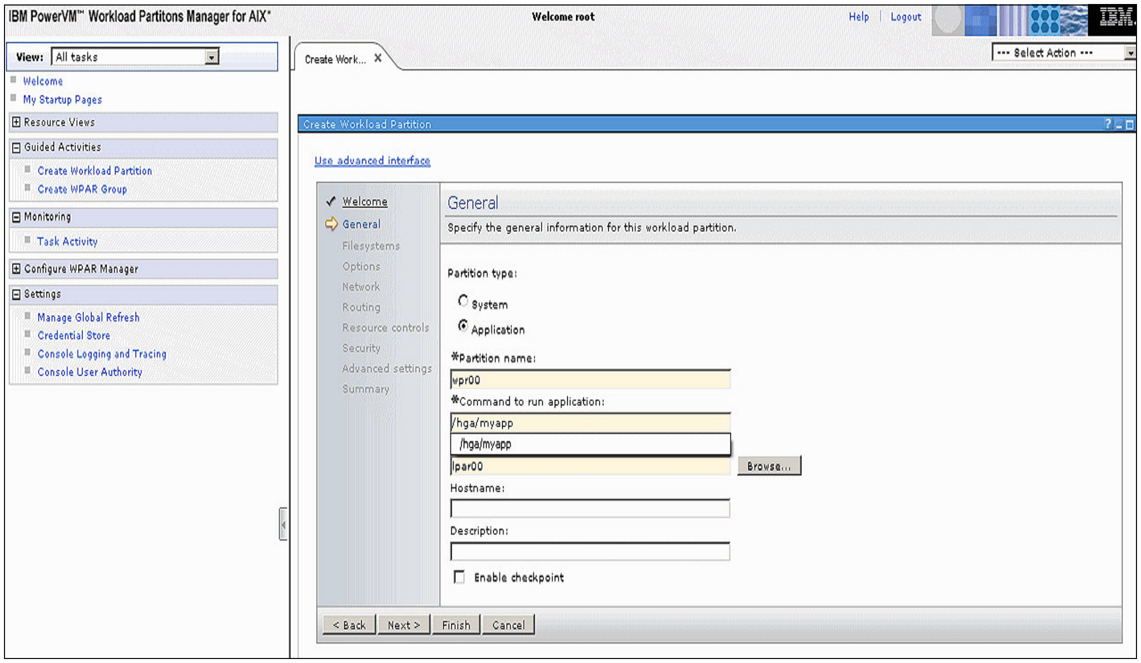
The following sections only show the WPAR Manager screens that illustrate how easy it is to use WPAR Manager.

## 5.4.4 Creating an application WPAR eligible for static mobility

Now you create your first WPAR using WPAR Manager. From the main window select **Guided Activities** -> **Create Workload Partition**. Mark the Application selection. Figure 5-9 on page 123 shows the Create WPAR main window.

Fill in the required fields, which are marked with an (*) asterisk:

- Partition name: The WPAR name you want
- Command line to run application: the full PATH to our application
- Managed System: the name of the managed system where you want your WPAR to run



General

Specify the general information for this workload partition.

Partition type:
System
[✓ Application]

*Partition name: wpr00

*Command to run application: /hga/myapp

/hga/myapp

lpar00

Hostname:

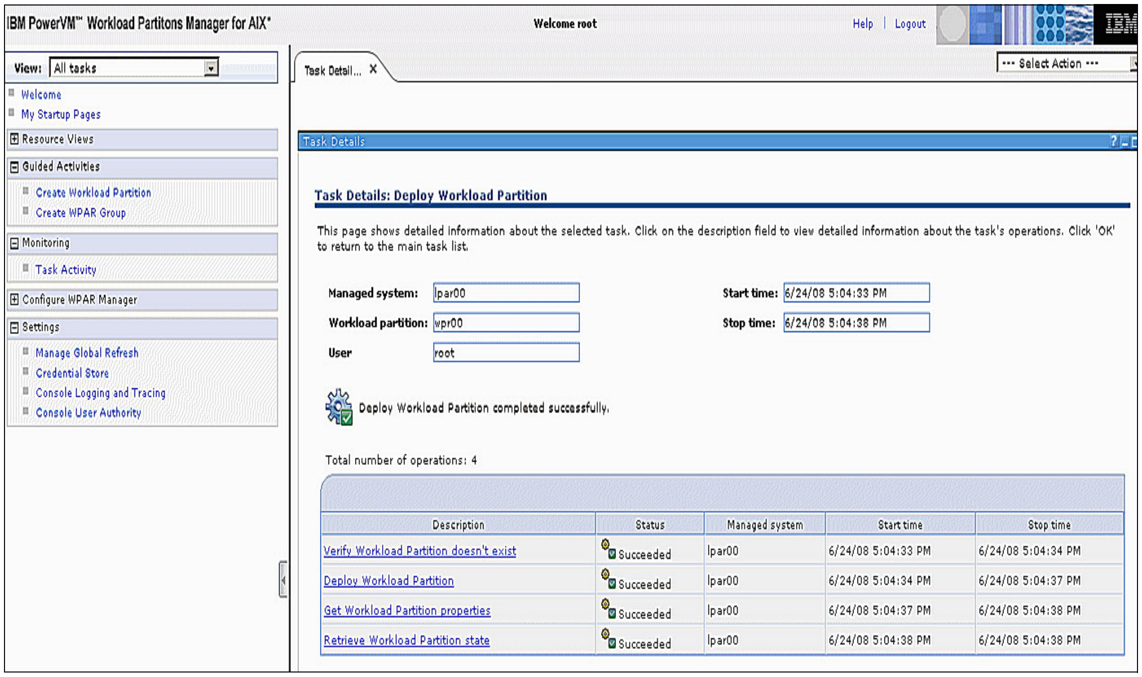
Description:

☐ Enable checkpoint

*Figure 5-9 Creating Application WPAR - the main window*

This supplies enough information to create a simple application WPAR, so click **Finish**.

After a short time your application WPAR will deploy; see Figure 5-10 on page 124.



**Task Details: Deploy Workload Partition**

This page shows detailed information about the selected task. Click on the description field to view detailed information about the task's operations. Click 'OK' to return to the main task list.

| | | | |
|---|---|---|---|
| **Managed system:** | lpar00 | **Start time:** | 6/24/08 5:04:33 PM |
| **Workload partition:** | wpr00 | **Stop time:** | 6/24/08 5:04:38 PM |
| **User** | root | | |

Deploy Workload Partition completed successfully.

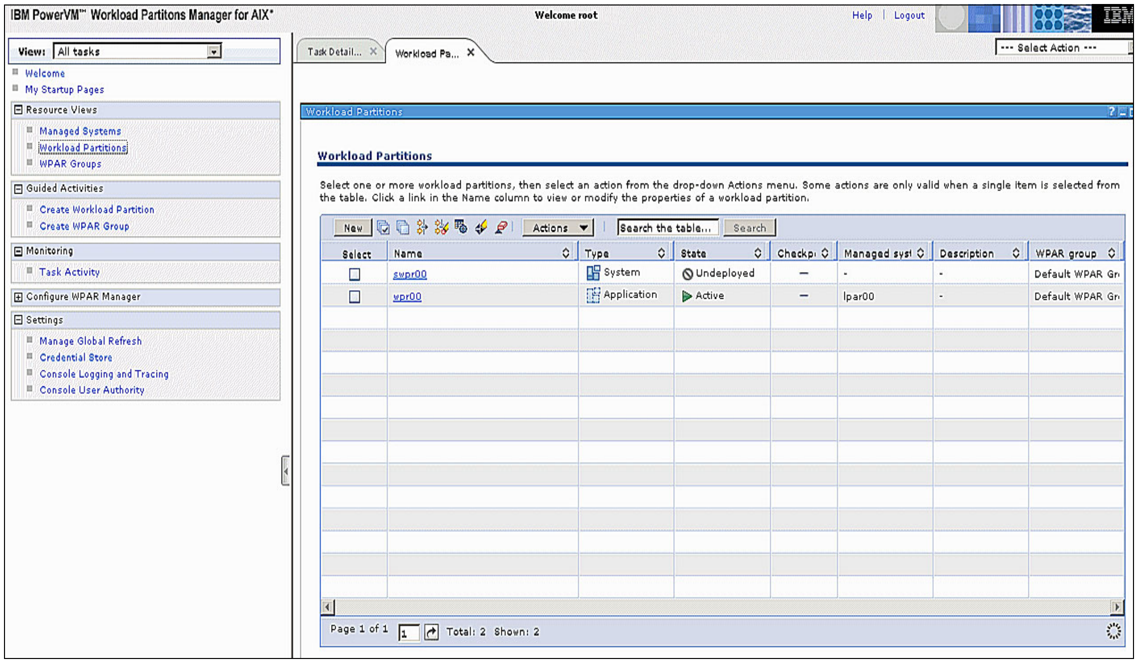
Total number of operations: 4

| Description | Status | Managed system | Start time | Stop time |
|---|---|---|---|---|
| Verify Workload Partition doesn't exist | Succeeded | lpar00 | 6/24/08 5:04:33 PM | 6/24/08 5:04:34 PM |
| Deploy Workload Partition | Succeeded | lpar00 | 6/24/08 5:04:34 PM | 6/24/08 5:04:37 PM |
| Get Workload Partition properties | Succeeded | lpar00 | 6/24/08 5:04:37 PM | 6/24/08 5:04:38 PM |
| Retrieve Workload Partition state | Succeeded | lpar00 | 6/24/08 5:04:38 PM | 6/24/08 5:04:38 PM |

*Figure 5-10 Task Details: Deploy Workload Partition*

Select **Workload Partitions** from Resource Views. Figure 5-11 on page 125 shows that your application is now running (`Active`) in Managed System lpar00.

The WPAR created is eligible for static mobility. Now, when it is running, move the WPAR to managed system lpar01. Mark wpr00 in the Select column, then select **Action, Relocate**. This is shown in Figure 5-12 on page 126.



**Workload Partitions**

Select one or more workload partitions, then select an action from the drop-down Actions menu. Some actions are only valid when a single item is selected from the table. Click a link in the Name column to view or modify the properties of a workload partition.

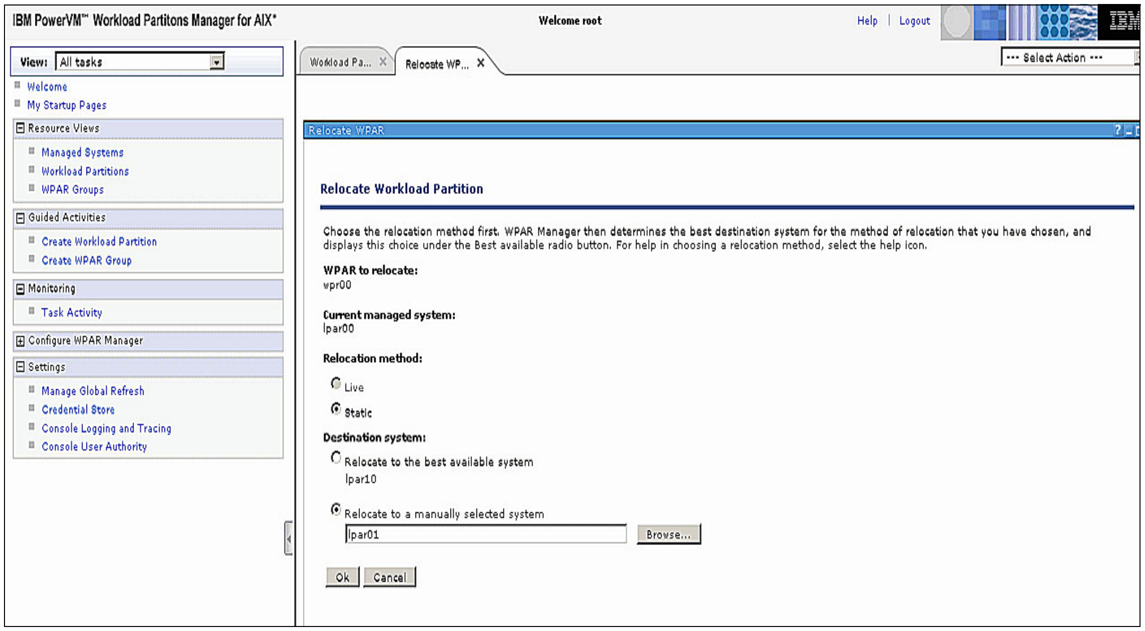
| Select | Name | Type | State | Checkp | Managed syst | Description | WPAR group |
|---|---|---|---|---|---|---|---|
| ☐ | swpr00 | System | Undeployed | – | - | - | Default WPAR Gr |
| ☐ | wpr00 | Application | Active | – | lpar00 | - | Default WPAR Gr |

Page 1 of 1 Total: 2 Shown: 2

Figure 5-11 Workload Partition overview

**Note:** When moving a WPAR using static relocation, WPAR Manager actually preforms a stop and redeploy. This means the application is stopped, and restarted at the new location.

To avoid having the application stopped, you must use a WPAR eligible for live mobility, as explained in 5.4.5, “Creating an application WPAR eligible for live mobility” on page 128.



Relocate Workload Partition

Choose the relocation method first. WPAR Manager then determines the best destination system for the method of relocation that you have chosen, and displays this choice under the Best available radio button. For help in choosing a relocation method, select the help icon.

**WPAR to relocate:**
wpr00

**Current managed system:**
lpar00

**Relocation method:**

Live [✓ Static]

**Destination system:**

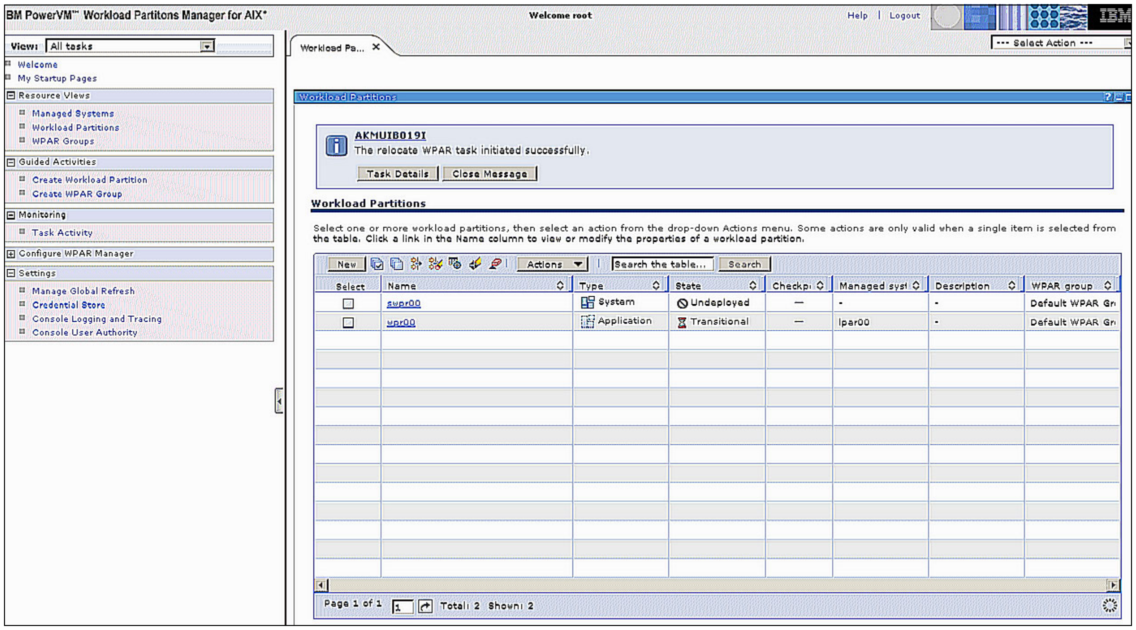
Relocate to the best available system
lpar10

[✓ Relocate to a manually selected system]
lpar01

*Figure 5-12 Static relocating*

The system is aware that you are only able to perform a static relocation in this instance. Simply fill out the name of the managed system to which you want to move your WPAR.

After you press OK, WPAR Manager starts to move the WPAR. Figure 5-13 on page 127 shows that the state of the WPAR is `Transitional`, which is the state while the move is in progress.



AKMUIB019I
The relocate WPAR task initiated successfully.

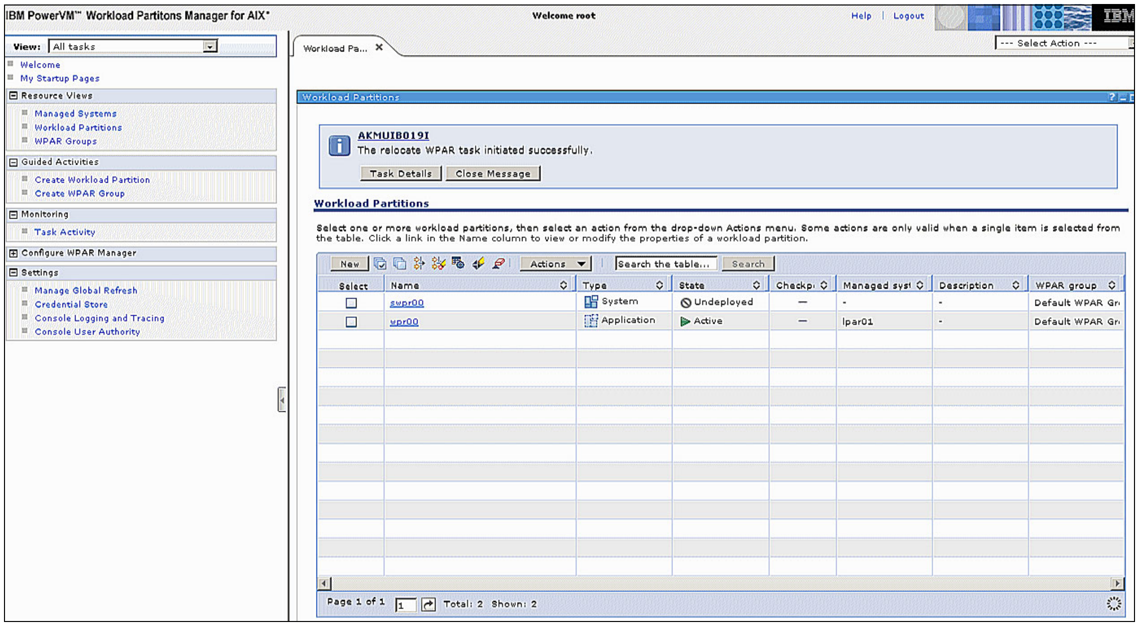
Workload Partitions

Select one or more workload partitions, then select an action from the drop-down Actions menu. Some actions are only valid when a single item is selected from the table. Click a link in the Name column to view or modify the properties of a workload partition.

| Select | Name | Type | State | Checkpoint | Managed syst | Description | WPAR group |
|---|---|---|---|---|---|---|---|
| ☐ | swpr00 | System | Undeployed | — | - | - | Default WPAR Gr |
| ☐ | wpr00 | Application | Transitional | — | lpar00 | - | Default WPAR Gr |

Figure 5-13 WPAR wpr00 is moving to new Managed System

Figure 5-14 shows your WPAR is now running in lpar01 with a state of `Active`.



AKMUIB019I
The relocate WPAR task initiated successfully.

Workload Partitions

Select one or more workload partitions, then select an action from the drop-down Actions menu. Some actions are only valid when a single item is selected from the table. Click a link in the Name column to view or modify the properties of a workload partition.

| Select | Name | Type | State | Checkpoint | Managed syst | Description | WPAR group |
|---|---|---|---|---|---|---|---|
| ☐ | swpr00 | System | Undeployed | — | - | - | Default WPAR Gr |
| ☐ | wpr00 | Application | Active | — | lpar01 | - | Default WPAR Gr |

Figure 5-14 WPAR wpr00 moved to lpar01

When using WPAR Manager to create or move WPARs, you will sometimes want to go back and check the status, or determine what happened the last time you performed the same operation. Or, you may simply want to check the status of the last activity. You can perform this checking from the WPAR Manager main window by selecting **Monitoring** -> **Task Activity**.

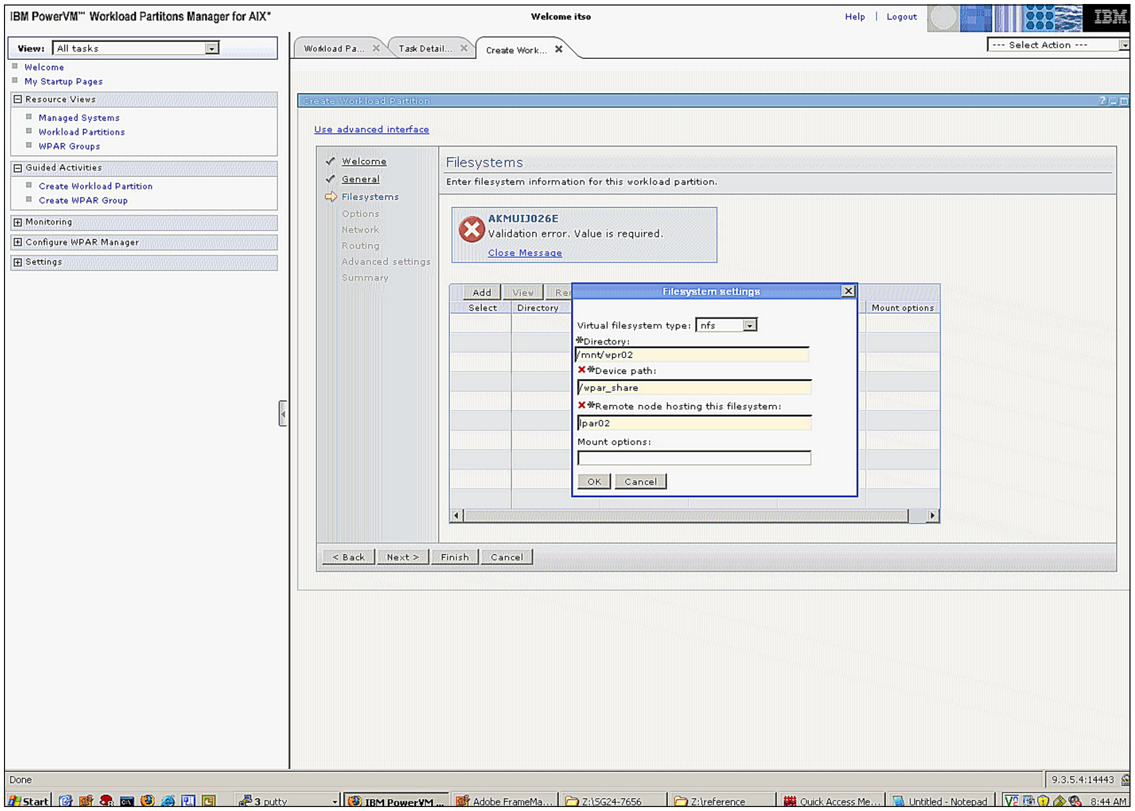
### 5.4.5 Creating an application WPAR eligible for live mobility

The differences between creating an application WPAR eligible for *live* mobility versus creating an application WPAR eligible for static mobility involve the following items:

- The Enable checkpoint
- Creating file systems on the NFS server for your application

To create an application WPAR eligible for live mobility, select **Enable checkpoint** and click **Next**. A new dialog will open. This dialog enables you to fill in the required NFS information, as shown in Figure 5-15 on page 129.

Fill in the required information (your directory for the application, the share name and the NFS server name). After you click **Finish**, WPAR Manager starts creating your application WPAR.



Filesystems

Enter filesystem information for this workload partition.

AKMUIJ026E
Validation error. Value is required.
Close Message

Filesystem settings

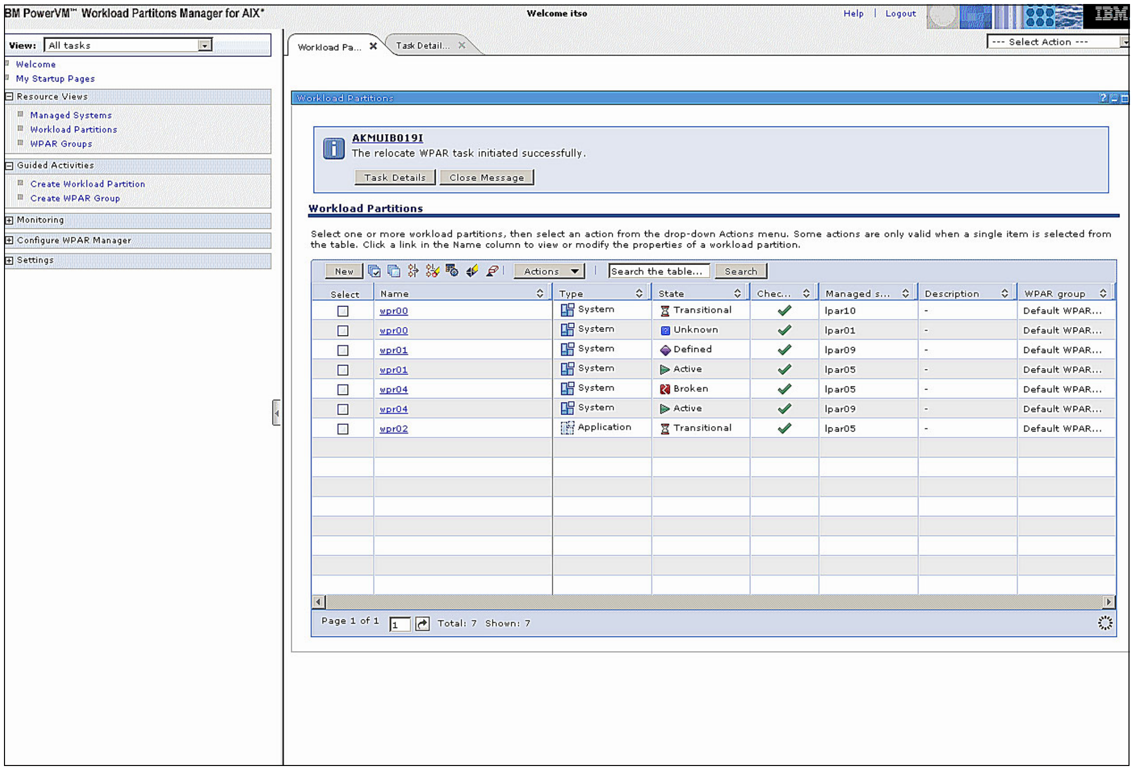
Virtual filesystem type: nfs
*Directory: /mnt/wpr02
*Device path: /wpar_share
*Remote node hosting this filesystem: lpar02
Mount options:

OK Cancel

*Figure 5-15 Creating Application WPAR eligible for live mobility*

In our case, we created our application WPAR in lpar05, and our application WPAR started. Next we moved it to lpar09 the same way as shown in Figure 5-12 on page 126. This time we ensured Live was activated, and then clicked **OK**.

As shown in Figure 5-16 on page 130, our application began moving. It suspended application operation and resumed it after our WPAR arrived on our new managed system.



**AKMUIB019I**
The relocate WPAR task initiated successfully.

**Workload Partitions**

Select one or more workload partitions, then select an action from the drop-down Actions menu. Some actions are only valid when a single item is selected from the table. Click a link in the Name column to view or modify the properties of a workload partition.

| Select | Name | Type | State | Chec... | Managed s... | Description | WPAR group |
|---|---|---|---|---|---|---|---|
| ☐ | wpr00 | System | Transitional | ✓ | lpar10 | - | Default WPAR... |
| ☐ | wpr00 | System | Unknown | ✓ | lpar01 | - | Default WPAR... |
| ☐ | wpr01 | System | Defined | ✓ | lpar09 | - | Default WPAR... |
| ☐ | wpr01 | System | Active | ✓ | lpar05 | - | Default WPAR... |
| ☐ | wpr04 | System | Broken | ✓ | lpar05 | - | Default WPAR... |
| ☐ | wpr04 | System | Active | ✓ | lpar09 | - | Default WPAR... |
| ☐ | wpr02 | Application | Transitional | ✓ | lpar05 | - | Default WPAR... |

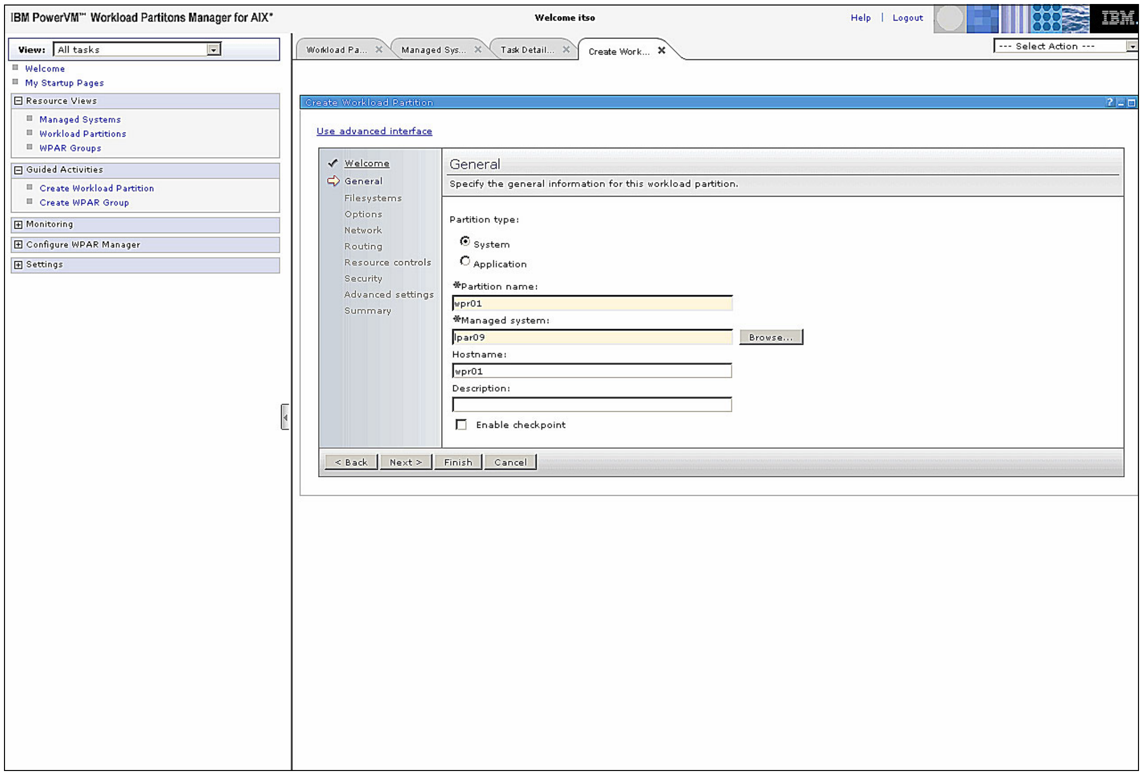
Page 1 of 1  1  Total: 7  Shown: 7

*Figure 5-16 Application WPAR live mobility in progress*

In Figure 5-16, notice that several WPARs are in different states. Some WPARs are in the `Transitional` state, some are in the `Active` state, and others are in `Unknown`, `Broken`, and `Defined` states. Using WPAR Manager, you can easily obtain an overview of all your WPARs, including where are they running and which state they are in at the moment.

## 5.4.6 Creating a system WPAR eligible for static mobility

Creating a system WPAR is similar to creating an application WPAR. As shown in Figure 5-17 on page 131, simply select **System** (instead of Application). Fill in hostname and the name of the managed system where you want your WPAR to run. Then click **Finish**.



General

Specify the general information for this workload partition.

Partition type:

[✓ System] Application

*Partition name: wpr01

*Managed system: lpar09

Hostname: wpr01

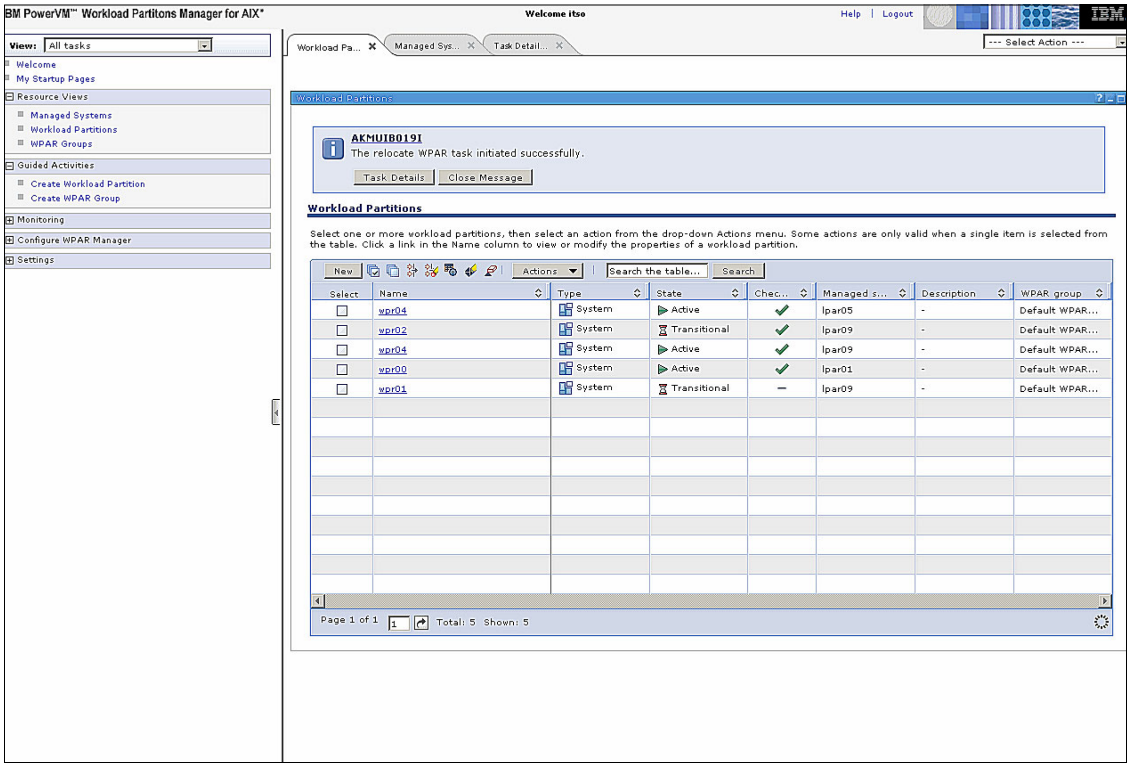
Description:

☐ Enable checkpoint

*Figure 5-17 Creating System WPAR for static mobility*

As done with your application WPAR, move your WPAR to a new managed system by using the same method shown in Figure 5-12 on page 126.

As shown in Figure 5-18 on page 132, your system starts moving to its new location.



AKMUIB019I
The relocate WPAR task initiated successfully.

Workload Partitions

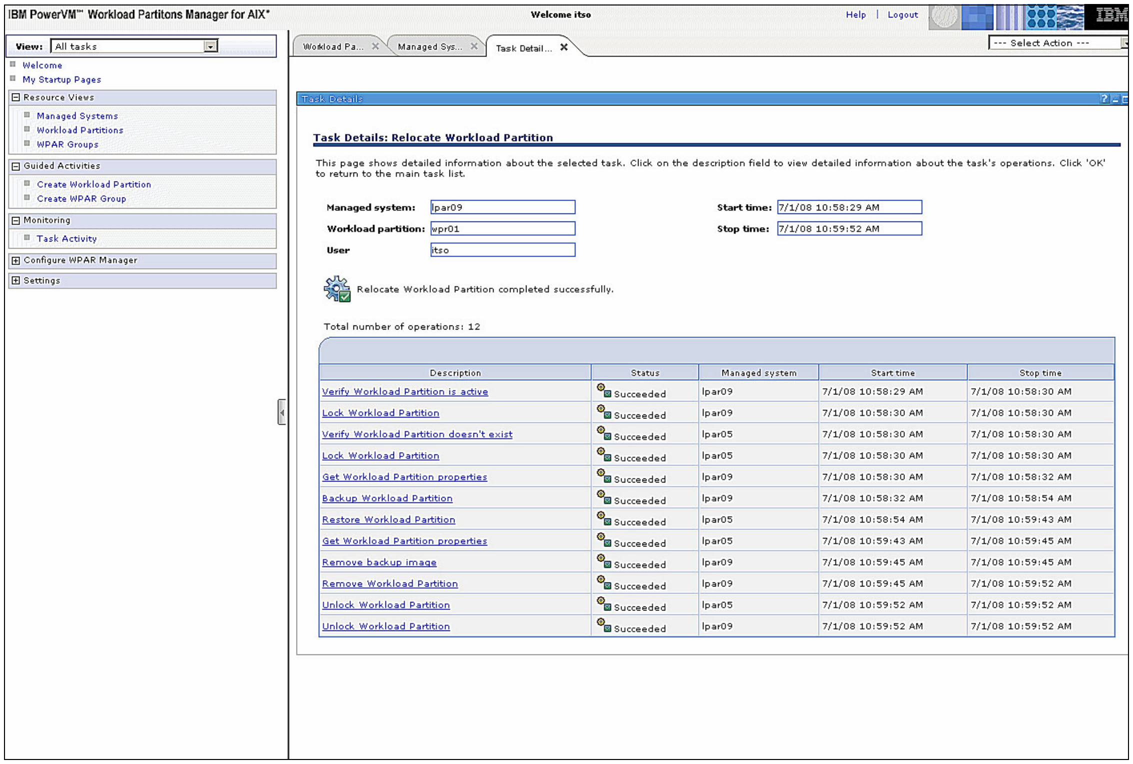
Select one or more workload partitions, then select an action from the drop-down Actions menu. Some actions are only valid when a single item is selected from the table. Click a link in the Name column to view or modify the properties of a workload partition.

| Select | Name | Type | State | Chec... | Managed s... | Description | WPAR group |
|---|---|---|---|---|---|---|---|
| ☐ | wpr04 | System | Active | ✓ | lpar05 | - | Default WPAR... |
| ☐ | wpr02 | System | Transitional | ✓ | lpar09 | - | Default WPAR... |
| ☐ | wpr04 | System | Active | ✓ | lpar09 | - | Default WPAR... |
| ☐ | wpr00 | System | Active | ✓ | lpar01 | - | Default WPAR... |
| ☐ | wpr01 | System | Transitional | — | lpar09 | - | Default WPAR... |

Page 1 of 1 Total: 5 Shown: 5

*Figure 5-18 System WPAR static mobility in progress*

Notice that the `Transitional` state is displayed for your WPAR. To view details from the process, select the WPAR and click **Task details**. When the process is finished, you see a display as shown Figure 5-19 on page 133.



**Task Details: Relocate Workload Partition**

This page shows detailed information about the selected task. Click on the description field to view detailed information about the task's operations. Click 'OK' to return to the main task list.

Managed system: lpar09
Workload partition: wpr01
User itso
Start time: 7/1/08 10:58:29 AM
Stop time: 7/1/08 10:59:52 AM

Relocate Workload Partition completed successfully.

Total number of operations: 12

| Description | Status | Managed system | Start time | Stop time |
|---|---|---|---|---|
| Verify Workload Partition is active | Succeeded | lpar09 | 7/1/08 10:58:29 AM | 7/1/08 10:58:30 AM |
| Lock Workload Partition | Succeeded | lpar09 | 7/1/08 10:58:30 AM | 7/1/08 10:58:30 AM |
| Verify Workload Partition doesn't exist | Succeeded | lpar05 | 7/1/08 10:58:30 AM | 7/1/08 10:58:30 AM |
| Lock Workload Partition | Succeeded | lpar05 | 7/1/08 10:58:30 AM | 7/1/08 10:58:30 AM |
| Get Workload Partition properties | Succeeded | lpar09 | 7/1/08 10:58:30 AM | 7/1/08 10:58:32 AM |
| Backup Workload Partition | Succeeded | lpar09 | 7/1/08 10:58:32 AM | 7/1/08 10:58:54 AM |
| Restore Workload Partition | Succeeded | lpar05 | 7/1/08 10:58:54 AM | 7/1/08 10:59:43 AM |
| Get Workload Partition properties | Succeeded | lpar05 | 7/1/08 10:59:43 AM | 7/1/08 10:59:45 AM |
| Remove backup image | Succeeded | lpar09 | 7/1/08 10:59:45 AM | 7/1/08 10:59:45 AM |
| Remove Workload Partition | Succeeded | lpar09 | 7/1/08 10:59:45 AM | 7/1/08 10:59:52 AM |
| Unlock Workload Partition | Succeeded | lpar05 | 7/1/08 10:59:52 AM | 7/1/08 10:59:52 AM |
| Unlock Workload Partition | Succeeded | lpar09 | 7/1/08 10:59:52 AM | 7/1/08 10:59:52 AM |

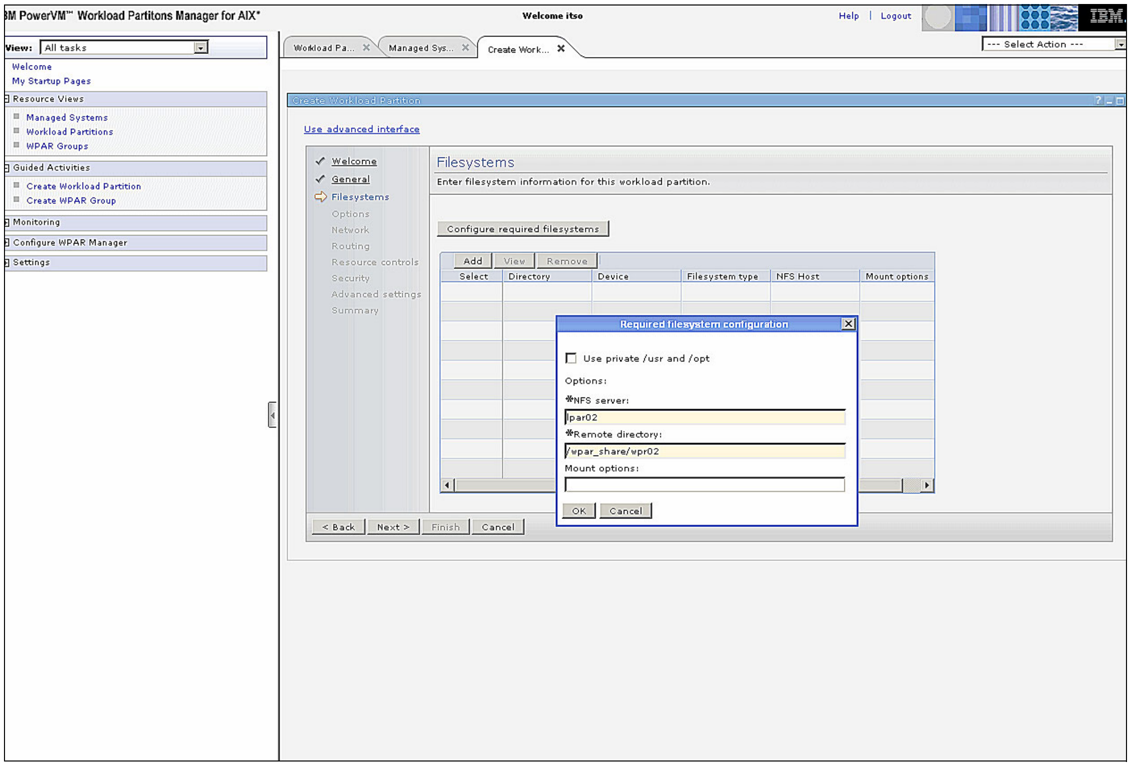
*Figure 5-19 System WPAR static move completed*

## Creating a system WPAR eligible for live mobility

To create a system WPAR that is eligible for live mobility, you start the same way by filling in the WPAR name and Managed System name; see Figure 5-17 on page 131.

**Tip:** Before creating your WPAR, make sure that the NFS server has exported the necessary file systems with the correct permissions.

After filling in the basic information, select **Enable checkpoint** and click **Next**. In the file system window, click **Configure required filesystems**. A pop-up box appears, as shown in Figure 5-20 on page 134. In the box, supply the required information about the NFS server and sharename, and then click **OK**. The system now displays the predefined filesystems; click **Finish**.



Filesystems

Enter filesystem information for this workload partition.

☐ Use private /usr and /opt

Options:

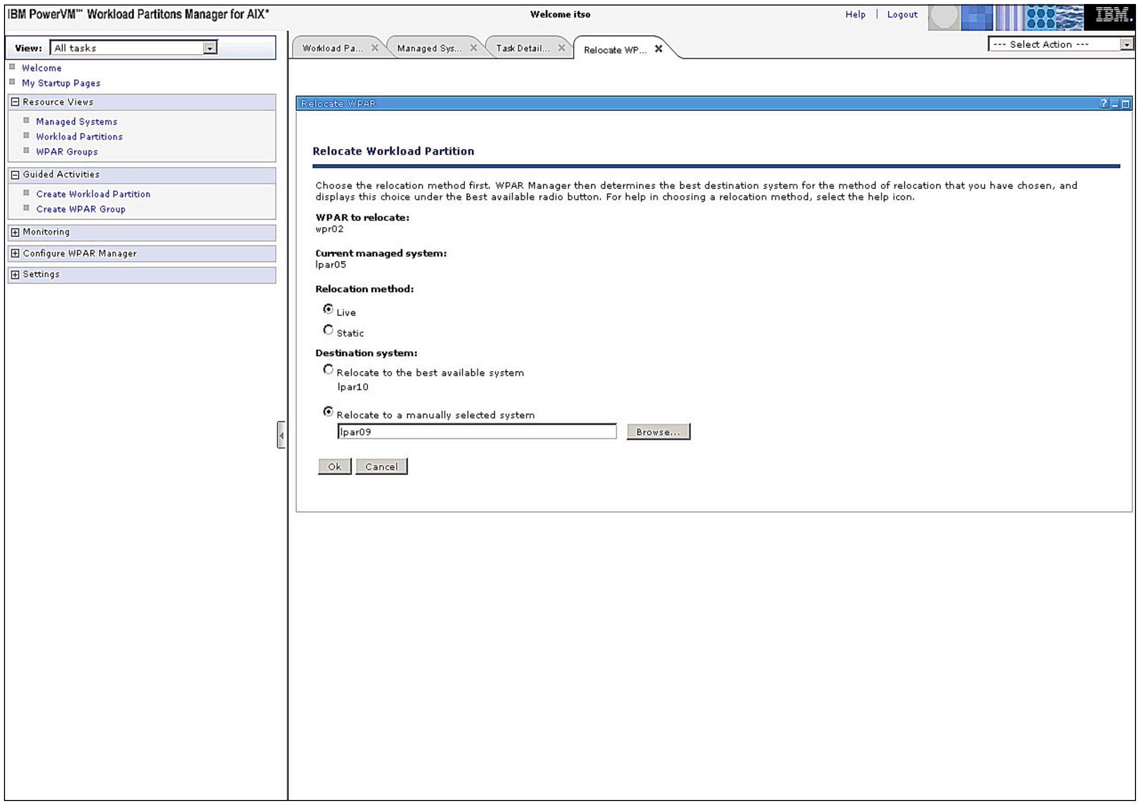
*NFS server: lpar02

*Remote directory: /wpar_share/wpr02

Mount options:

Figure 5-20 System WPAR NFS server settings

After creating your WPAR, start it. Then initiate live mobility so that you can move your WPAR to another managed system. You do this just as shown in Figure 5-12 on page 126—but in this case ensure that **Live** is selected and that the Managed System to which you want to move your WPAR is filled in; see Figure 5-21 on page 135.



**Relocate Workload Partition**

Choose the relocation method first. WPAR Manager then determines the best destination system for the method of relocation that you have chosen, and displays this choice under the Best available radio button. For help in choosing a relocation method, select the help icon.

**WPAR to relocate:**
wpr02

**Current managed system:**
lpar05

**Relocation method:**

[✓ Live]
Static

**Destination system:**

Relocate to the best available system
lpar10

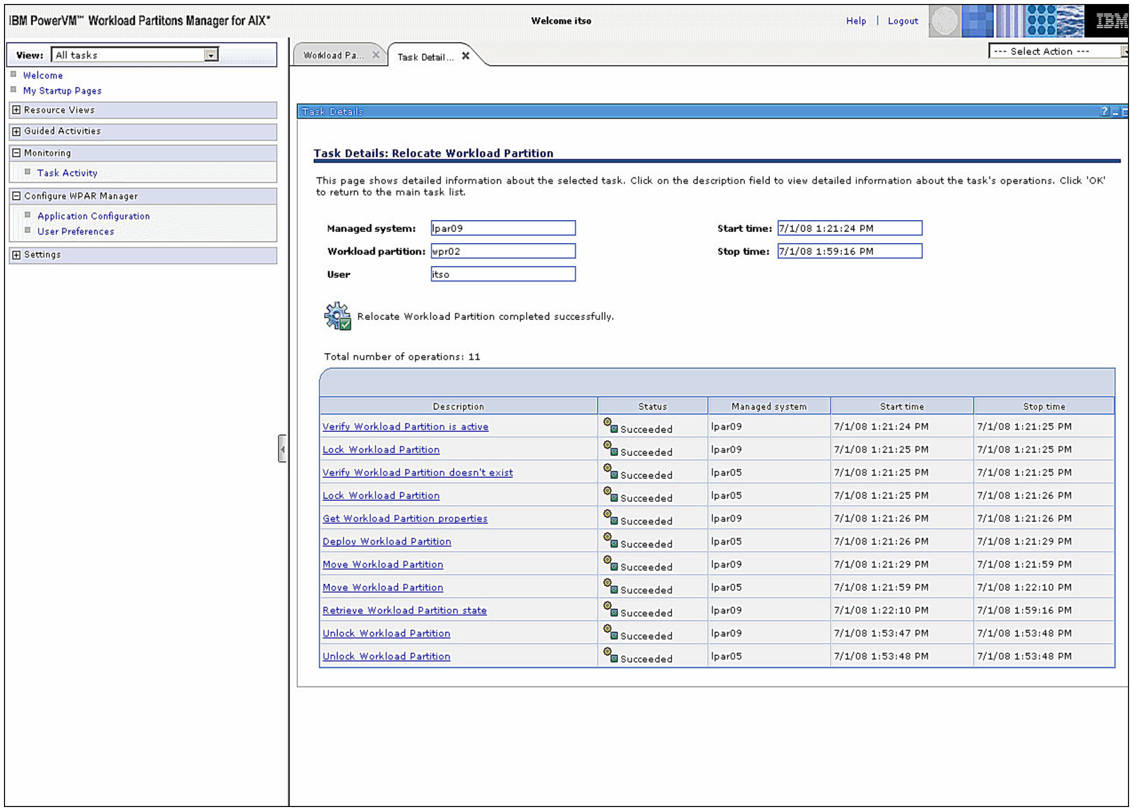
[✓ Relocate to a manually selected system]
lpar09 Browse...

Ok Cancel

*Figure 5-21 Activating System WPAR moving from one managed system to another*

After you click **OK**, your WPAR begins to move to the new managed system. After a few moments the move completes and it is shown as running in lpar09.

Figure 5-22 on page 136 displays the Task Activity window.



**Task Details: Relocate Workload Partition**

This page shows detailed information about the selected task. Click on the description field to view detailed information about the task's operations. Click 'OK' to return to the main task list.

| | | | |
|---|---|---|---|
| **Managed system:** | lpar09 | **Start time:** | 7/1/08 1:21:24 PM |
| **Workload partition:** | wpr02 | **Stop time:** | 7/1/08 1:59:16 PM |
| **User** | itso | | |

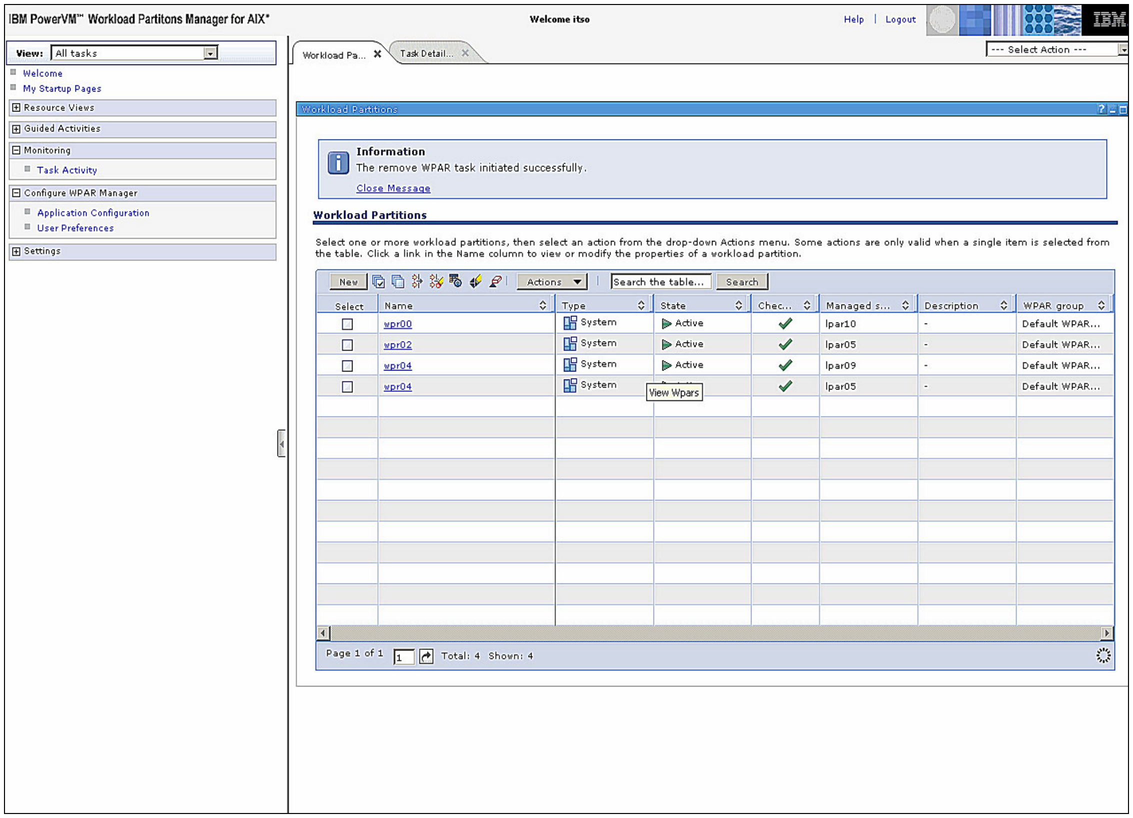
Relocate Workload Partition completed successfully.

Total number of operations: 11

| Description | Status | Managed system | Start time | Stop time |
|---|---|---|---|---|
| Verify Workload Partition is active | Succeeded | lpar09 | 7/1/08 1:21:24 PM | 7/1/08 1:21:25 PM |
| Lock Workload Partition | Succeeded | lpar09 | 7/1/08 1:21:25 PM | 7/1/08 1:21:25 PM |
| Verify Workload Partition doesn't exist | Succeeded | lpar05 | 7/1/08 1:21:25 PM | 7/1/08 1:21:25 PM |
| Lock Workload Partition | Succeeded | lpar05 | 7/1/08 1:21:25 PM | 7/1/08 1:21:26 PM |
| Get Workload Partition properties | Succeeded | lpar09 | 7/1/08 1:21:26 PM | 7/1/08 1:21:26 PM |
| Deploy Workload Partition | Succeeded | lpar05 | 7/1/08 1:21:26 PM | 7/1/08 1:21:29 PM |
| Move Workload Partition | Succeeded | lpar09 | 7/1/08 1:21:29 PM | 7/1/08 1:21:59 PM |
| Move Workload Partition | Succeeded | lpar05 | 7/1/08 1:21:59 PM | 7/1/08 1:22:10 PM |
| Retrieve Workload Partition state | Succeeded | lpar09 | 7/1/08 1:22:10 PM | 7/1/08 1:59:16 PM |
| Unlock Workload Partition | Succeeded | lpar09 | 7/1/08 1:53:47 PM | 7/1/08 1:53:48 PM |
| Unlock Workload Partition | Succeeded | lpar05 | 7/1/08 1:53:48 PM | 7/1/08 1:53:48 PM |

*Figure 5-22 Task Activity window for Live system WPAR move*

Figure 5-23 on page 137 shows the system WPAR moved to the end location.



**Information**
The remove WPAR task initiated successfully.
Close Message

**Workload Partitions**

Select one or more workload partitions, then select an action from the drop-down Actions menu. Some actions are only valid when a single item is selected from the table. Click a link in the Name column to view or modify the properties of a workload partition.

| Select | Name | Type | State | Chec... | Managed s... | Description | WPAR group |
|---|---|---|---|---|---|---|---|
| ☐ | wpr00 | System | Active | ✓ | lpar10 | - | Default WPAR... |
| ☐ | wpr02 | System | Active | ✓ | lpar05 | - | Default WPAR... |
| ☐ | wpr04 | System | Active | ✓ | lpar09 | - | Default WPAR... |
| ☐ | wpr04 | System | | ✓ | lpar05 | - | Default WPAR... |

Page 1 of 1 Total: 4 Shown: 4

*Figure 5-23 System WPAR moved to end location*

This chapter provides simple examples of both application WPARs and system WPARs. In the real world, you will need to define more file systems, customize the WPAR to fit your application, and perhaps also install additional products.

## 5.5 Compatibility API

WPAR Manager can be used to manage mobility events among the servers under its control. In order to minimize costly and time-consuming failures, WPAR Manager would like to be able to determine whether the target of a mobility operation (the arrival LPAR) is likely to be an acceptable candidate.

The compatibility API provides the tools to facilitate the collection of system properties on both the departure and arrival systems, and the testing of those profiles for potential compatibility.

This section provides an overview of the compatibility API.

## 5.5.1 Compatibility test flow

You need a system with the ability to retrieve values for a predetermined set of system properties. The best way to illustrate how this is done in the compatibility API is shown in Figure 5-24.

Figure 5-24 shows a typical WPAR move scenario.

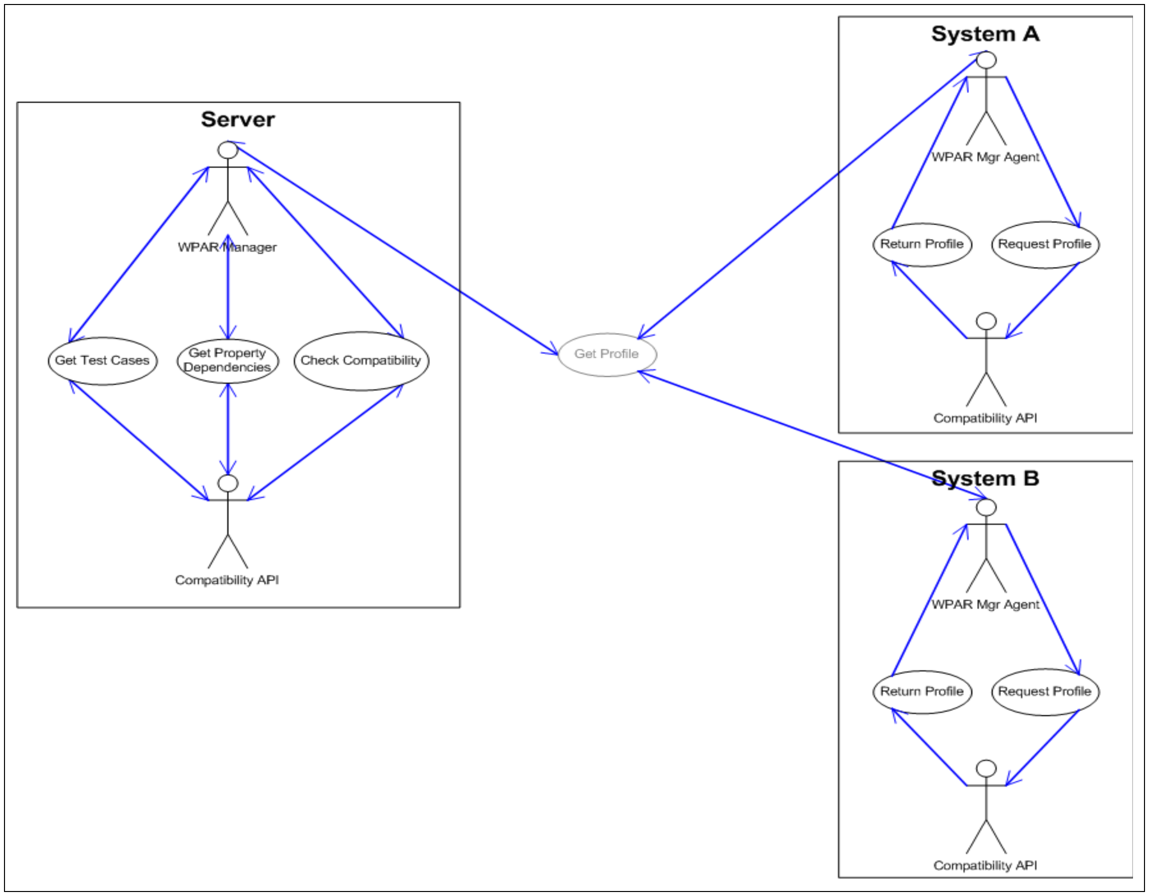


*Figure 5-24 A typical WPAR move scenario*

Table 5-2 on page 139 lists and explains the steps shown in Figure 5-24.

*Table 5-2 Steps of a typical WPAR move scenario*

| Seq no. | Activity | Actor | Description |
|---|---|---|---|
| 1 | Get Test Cases | WPAR Manager (Server) | Having determined that a mobility operation from departure System A is called for, and having selected System B as a candidate arrival server, the WM requests a comprehensive list of all available Test Cases. The WM then filters the list, for example, by displaying those test cases to the human user with check boxes. |
| 2 | Get Property Dependencies | WPAR Manager (Server) | The WM iterates through the filtered list of test cases to construct a set of System Property Keys upon which the test cases are dependent. |
| 3 | Get Profile (System A) | WPAR Manager (Server) | The WM passes the set of properties gleaned in step 2 to System A. (Note: The Get Profile activity is a network request between the WPAR Manager Server and the WPAR Manager Agent; this functionality is not provided by the Compatibility API.) |
| 4 | Request Profile | WPAR Manager (Server) System A | On System A, the WPAR Manager Agent passes the set of properties gleaned in step 2 to the I System Property Collector implementor. |
| 5 | Return Profile | Compatibility API (System A) | The Compatibility API on System A iterates through the passed-in list of keys, querying the underlying system to discover the associated values. The Compatibility API populates a System Profile with these values and returns it to the WPAR Manager Agent. |
| 6 | Get Profile (System A) | WPAR Mgr Agent (System A) | The WPAR Manager Agent on System A passes the populated System Profile back to the WPAR Manager Server. (Note: The Get Profile activity is a network request between the WPAR Manager Server and the WPAR Manager Agent; this functionality is not provided by the Compatibility API.) |
| 7-10 | (See steps 3-6) | See steps 3-6) | Steps 3-6 are repeated for System B and for any other candidate arrival systems. The WM Server now has one System Profile for System A, and one for each candidate arrival system. |

| Seq no. | Activity | Actor | Description |
|---|---|---|---|
| 11 | Check Compatibility | WPAR Manager (Server) | The WPAR Manager Server calls upon the Compatibility Tester to check compatibility between pairs of servers. The System Profile for System A is always used as the departure profile. (If multiple arrival candidates require testing, each subsequent call uses a profile from a different candidate as the arrival profile.) The filtered set of Test Cases from step 1 is passed to the Compatibility Tester along with the profiles. The Compatibility Tester returns a Test Results object populated with one Test Result instance for each Test Case. |

## 5.5.2 Compatibility test cases

To review the following points:

- Live application mobility is the process of relocating a WPAR while preserving the state of the application stack.
- Static application mobility is the process of relocating a WPAR while preserving its filesystem state.

Live relocation requires more extensive compatibility testing than static relocation. Therefore, it is possible that two systems could be incompatible for live relocation, but compatible for static relocation. Therefore, compatibility testing is needed.

The test can be categorized as:

- Hardware levels (the two systems must have identical processor types)
- Installed hardware features
- Installed devices
- Operating system levels and patch levels
- Special attention for filesets
  - bos.rte
  - bos.wpars
  - bos.rte.libc
  - mcr.rte
- Other software or file systems installed with the operating system
- Additional user-selected tests
  - NTP enablement
  - CPU speed
  - Available memory
  - User-written scripts

### 5.5.3 API usage

Note the following points regarding API usage:

- The WPAR Manager can run on several platforms, thus the compatibility API is written in Java.
- No CLI implementation of these API functions is available.

The API is shipped in the bos.wpars fileset as three jar files and one shared library:

- /usr/lib/wpars/compat/common.jar
- /usr/lib/wpars/compat/aix.jar
- /usr/lib/wpars/compat/aixspc.jar
- /usr/lib/libwparcompat.so

## 5.6 Performance monitoring

The WPAR agent that communicates with WPAR Manager can periodically send back matrixes that can allow the administrator to track the performance of servers on a variety of levels. These matrixes track performance levels of both individual WPARs and the managed systems. The matrixes are used both for informational purposes and to drive automated relocations (for details, see 5.7, “WPAR groups and load management” on page 142). The manager maintains this performance data in a database, and can display historical visualizations of the data to the administrator.

By default, XML data is sent back to the manager every minute. This interval can be changed. However, the setting is a global value that applies to all managed systems.

The data sent back by the agent can contain the following information:

- System Status - this includes the GUID and version of the agent, number and identity of WPARs running underneath, and so on.
- WPAR Status - this includes the type, current state, whether or not it is checkpointable, and so on.
- Performance Matrixes - these are sent not only for the system as a whole but also for the individual WPARs. They can contain information such as CPU and memory utilization, process and thread counts, filesystem usage, amount of time spent using the disk, and so on.

Care must be taken when interpreting performance matrixes for a workload partition. These matrixes are useful tools for inferring the performance of an application. However. they must be interpreted alongside system-wide performance matrixes to arrive at a complete picture.

For example, if an agent reports that a WPAR is consuming 20% of the available CPU resource in a system, this data point could indicate low levels of demand on that workload. However, if four other workloads are consuming 80% of the CPU during this period, then all workloads on this system are, in fact, resource-constrained.

To minimize these kinds of misinterpretations, WPAR Manager reports performance matrixes from WPARs using the following formula. Using the formula, each WPAR in the preceding example would indicate CPU utilization of 100%:

```
WPAR_utilization / (1 - (LPAR_utilization - WPAR_utilization))
```

The goal, of course, is to estimate overall performance level of the workload running in the partition, so matrixes should be selected based on the demands of the workload (some may be CPU-bound, while others may depend more heavily on disk I/O).

The ideal metric in these cases is application response time. Administrators may optionally provide scripts to test a workload and measure the response time of the application. These values can then be submitted into the WPAR Manager as matrixes.

If this approach is utilized, take care to ensure that the response time measured only applies to the application under test. For example, a test causing an application to query a database workload running in a different WPAR would not be a good candidate for measuring application response time, because two tiers of workloads are being tested.

## 5.7 WPAR groups and load management

A WPAR group is a collection of WPARs containing similar application instances across a number of managed systems. All WPARs in the group are governed by a single policy, which can be used as a basis for WPAR Manager's automated relocation functions. These features are typically used for automated load management.

When an administrator creates a WPAR, WPAR Manager automatically allocates the WPAR to the Default WPAR Group. On a fresh installation of AIX, this is the only group that exists. Administrators can easily create their own groups by selecting the function **Guided Activities** -> **Create WPAR Group** from the GUI.

**Tip:** A WPAR can only belong to one group at time.

Generally, it is recommended that WPARs fulfilling similar functions be grouped together. For example, Web server WPARs running across multiple managed systems could be grouped under a single policy. Database WPARs could be grouped the same way. Following this methodology, the administrator would end up with several tiers of applications, each comprised of a group of clustered WPARs.

A WPAR group has a policy associated with it. This policy consists of settings that fall into one of the following categories:

- General settings - the name, description, averaging period for measurements, and so on
- Matrixes - measurements averaged together and used to determine when a WPAR should be relocated
- Thresholds - limits to the matrixes calculation that will trigger a relocation if exceeded
- System profiles - list of systems that will be considered as potential relocation targets
- Compatibility tests - optional tests run before a relocation is attempted (refer to 5.5, “Compatibility API” on page 137

WPAR Manager periodically collects the matrixes specified in the policy, and averages them together to derive a performance state for the group as a whole. Some matrixes may be weighted more than others, depending on the policy is defined. Some matrixes may be defined and provided by the administrator using scripts. If the performance state for the group falls outside a threshold, WPAR Manager triggers a relocation event.

Based on performance analysis of the WPARs, WPAR group, and overall utilization of the managed systems, one of the following actions is triggered:

- Scale up - relocation to a more powerful machine
- Scale down - relocation to a less powerful machine
- Scale out - relocation to a dedicated machine
- Scale in - consolidation to a shared machine

Only one WPAR is relocated at a time. After each relocation action, the performance of the WPAR group is reassessed to determine if additional action is required.

Chapter 8, “Resource control” on page 241 provides more detailed information about how to set up and use resource controls (RAS).

## 5.8 WPAR Manager availability

Keeping management system availability is crucial; you do not want to lose functionality in the management area if one or more hardware systems is down. You will need to be able to use your known management tools in a disaster recovery situation.

### Duplicate WPAR Managers

One method of maintaining management system availability is to have two LPARs (lpar00 and lpar01) installed and configured with WPAR Manager software. In our case, all of our agents were configured to communicate with lpar00 and the WPAR Manager node is operational.

If we lost lpar00 due to software or hardware problems, we simply had to re-reregister our client to our manager on lpar01 using this command:

```
/opt/IBM/WPAR/agent/bin/configure-agent -force -amhost lpar01 \
-passwd itsoadmin
```

The agent password on lpar00 and lpar01 do not need to be the same.

In the preceding command the **-force** option was required because the agent was already working with lpar00; therefore we had to force the change so it could use lpar01 instead.

You must run the re-register command on all managed nodes on which you want the new manager to take control. During this process, the running WPARs will not be disturbed. They continue to run undisturbed.

However, if any WPAR Management activities were active that cause a state change in the WPAR (that is, creating or moving WPARs), then these WPARs will probably end in the Broken state (refer to Table 4-1 on page 66 for a definition of the Broken state).

Part 3

# Advanced topics

This part covers the following advanced WPAR topics:

- Security in WPAR environments
- Advanced configuration features
- Resource control
- Developer considerations

6

# Security in workload partition environments

This chapter discusses various aspects regarding security and isolation in WPAR environments. Each section includes also practical examples and real life scenarios. Although the global environment and workload partitions are completely separate entities, they are closely connected. Therefore, you must understand how the usual security concerns related to stand-alone AIX systems apply to WPARs, given their specific nature.

This chapter discusses the following security issues related to WPARs:

- WPAR isolation
- File system isolation and security
- Encrypted file systems
- Users and groups
- RBAC in WPAR environments
- Auditing and accounting in WPAR environments

## 6.1 WPAR isolation

To understand WPAR isolation, keep in mind that each WPAR represents an individual environment that relies on and uses services provided by the global environment.

WPARs are able to communicate with other WPARs residing in the same global environment or any other external system. At the same time, each WPAR is separated by the global environment and other WPARs. Each application running in a workload partition is provided with a private execution environment. This has been achieved by implementing a strictly-controlled set of features by which a WPAR is allowed to interact with the global environment, another WPAR, or an external system.

The global environment represents a framework that controls all user-level and system-level objects that are usual for any regular AIX operating system image such as devices, file systems, or processes. At the core of the whole WPAR technology is the shared kernel, meaning that the global environment and all active WPAR instances share a *single* kernel.

WPARs and global environment have been designed so that administrative tasks and commands that can be run from the global environment have the ability to affect WPAR operation. However, the potential of a partition to interfere with a different partition or the global environment is strictly limited. The result is a set of WPAR restrictions and customizations that includes the following categories:

- Device access
  - Access to storage devices. Physical-disk devices and LVM associated objects such as physical volumes, volume groups, or logical volumes are not available within a WPAR. Access to data is performed through file systems that are mounted from the global environment into the WPAR.
  - Physical devices such as network devices are not available in a WPAR.
  - WPARs have access only to a set of pseudo devices such as /dev/ptyp0.
  - Devices that could provide a more global view of the system such as /dev/mem or /dev/kmem have been removed.
  - Access to system-generic devices that are safe, /dev/null or /dev/zero, is allowed.
  - The WPARs are *not* capable of creating new devices by themselves, for instance, by using the **mknod** command.
- File system access
  - File systems within a system WPAR can only be accessed from that WPAR or from the global environment.

  - File system objects belonging to the global environment and exported may be accessed within the normal constraints of shared access for those objects from both environments.
  - Even though a physical file system can be mounted in a WPAR, some device interfaces associated to the file system do not appear within the WPAR environment.
  - Some utilities that access file system metadata and require access to certain /dev devices do not work when invoked from the WPAR. In consequence, certain file system operations such as extension or defragmentation are not allowed from the WPAR.
  - Regular global users cannot access WPAR file systems unless explicitly given specific privileges.
  - WPAR administrators cannot mount or umount WPAR file systems that are mounted or umounted from the global environment.
- Network access
  - Modification of network settings such as the IP address is not allowed from the WPAR.
  - Processes from a WPAR can bind, send, or receive packets only from the the IP addresses and ports configured in the WPAR.
  - Outgoing IP datagrams must use, as the source IP, the IP address that is assigned to the WPAR.
  - Application processes are not allowed to bypass the network stack.
- System settings
  - Access to system-wide settings is restricted.
  - Access to system-wide objects is restricted.
  - Access to certain system-wide configuration files such as those contained in /etc has been restricted.
- Command line interface
  - The ability to use certain administrative commands that could affect the global environment has been removed. For instance, it is not allowed to modify the date or time from a WPAR or to bind a process to a specific processor.
  - System-wide parameters cannot be changed from a WPAR using the command line interface.
- Security and user management
  - User, group, and security repositories of all WPARs represent distinct entities. They are also different from the global repository.

  - Applications running in a WPAR derive their credentials from the local WPAR repositories. The scope of credentials is confined to the WPAR limits.
  - The scope of WPAR root privileges is contained within the WPAR boundaries and cannot interfere with the global environment.
- Process resources and intercommunication
  - Processes that run in the global environment and have appropriate privileges can view and signal processes within a WPAR.
  - Processes belonging to a WPAR cannot be reassigned to a different WPAR.
  - Processes within a WPAR can only create, view, and access the resources owned by the WPAR. These resources include non-shared file systems, devices, network interfaces, or ports.
  - Processes within a WPAR can only access files located in the file systems mounted in the same WPAR.
  - Processes within a WPAR can only see processes running in the same WPAR.
  - Processes within a WPAR can send signals only to processes in the same WPAR
  - Processes within a WPAR can share System V IPC mechanisms (shared memory areas, message queues, semaphores) or POSIX IPCs only with other processes executing in the same WPAR.
  - Resources used by applications running in a WPAR are tagged with the ID of that WPAR.
- Kernel manipulation
  - The ability to load or unload system-level device drivers and kernel extensions from a WPAR has been removed.

WPAR isolation has been achieved by implementing a set of changes that includes modifications of APIs, commands, and kernel as discussed in the next sections.

### 6.1.1 APIs

Usually, hosted applications will not be aware that they are running in a WPAR. In order to ensure full compatibility and be entirely functional, applications must use only supported interfaces and standard system calls so that the shared kernel always returns the appropriate identifier when a system call is invoked from a WPAR.

All AIX APIs are available in a WPAR. However, there are certain function calls which are not permissible in a WPAR. For instance, if a function call in a WPAR may affect the global environment, it will fail and return one of the following errors:

- EPERM
- EACCES
- EINVAL
- ENOSYS
- ENXIO
- EADDRNOTAVAIL

### 6.1.2 Commands

Most of the usual AIX commands are available in a WPAR. Certain commands have been modified or added new parameters. The output of the commands is significant to the context in which the command was run.

For instance, the behavior of the `df` command depends on whether is run from the WPAR or the global environment. When run from a WPAR, it displays information on WPAR mounted file systems only and the paths displayed are relative to the WPAR root. When run in the global environment, it displays information on all file systems and the paths returned are absolute.

Commands that could affect the global environment have been restricted.

### 6.1.3 Kernel changes

Having a shared kernel implies that both global environment and all WPAR instances use services provided by a unique AIX kernel. The kernel is able to handle and manage resources belonging to different WPARs and still maintain complete separation. For instance, file handles or memory areas allocated to a WPAR cannot be accessed from a different WPAR.

There have been changes implemented for new AIX kernel services. Using these services, the kernel can determine the status of a partition or can identify the WPAR which owns the process that initiates a particular system call.

## 6.2 File system isolation and security

File systems from each System WPAR represent a different branch of the entire file system space that is available at the global level. This ensures that file systems local to a WPAR cannot be accessed from another WPAR. Regular global users cannot access WPAR file systems unless explicitly given specific privileges.

Because device access within a WPAR is restricted, file system operations such as extension or defragmentation are not allowed from the WPAR.

Access to critical file system components, such as the superblock, allows seamless functionality for usual file system read, write, and execute operations while maintaining isolation and security.

The next sections discuss file system-specific security features and show how they can be used in WPAR environments.

### 6.2.1 Discretionary Access Control

The traditional AIX security model is often referred to as Discretionary Access Control (DAC). This approach is based on file permissions read/write/execute and designates a single user ID as the owner of a file. File permissions can be controlled by the root or the file owner. They can grant or deny privileges to owner/group/other based on the UID and GID.

This file system security model also applies within the limits of a WPAR. Because WPARs have completely independent user spaces, the scope of granting or denying privileges is restricted to WPAR boundaries. This implies that a WPAR file owner can grant or deny privileges only for files located in the WPAR and only to users local to the WPAR.

### 6.2.2 Access Control Lists

The base permissions which are handled by the DAC security model control the rights for the owner, the group, and all other users in the system. Access Control Lists allow a file owner to grant individual users and groups file permissions. This set of additional permissions, which extends beyond traditional base permissions, are named *extended permissions*. ACLs represent a way of specifying permissions for a file at a more granular level.

The extended set of permissions are kept in the file metadata section named Extended Attributes. This section is accessible from the WPAR space by the file owner and WPAR root. Global root has also access to this section of file metadata and can modify it at will.

Example 6-1 on page 153 presents a scenario that illustrates how the concept of ACL applies to a WPAR environment. It also shows implications upon data accessibility when ACL operations are performed from both WPAR and global environments as follows:

- User1wp from WPAR wpr03 owns a regular file named file_user1wp. The file has only read and write permissions for the owner. User2wp cannot access file_user1wp. file_user1wp has not any ACL associated.
- User1wp decides to use ACL to grant user2wp read access to file_user1wp, so user1wp modifies the ACL accordingly and applies it.
- Now user2wp has read access to file_user1wp.
- Global root is able to notice that file_user1wp has an ACL associated and decides to modify it so that user2wp has no longer read access.
- Now user2wp has no longer read access to file_user1wp.
- User1wp decides to grant user2wp read access again to file_user1wp, so user1wp again modifies the ACL and applies it.
- Now user2wp again has read access to file_user1wp.

*Example 6-1 Using Access Control Lists in WPAR environment*

```
user1wp@wpr03#ls -le /home/user1wp/file_user1wp
-rw--------    1 user1wp  staff           18 Jun 26 16:21
/home/user1wp/file_user1wp

user1wp@wpr03#aclget /home/user1wp/file_user1wp
*
* ACL_type   AIXC
*
attributes:
base permissions
    owner(user1wp):  rw-
    group(staff):  ---
    others:  ---
extended permissions
    disabled

user2wp@wpr03#cat /home/user1wp/file_user1wp
cat: 0652-050 Cannot open /home/user1wp/file_user1wp.

user1wp@wpr03#acledit /home/user1wp/file_user1wp
Should the modified ACL be applied? (yes) or (no) y
user1wp@wpr03#aclget /home/user1wp/file_user1wp
*
* ACL_type   AIXC
*
```

```
attributes:
base permissions
    owner(user1wp):  rw-
    group(staff):  ---
    others:  ---
extended permissions
    enabled
    permit   r--     u:user2wp

user2wp@wpr03#cat /home/user1wp/file_user1wp
11111111
22222222

root@lpar13#ls -le /wpars/wpr03/home/user1wp/file_user1wp
-rw-------+    1 user1    staff            18 Jun 26 16:21
/wpars/wpr03/home/user1wp/file_user1wp

root@lpar13#acledit /wpars/wpr03/home/user1wp/file_user1wp
Should the modified ACL be applied? (yes) or (no) y

root@lpar13#aclget /wpars/wpr03/home/user1wp/file_user1wp
*
* ACL_type   AIXC
*
attributes:
base permissions
    owner(user1):  rw-
    group(staff):  ---
    others:  ---
extended permissions
    enabled
    permit   ---     u:user1ge

user2wp@wpr03#cat /home/user1wp/file_user1wp
cat: 0652-050 Cannot open /home/user1wp/file_user1wp.

user1wp@wpr03#acledit /home/user1wp/file_user1wp
Should the modified ACL be applied? (yes) or (no) y
user1wp@wpr03#aclget /home/user1wp/file_user1wp
*
* ACL_type   AIXC
*
attributes:
base permissions
    owner(user1wp):  rw-
    group(staff):  ---
    others:  ---
extended permissions
    enabled
    permit   r--     u:user2wp


user2wp@wpr03#cat /home/user1wp/file_user1wp
11111111
22222222
```

### 6.2.3 Controlling file access

Apart from security provided by the access control mechanisms presented in the preceding sections, there is an additional degree of file system security that can be obtained by using cryptography as explained in detail in 6.3, “Encrypted File Systems” on page 156.

However, it is very important at this stage to understand how file protection mechanisms interact in terms of controlling file access permissions. These three mechanisms do not overlap. They cooperate and complement to offer even stricter control of file access.

If all three mechanisms are used, the security credentials of a user are verified as follows:

1. The Discretionary Access Control (DAC) level is used to determine whether the user is entitled to the required file permission according to the user/group/other paradigm. If the user does not get any of the traditional read/write/execute permissions and the file does not have an ACL, the user is denied access. If the file has either the required base permissions or an extended set of permissions, verification proceeds with next step.
2. The file Access Control List is checked for extended file access permissions. The combined analysis of base permissions and explicitly granted extended permissions determine whether the user is denied or granted the requested privilege. If the user has been granted a specific permission, the verification proceeds with next step. If not, the verification fails.
3. User security credentials in terms of cryptographic key are verified at this level. If the user has at least one matching valid key, the user is granted the requested privilege.

A more in-depth discussion, along with several practical examples of using encryption to strengthen WPAR security, is provided in 6.3, “Encrypted File Systems” on page 156.

## 6.3 Encrypted File Systems

When using Encrypted File Systems (EFS), each file is encrypted using the AES algorithm. EFS is designed so that each file is assigned a unique AES key named the *secret key*. The secret AES key is used to encrypt file data blocks when data is written on the disk, and to decrypt file data blocks when they are read from the disk. Thus, only someone who has the secret key can have access to data.

In EFS-enabled environments, each user is assigned a pair or RSA keys: a private and a public key.

When a user creates a file, a new AES file-specific secret key is generated. Data is encrypted with the secret key and written on the disk. The secret key is encrypted with the public key of the file owner and stored in the file extended attributes. This entity represents the file cryptographic metadata.

When the file owner tries to open the file, the owner first opens the file cryptographic metadata. Because the owner’s private key matches the owner’s public key that protects the secret key, the owner is able to get access to the secret key and then decrypt the file data.

When the legitimate file owner decides to grant another user access to the file data, the owner opens the file cryptographic metadata as explained before, encrypts the secret key with the new user public key, and adds it to the file cryptographic metadata.

### 6.3.1 Confidentiality of WPAR data

You can easily imagine various situations in which the WPAR administrator decides that WPAR local data should not be accessed from the global environment. The WPAR would still be dependent on the global environment from a operational perspective. However, the WPAR data should remain confidential in the sense that the information contained in the WPAR data files cannot be disclosed to an unauthorized user.

The practical scenario shown in Example 6-2 on page 157 is based on these assumptions:

- File system encryption is enabled at the global environment level.

- File system encryption is enabled at the WPAR level.
- An EFS-enabled file system named `fs` has been created in the global environment and mounted in into WPAR `wpr03`.
- The `userwp` user has been defined in the `wpr03` WPAR and the corresponding user keystore has been created using default settings.
- A plain text file named `file_userwp` owned by `userwp` has been created.
- Traditional DAC permissions have been set to provide proper access to files and directories.

By default, global root has full read/write access to any file. However, global root can be prevented from reading file information while still being able to have access to data contained in the file data blocks.

Example 6-2 illustrates how the regular WPAR user named `userwp` has ensured the confidentiality of its data:

- User `userwp` has no security credentials loaded in the current shell.
- A file named `file_userwp` is owned by userwp. The file is in clear format.
- The `userwp` has all DAC privileges and can access its own file. The file is in clear format.
- Global root has all required DAC privileges and can access `file_userwp`.
- The `userwp` decides to use encryption and loads its private key into the current shell.
- The `userwp` encrypts `file_userwp`. As shown by displaying metadata of `file_userwp`, only `userwp`'s private key can decrypt `file_userwp`.
- Global root cannot access `file_userwp` anymore.
- Global root becomes malicious and still wants to look at `file_userwp` and has all system tools available. Global root decides to use the powerful **`fsdb`**.
- Global root determines that inode 7 corresponds to `file_userwp`.
- Global root determines that `file_userwp` uses block `2f`.
- Global root displays block `2f` and finds out that file_userwp has been encrypted, so global root cannot read the data in the file.

*Example 6-2 WPAR-level encrypted files cannot be accessed from global environment*

```
userwp@wpr03#efskeymgr -V
There is no key loaded in the current process.

userwp@wpr03#ls -lU /fs/file_userwp
```

```
-rw-r--r---    1 userwp   staff            35 Jun 24 17:26
/fs/file_userwp

userwp@wpr03#cat /fs/file_userwp
1111111111111111
2222222222222222

root@lpar13#cat /wpars/wpr03/fs/file_userwp
1111111111111111
2222222222222222

userwp@wpr03#efskeymgr -o ksh
userwp's EFS password:

userwp@wpr03#efsmgr -e /fs/file_userwp

userwp@wpr03#efsmgr -l /fs/file_userwp
EFS File information:
 Algorithm: AES_128_CBC
List of keys that can open the file:
 Key #1:
  Algorithm       : RSA_1024
  Who             : uid 333
  Key fingerprint : c064ff7e:7d26b6a0:2b073e7b:62b9ba0e:4ab75b8b

root@lpar13#cat /wpars/wpr03/fs/file_userwp
cat: 0652-050 Cannot open /wpars/wpr03/fs/file_userwp.

root@lpar13#ls -liU /wpars/wpr03/fs/file_userwp
    7 -rw-r--r--e    1 333      staff            35 Jun 24 17:41
/wpars/wpr03/fs/file_userwp

root@lpar13#istat 7 /dev/fslv08
Inode 7 on device 10/30 File
Protection: rw-r--r--
Owner: 333(<UNKNOWN>)           Group: 1(staff)
Link count:   1         Length 35 bytes

Last updated:   Tue Jun 24 17:41:57 CDT 2008
Last modified:  Tue Jun 24 17:41:57 CDT 2008
Last accessed:  Tue Jun 24 17:41:57 CDT 2008

Block pointers (hexadecimal):
2f
```

```
root@lpar13#fsdb /dev/fslv08
Filesystem /dev/fslv08 is mounted.  Modification is not permitted.

File System:                    /dev/fslv08

File System Size:               261728  (512 byte blocks)
Aggregate Block Size:           4096
Allocation Group Size:          8192    (aggregate blocks)

> display 0x2f
Block: 47     Real Address 0x2f000
00000000:  FCED3525 7D66523B 8BBC2473 F8EA9FC9   |..5%}fR;..$s....|
00000010:  B90507EE D1D3F589 2AC9822E BF934403   |........*.....D.|
00000020:  4AD93C0C 1D2880B7 DBC95C23 A1D45E04   |J.<..(....\#..^.|
00000030:  6D78F1CE A15D0183 39B77036 FA9DC021   |mx...]..9.p6...!|
00000040:  B383D21A 77E63759 C27A530D 677F974E   |....w.7Y.zS.g..N|
00000050:  17078023 60B89D61 B27F566D 169C772D   |...#`..a..Vm..w-|
00000060:  2CD59C56 A15875D2 B96E833F 01E00737   |,..V.Xu..n.?...7|
00000070:  44B4B057 CA5398C2 A73284DC B64E950F   |D..W.S...2...N..|
00000080:  4A55698A A8DEFCDF 6FE2F472 DC4BE416   |JUi.....o..r.K..|
00000090:  83660AF4 A0A912E8 83F79233 892DAB91   |.f.........3.-..|
000000a0:  C8E0F282 DFFA60D4 3AAFBE43 92B2D150   |......`.:..C...P|
000000b0:  F22E8331 A8795E94 478D3E89 8DE5BCE5   |...1.y^.G.>.....|
000000c0:  7CD93301 4A3D1C3B FB617CB6 89314068   ||.3.J=.;.a|..1@h|
000000d0:  149E4208 B10E86C3 4CA6AD2B 63BA5C0B   |..B.....L..+c.\.|
000000e0:  FD569618 5AF03BEC BB3B8522 50AAFF99   |.V..Z.;..;."P...|
000000f0:  2186A8A9 0DEA5FA2 9B87395D 4B77DD5C   |!....._...9]Kw.\|
-hit enter for more-
```

This technique can be used to protect user data from a malicious global root. The information is also protected against local root because WPAR root cannot get access to privileges and tools available for global root.

## 6.3.2 Granting global environment users access to WPAR data

You can also imagine various situations in which certain global environment users can be granted access to WPAR local data. To do that, the WPAR data owner and global receiver must reach a mutual agreement. They also need global root cooperation. However, if the WPAR user decides to revoke access to WPAR local data, that user can do it at any time without the consent of the global root.

Imagine a practical scenario based on the following assumptions:

- File system encryption is enabled at the global environment level.
- File system encryption is enabled at the WPAR level.
- An EFS-enabled file system named fs has been created in the global environment and mounted in into WPAR wpr03.
- The userwp user in the wpr03 WPAR has been defined and the corresponding user keystore has been created using default settings. A global environment user will be granted access to data owned by userwp.
- The proxywp user in the local environment has been defined and the corresponding user keystore has been created using default settings. The proxywp user is simply a regular user that, in this scenario, acts as a proxy for granting and denying access to WPAR data. From the WPAR perspective, proxywp is the relay that conveys data access permissions to the global environment.
- The proxyge user was defined in the global environment and the corresponding user keystore was created using default settings. The proxyge user is also simply a regular user that, in this scenario, acts as proxy for receiving data access permissions on behalf of global users.
- An encrypted file was created named file_userwp, owned by userwp.
- Assume that traditional DAC permissions have been set to provide proper access to files and directories.

Example 6-3 on page 161 illustrates how global user `proxyge` has been granted and then revoked access to a file owned by the regular WPAR user named `userwp` as follows:

- `userwp` decides to grant `proxywp` access to `file_userwp`. `userwp` finds out that `proxywp` has the ID 555.
- `userwp` grants `proxywp` access to `file_userwp` and verifies that `file_userwp` metadata has been updated accordingly. `file_userwp` metadata shows that the private key of user `555` can access the file.
- global `root` replaces the `proxyge` keystore with the `proxywp` keystore. This operation requires global root privileges. WPAR root cannot perform this action.
- The `proxywp` keystore password is communicated to the `proxyge` user. This is a human activity and requires mutual agreement between the two users.
- `proxyge` logs in the global environment using its own login password. Because this login password is different from the replaced keystore password, `proxyge` has no security credentials loaded in the login shell.

- proxyge opens the replaced keystore, enters the password for proxywp, and loads the proxywp security credentials.
- At this point, proxyge is able to access file_userwp.
- userwp decides to revoke proxywp access to file_userwp.
- userwp verifies that file_userwp metadata has been updated accordingly. file_userwp metadata shows that the private key of user 555 has been removed.
- Now, proxyge is no longer able to access file_userwp.

*Example 6-3 Granting data access to global users*

```
userwp@wpr03#lsuser proxywp
proxywp id=555 pgrp=staff groups=staff home=/home/proxywp
shell=/usr/bin/ksh

userwp@wpr03#efsmgr -a file_userwp -u proxywp

userwp@wpr03#efsmgr -l file_userwp
EFS File information:
 Algorithm: AES_128_CBC
List of keys that can open the file:
 Key #1:
  Algorithm       : RSA_1024
  Who             : uid 222
  Key fingerprint : b514c3be:b4e2e03c:5022d5d1:b559c52d:520d664b
 Key #2:
  Algorithm       : RSA_1024
  Who             : uid 555
  Key fingerprint : 61fc731a:68598434:b98ad73c:fc3133ae:f374a061

root@lpar13#cp /wpars/wpr03/var/efs/users/proxywp/keystore
/var/efs/users/proxyge/keystore

proxyge@lpar13#efskeymgr -V
There is no key loaded in the current process.

proxyge@lpar13#id
uid=555(proxyge) gid=1(staff)

proxyge@lpar13#efskeymgr -o ksh
proxyge's EFS password:

proxyge@lpar13#efskeymgr -V
```

```
List of keys loaded in the current process:
 Key #0:
                           Kind ..................... User key
                           Id   (uid / gid) ......... 555
                           Type ..................... Private key
                           Algorithm ................ RSA_1024
                           Validity ................. Key is valid
                           Fingerprint ..............
61fc731a:68598434:b98ad73c:fc3133ae:f374a061


proxyge@lpar13#cat /wpars/wpr03/fs/file_userwp
1111111111111111
2222222222222222

userwp@wpr03#efsmgr -r file_userwp -u proxywp

userwp@wpr03#efsmgr -l file_userwp
EFS File information:
 Algorithm: AES_128_CBC
List of keys that can open the file:
 Key #1:
  Algorithm       : RSA_1024
  Who             : uid 222
  Key fingerprint : b514c3be:b4e2e03c:5022d5d1:b559c52d:520d664b

proxyge@lpar13#cat /wpars/wpr03/fs/file_userwp
cat: 0652-050 Cannot open /wpars/wpr03/fs/file_userwp.
```

This technique can be used when access to WPAR data needs to be granted to global users and WPAR still retains control over the data.

### 6.3.3 Granting WPAR users access to global environment data

There may be situations in which certain users of certain WPARs should granted access to global environment data.

To accomplish that, the WPAR data receiver and the global user must reach a mutual agreement. They also need global root cooperation. However, if the global user decides to revoke access to global data, the global user can do that at any time without the consent of the global root.

Imagine a practical scenario based on the following assumptions:

- File system encryption is enabled at the global environment level.
- File system encryption is enabled at the WPAR level.
- An EFS-enabled file system named fs has been created in the global environment and mounted in into WPAR wpr04.
- The global user has been defined in the global environment and the corresponding user keystore has been created using default settings. The global user will grant a WPAR user access to is data.
- The glagent user has been defined in the global environment and the corresponding user has been created keystore using default settings. glagent is simply a regular user which, in this scenario, acts as a proxy for granting and denying access to global data. From the global perspective, the glagent user is the relay that conveys data access permissions to the WPAR environment.
- The locagent user in the WPAR environment has been defined and the corresponding user keystore has been created using default settings. The locagent is also simply a regular user which, in this scenario, acts as proxy for receiving data access permissions on behalf of local users.
- An encrypted file has been created named global_file owned by global.
- Assume that traditional DAC permissions have been set to provide proper access to files and directories.

Example 6-4 on page 164 illustrates how global user `global` has granted and then revoked a WPAR user named `locagent` access to a file as follows:

- `global` decides to grant `glagent` access to `global_file`. `global` finds out that `glagent` has the ID `205`.
- `global` grants `glagent` access to `global_file` and verifies that `global_file` metadata has been updated accordingly. `global_file` metadata shows that the private key of user `205` can access the file.
- global `root` replaces the `locagent` keystore with the `glagent` keystore. This operation requires global root privileges. WPAR root cannot perform this action.
- `glagent` keystore password is communicated to the `locagent` user. This is a human activity and requires mutual agreement between the two users.
- `locagent` logs in the local environment using its own login password. Because its login password is different from the replaced keystore password, `locagent` has no security credentials loaded in the login shell.
- `locagent` opens the replaced keystore, enters the password for `glagent`, and loads the `glagent` security credentials.

- `locagent` is able to access `global_file`.
- `global` decides to revoke `glagent` access to `global_file`.
- `global` verifies that `global_file` metadata has been updated accordingly. `global_file` metadata shows that the private key of user 205 has been removed.
- `locagent` is no longer able to access `global_file`.

*Example 6-4 Grating local users access to global data*

```
global@lpar13#lsuser glagent
glagent id=205 pgrp=staff groups=staff home=/home/glagent
shell=/usr/bin/ksh

global@lpar13#efsmgr -a global_file -u glagent

global@lpar13#efsmgr -l global_file
EFS File information:
 Algorithm: AES_128_CBC
List of keys that can open the file:
 Key #1:
  Algorithm       : RSA_1024
  Who             : uid 204
  Key fingerprint : 3f77390a:ac67372e:d3868009:dc4b4cfb:3529dd68
 Key #2:
  Algorithm       : RSA_1024
  Who             : uid 205
  Key fingerprint : ab943a46:34d53a98:35aae495:c42063b5:4954b9fb

root@lpar13#cp /var/efs/users/glagent/keystore
/wpars/wpr04/var/efs/users/locagent/keystore

locagent@wpr04#efskeymgr -V
There is no key loaded in the current process.

locagent@wpr04#efskeymgr -o ksh
locagent's EFS password:

locagent@wpr04#efskeymgr -V
List of keys loaded in the current process:
 Key #0:
                           Kind ..................... User key
                           Id   (uid / gid) ......... 205
                           Type ..................... Private key
                           Algorithm ................ RSA_1024
                           Validity ................. Key is valid
                           Fingerprint ..............
ab943a46:34d53a98:35aae495:c42063b5:4954b9fb

locagent@wpr04#cat /fs/global_file
11111111
22222222


global@lpar13#efsmgr -r global_file -u glagent

global@lpar13#efsmgr -l global_file
EFS File information:
 Algorithm: AES_128_CBC
List of keys that can open the file:
 Key #1:
  Algorithm       : RSA_1024
  Who             : uid 204
  Key fingerprint : 3f77390a:ac67372e:d3868009:dc4b4cfb:3529dd68

locagent@wpr04#cat /fs/global_file
cat: 0652-050 Cannot open /fs/global_file.
```

This technique can be used when access to global data needs to be granted to local users and global environment still retains control over the data.

## 6.4 Users and groups

To understand user and group management features related to WPAR, you must first understand the nature of WPARs as they relate to users.

Application WPARs are created mostly for running a specific application, so they usually share the user space and security context with the global environment.

On the other hand, system WPARs are intended to behave like independent environments. They inherit the traditional AIX management tasks, control features, and security attributes with respect to both users and groups, so all user and group management activities are similar to those performed on stand-alone AIX systems.

However, the users set and groups set of a WPAR are completely different from the users set and groups set existing in the global environment or in another WPAR. With respect to users and groups, you can think of a WPAR as a completely independent system. Even in the case when a user from the WPAR environment has ID, password and security attributes identical to those of a user from the global environment, they are still completely *separate* entities.

Each WPAR has its own security-related files and settings. This means that the /etc/passwd file from the global environment has nothing to do with the /etc/passwd file from a WPAR.

Each WPAR set of users and groups can be administered either locally using the local security repository, or remotely through a centralized network-based repository, such as NIS, NIS+, or LDAP with the RFC2307 schema.

### 6.4.1 Security credentials WPAR environments

Applications running within a WPAR derive their security credentials from the rights of the local WPAR users and groups. Applications running in EFS-enabled environments obtain their security privileges using *both* AIX traditional discretionary access control mechanism and security credentials contained in user and group keystores.

This section demonstrates how EFS-related features interact in WPAR environments, and how they can be used to strengthen the security. A prior understanding of concepts regarding EFS is required. For more detailed information about this topic, refer to *AIX V6 Advanced Security Features Introduction and Configuration*, SG24-7430.

#### User security credentials at WPAR level

Just as on stand-alone AIX systems, in the case of WPARs or the global environment being EFS-enabled, there is no connection whatsoever between the key contained in user and groups keystores.

Example 6-5 illustrates the user named `user1`, which has identical ID and security attributes in the global environment and in two different system partitions. Each private key fingerprint is different, which means that the private keys themselves are different.

*Example 6-5 Security repositories are completely independent*

```
$ uname -n
lpar13
```

```
$ cat /etc/passwd|grep user1
user1:!:222:1::/home/user1:/usr/bin/ksh

$ lsuser -a -f efs_initialks_mode efs_keystore_algo efs_keystore_access
efs_adminks_access efs_allowksmodechangebyuser efs_file_algo user1
user1:
        efs_initialks_mode=admin
        efs_keystore_algo=RSA_1024
        efs_keystore_access=file
        efs_adminks_access=file
        efs_allowksmodechangebyuser=yes
        efs_file_algo=AES_128_CBC

$ efskeymgr -v
 Keystore content:
                             Keystore owner ............ : uid 222
                             Keystore mode ............. : admin:
managed by EFS administrator
                             Password changed last on .. : 06/19/08 at
14:35:55
    Private key:
          Algorithm : RSA_1024
        Fingerprint : af0dfb52:f7fc349f:97b5d29a:d22be9dd:408ba8a0
           Validity : This key is valid.

$ uname -n
wpr03

$  cat /etc/passwd|grep user1
user1:!:222:1::/home/user1:/usr/bin/ksh

$ lsuser -a -f efs_initialks_mode efs_keystore_algo efs_keystore_access
efs_adminks_access efs_allowksmodechangebyuser efs_file_algo user1
user1:
        efs_initialks_mode=admin
        efs_keystore_algo=RSA_1024
        efs_keystore_access=file
        efs_adminks_access=file
        efs_allowksmodechangebyuser=yes
        efs_file_algo=AES_128_CBC

$ efskeymgr -v
 Keystore content:
                             Keystore owner ............ : uid 222
                             Keystore mode ............ : admin:
managed by EFS administrator
                             Password changed last on .. : 06/19/08 at
14:36:17
    Private key:
          Algorithm : RSA_1024
        Fingerprint : 9b23ef4b:82d28d7b:a51af65a:77059cff:422dc2d3
           Validity : This key is valid.

$ uname -n
wpr04

$ cat /etc/passwd|grep user1
user1:!:222:1::/home/user1:/usr/bin/ksh

$ lsuser -a -f efs_initialks_mode efs_keystore_algo efs_keystore_access
efs_adminks_access efs_allowksmodechang>
user1:
        efs_initialks_mode=admin
        efs_keystore_algo=RSA_1024
        efs_keystore_access=file
        efs_adminks_access=file
        efs_allowksmodechangebyuser=yes
        efs_file_algo=AES_128_CBC

$ efskeymgr -v
 Keystore content:
                             Keystore owner ........... : uid 222
                             Keystore mode ............ : admin:
managed by EFS administrator
                             Password changed last on .. : 06/19/08 at
15:01:42
    Private key:
          Algorithm : RSA_1024
        Fingerprint : 3e08bf69:675eb594:904c3207:6750d420:c63c641a
           Validity : This key is valid.
```

## Application security credentials at WPAR level

Applications that access data stored in EFS-enabled environments must have AIX traditional read, write, and execute permissions. They must *also* have a private key obtained from a user or group keystore. Data access is not allowed if the corresponding private key is not present in the context of a process.

Getting access to a keystore that contains the proper key is, therefore, *mandatory*. However, being able to easily reset the user login password does *not* mean getting access to the user keystore.

Example 6-6 illustrates how the login password and keystore password are independent. It also demonstrates how to open a keystore and associate security credentials to an individual process, as follows:

- `root` user on `wpr03` WPAR is logged on and has its security credentials loaded into the current shell.
- `user1` on `wpr03` WPAR is logged on and has its security credentials loaded into the current shell.
- `root` user on the `lpar13` global environment is logged on and has its security credentials loaded into the current shell.
- `root` user on the `lpar13` global environment generates a shell into the `wpr03` WPAR and then resets the login password for both `root` and `user1` from the `wpr03` WPAR.
- However, the shell executed on the WPAR retained the security credentials of the global `root` user.
- `root` user on `wpr03` WPAR logs in again. Because it login password is now different from its keystore password, its security credentials are no longer automatically loaded into the current shell. When `root` tries to associated its security credentials with a new shell, `root` is prompted for the keystore password. After `root` enters the keystore password, then its security credentials are loaded in the new shell.
- `user1` on `wpr03` WPAR logs in again. As usual after password reset, `user1` is prompted to change its login password. `user1` changes it to something different from its keystore password. Because the `user1` login password is different from its keystore password, that user's security credentials are no longer automatically loaded into the current shell. When `user1` tries to associated its security credentials with a new shell, that user is prompted for the keystore password. After `user1` enters the keystore password, that user's security credentials are loaded in the new shell.

*Example 6-6 Resetting a user root password does not affect the keystore password*

```
# uname -n
wpr03

# 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 ..............
b9ebab7e:8ea213c2:8b8c3198:bbe1bc19:053d92e5
 Key #1:
                           Kind ..................... Group key
                           Id   (uid / gid) ......... 7
                           Type ..................... Private key
                           Algorithm ................ RSA_1024
                           Validity ................. Key is valid
                           Fingerprint ..............
293049b3:bb86eabe:a395491f:4fd4bf6e:e8623693
 Key #2:
                           Kind ..................... Admin key
                           Id   (uid / gid) ......... 0
                           Type ..................... Private key
                           Algorithm ................ RSA_1024
                           Validity ................. Key is valid
                           Fingerprint ..............
8d4f18a9:892df405:aa148860:8806168e:8f1578ca

$ uname -n
wpr03

$ efskeymgr -V
List of keys loaded in the current process:
 Key #0:
                           Kind ..................... User key
                           Id   (uid / gid) ......... 222
                           Type ..................... Private key
                           Algorithm ................ RSA_1024
                           Validity ................. Key is valid
                           Fingerprint ..............
9b23ef4b:82d28d7b:a51af65a:77059cff:422dc2d3

# 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 .............
de525a00:51eab2db:b9a0fbf6:f5dafdc2:e8f0d636
 Key #1:
                          Kind ..................... Group key
                          Id   (uid / gid) ......... 7
                          Type ..................... Private key
                          Algorithm ................ RSA_1024
                          Validity ................. Key is valid
                          Fingerprint .............
57b4875b:f946bc2a:0e678cde:654ee6c4:d9f01257
 Key #2:
                          Kind ..................... Admin key
                          Id   (uid / gid) ......... 0
                          Type ..................... Private key
                          Algorithm ................ RSA_1024
                          Validity ................. Key is valid
                          Fingerprint .............
4b2d59b0:2ad4813f:baafa7dc:349143c3:177c3afe

# chroot /wpars/wpr03 /bin/ksh

# passwd root
Changing password for "root"
root's New password:
Enter the new password again:

# passwd user1
Changing password for "user1"
user1's New password:
Enter the new password again:

# 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 .............
de525a00:51eab2db:b9a0fbf6:f5dafdc2:e8f0d636
 Key #1:
                          Kind ..................... Group key
                          Id   (uid / gid) ......... 7
                          Type ..................... Private key
                          Algorithm ................ RSA_1024
                          Validity ................. Key is valid
                          Fingerprint ..............
57b4875b:f946bc2a:0e678cde:654ee6c4:d9f01257
 Key #2:
                          Kind ..................... Admin key
                          Id   (uid / gid) ......... 0
                          Type ..................... Private key
                          Algorithm ................ RSA_1024
                          Validity ................. Key is valid
                          Fingerprint ..............
4b2d59b0:2ad4813f:baafa7dc:349143c3:177c3afe

# uname -n
wpr03

# efskeymgr -V
There is no key loaded in the current process.

# efskeymgr -o ksh
root's EFS password:

# 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 ..............
b9ebab7e:8ea213c2:8b8c3198:bbe1bc19:053d92e5
 Key #1:
                          Kind ..................... Group key
                          Id   (uid / gid) ......... 7
                          Type ..................... Private key
                          Algorithm ................ RSA_1024
                          Validity ................. Key is valid
                          Fingerprint ..............
293049b3:bb86eabe:a395491f:4fd4bf6e:e8623693
 Key #2:
                          Kind ..................... Admin key
                          Id   (uid / gid) ......... 0
                          Type ..................... Private key
                          Algorithm ................ RSA_1024
```

```
                           Validity ................ Key is valid
                           Fingerprint .............
8d4f18a9:892df405:aa148860:8806168e:8f1578ca



AIX Version 6
Copyright IBM Corporation, 1982, 2008.
login: user1
user1's Password:
[compat]: 3004-610 You are required to change your password.
        Please choose a new one.

user1's New password:
Enter the new password again:
***********************************************************************
********
*
*
*
*
*  Welcome to AIX Version 6.1!
*
*
*
*
*
*  Please see the README file in /usr/lpp/bos for information pertinent
to    *
*  this release of the AIX Operating System.
*
*
*
*
*
***********************************************************************
********
Last login: Thu Jun 19 17:54:52 CDT 2008 on /dev/pts/0 from 9.3.4.130

$ uname -n
wpr03

$ efskeymgr -V
There is no key loaded in the current process.
```

```
$ efskeymgr -o ksh
user1's EFS password:

$ efskeymgr -V
List of keys loaded in the current process:
 Key #0:
                           Kind ..................... User key
                           Id   (uid / gid) ......... 222
                           Type ..................... Private key
                           Algorithm ................ RSA_1024
                           Validity ................. Key is valid
                           Fingerprint .............
9b23ef4b:82d28d7b:a51af65a:77059cff:422dc2d3
```

---

**Tip**: In practice, login passwords are frequently identical to keystore passwords. In this case, when a user logs in, that user's security credentials are automatically associated to its processes. This has two implications:

- Using this method ensures seamless integration of security privileges with installed applications and all usual AIX commands. Users and applications may not even be aware of the EFS-related features they are using. Often, application-specific users such as `db2admin` do not have human identities associated with them.

  In case of disaster recovery, using this method means that global root knows and can reset both the login and keystore passwords, thereby regaining access to user files. However, it also means that the WPAR is not completely independent and its data can be accessed from outside the WPAR.
- Not using this method implies that global root will be unable to reset the keystore password or recover encrypted user files if the keystore password is lost. On the other hand, WPAR data is protected from any access outside the WPAR.

### Recovering user access to an EFS-enabled WPAR

As stated, user data is protected by *both* the login password and the user keystore password. So you may wonder how to regain access to data if these credentials are lost.

As described in 4.5.2, "Recovering a password for users inside WPARs" on page 87, you can recover from a lost WPAR root password and reset a regular WPAR user password. Following that procedure, the global root is able to regain complete control over a WPAR.

As mentioned, in EFS-enabled environments users have keystores, apart from the login passwords. To restore control to a WPAR, global root should also gain access to the user keystore, which requires the keystore access password.

This scenario is shown in Example 6-7:

- The global root from `lpar13` has all global security credentials loaded.
- The newly-spawned shell (just as in 4.5.2, "Recovering a password for users inside WPARs" on page 87) is owned by the WPAR root.
- The newly-spawned shell does *not* have any WPAR security credentials associated with it.
- The global root is able to successfully change the WPAR root password.
- Because the newly-spawned shell does *not* have the WPAR required security credentials, it fails to open the WPAR root keystore.

*Example 6-7 Resetting the root password does not affect the keystore password*

```
root@lpar13#id
uid=0(root) gid=0(system)
groups=2(bin),3(sys),7(security),8(cron),10(audit),11(lp)

root@lpar13#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 ..............
de525a00:51eab2db:b9a0fbf6:f5dafdc2:e8f0d636
 Key #1:
                           Kind ..................... Group key
                           Id   (uid / gid) ......... 7
                           Type ..................... Private key
                           Algorithm ................ RSA_1024
                           Validity ................. Key is valid
                           Fingerprint ..............
57b4875b:f946bc2a:0e678cde:654ee6c4:d9f01257
 Key #2:
                           Kind ..................... Admin key
                           Id   (uid / gid) ......... 0
                           Type ..................... Private key
                           Algorithm ................ RSA_1024
                           Validity ................. Key is valid
                           Fingerprint ..............
4b2d59b0:2ad4813f:baafa7dc:349143c3:177c3afe

root@lpar13#lswpar
Name   State  Type  Hostname  Directory
-------------------------------------------
wpr03  A      S     wpr03     /wpars/wpr03
wpr04  A      S     wpr04     /wpars/wpr04

root@lpar13#chroot /wpars/wpr03 /bin/ksh

root@lpar13#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 ..............
de525a00:51eab2db:b9a0fbf6:f5dafdc2:e8f0d636
 Key #1:
                           Kind ..................... Group key
                           Id   (uid / gid) ......... 7
                           Type ..................... Private key
                           Algorithm ................ RSA_1024
                           Validity ................. Key is valid
                           Fingerprint ..............
57b4875b:f946bc2a:0e678cde:654ee6c4:d9f01257
 Key #2:
                           Kind ..................... Admin key
                           Id   (uid / gid) ......... 0
                           Type ..................... Private key
                           Algorithm ................ RSA_1024
                           Validity ................. Key is valid
                           Fingerprint ..............
4b2d59b0:2ad4813f:baafa7dc:349143c3:177c3afe

root@lpar13#id
uid=0(root) gid=0(system)
groups=2(bin),3(sys),7(security),8(cron),10(audit),11(lp)

root@lpar13#passwd
Changing password for "root"
```

```
root's New password:
Enter the new password again:

root@lpar13#efskeymgr -v
root's EFS password:
Encryption framework returned an error: Read keystore failed
```

Because the global root cannot reset the WPAR root keystore password, it will never get access to WPAR root security credentials and will never be able to execute any operation that would require those credentials.

## 6.5 RBAC in WPAR environments

The traditional AIX approach of granting privileges to users relies on a discretionary security model. According to this model, every file is an individual entity that has a single owner and a set of permissions (read, write, and execute).

The file owner can give file privileges to either a whole group of which the owner is a member, or to a group consisting of all other system users. The only exception to this rule is the root superuser, which can bypass this mechanism at its discretion.

AIX 6 introduced Enhanced RBAC as a security mechanism. Enhanced RBAC allows individual users to perform specific administrative tasks which, in a standard UNIX approach, would require root privileges. This is a granular approach, as opposed to the traditional approach. It can be achieved by assigning certain roles and authorizations to individual user accounts.

In addition, the set of privileges that are usually assigned to a command can be further reduced to the minimum subset of privileges required to run that command. Thus, different levels of functionality can be assigned to different classes of users.

For more in-depth information regarding RBAC and roles, authorizations and privileges, refer to*AIX V6 Advanced Security Features Introduction and Configuration*, SG24-7430.

RBAC is activated by default in Enhanced Mode when the operating system is installed.

In a WPAR environment, the RBAC mode for the system can be configured only from the global system. It will affect simultaneously the global environment and all WPARs.

Enhanced RBAC is supported only on system WPARs.

## 6.5.1 Using RBAC to secure WPAR operations

This section illustrates how RBAC features can be used to implement strict control over usual operational procedures in a WPAR environment that uses RBAC features.

Securing an operational procedure involves the following:

- Determining the set of privileges required to use the procedure
- Creating a mechanism that allows granting and revoking the credentials required for users to use the procedure
- Ensuring that users which have not been explicitly granted the set of required privileges cannot execute the procedure

The scenario presented here demonstrates how to secure a daily routine operation such as starting and stopping a WPAR. Similar actions can be performed with respect to all WPAR-related activities.

Starting a WPAR requires access to a single system command, **`startwpar`**. Stopping a WPAR requires access to a single system command, **`stopwpar`**.

In RBAC-specific terms, this entails the following tasks:

- Determine the set of authorizations required to invoke both **`startwpar`** and **`stopwpar`** commands.
- Define a role that includes those privileges.
- Ensure the role is unique.

RBAC-specific concepts are implemented as follows:

- Determine the full path of the commands to include in the role as shown in Example 6-8.

*Example 6-8 Determining the full path of WPAR-related commands*

```
# find / -name "*startwpar*"
/usr/sbin/startwpar
/usr/share/man/info/EN_US/a_doc_lib/cmds/aixcmds5/startwpar.htm
/export/nim/spot/610spot_res/usr/sbin/startwpar
```

```
# find / -name "*stopwpar*"
/usr/sbin/stopwpar
/usr/share/man/info/EN_US/a_doc_lib/cmds/aixcmds5/stopwpar.htm
/export/nim/spot/610spot_res/usr/sbin/stopwpar
```

- Determine all system authorizations required for these commands as shown in Example 6-9. Three authorizations are required to run **startwpar** and **stopwpar** commands.

*Example 6-9 Determining startwpar and stopwpar authorizations*

```
# lssecattr -c /usr/sbin/startwpar
/usr/sbin/startwpar euid=0 egid=0
accessauths=aix.wpar.owner,aix.wpar.system.start
innateprivs=PV_AZ_ROOT,PV_DAC_,PV_PROC_PRIV,PV_SU_UID
inheritprivs=PV_AZ_ADMIN,PV_AZ_CHECK,PV_DAC_,PV_AZ_ROOT,PV_KER_
secflags=FSF_EPS

# lssecattr -c /usr/sbin/stopwpar
/usr/sbin/stopwpar accessauths=aix.wpar.owner,aix.wpar.system.stop
innateprivs=PV_AZ_ROOT,PV_DAC_O,PV_DAC_R,PV_DAC_X,PV_PROC_PRIV,PV_TCB
inheritprivs=PV_AU_ADMIN,PV_AZ_CHECK,PV_AZ_ROOT,PV_DAC_O,PV_DAC_R,PV_DA
C_W,PV_DAC_X,PV_DEV_CONFIG,PV_DEV_LOAD,PV_DEV_QUERY,PV_FS_CHOWN,PV_FS_M
OUNT,PV_KER_ACCT,PV_KER_DR,PV_KER_EXTCONF,PV_KER_RAC,PV_KER_VARS,PV_KER
_WLM,PV_KER_WPAR,PV_NET_CNTL,PV_NET_PORT,PV_PROC_PRIV,PV_PROC_RAC,PV_PR
OC_SIG,PV_SU_UID,PV_TCB,PV_FS_MKNOD secflags=FSF_EPS
```

- Determine the roles that include the authorizations identified, as shown in Example 6-10. All authorizations are included in the `SysConfig` role.

*Example 6-10 Determining the roles that include required authorizations*

```
# lsauth -a roles aix.wpar.owner
aix.wpar.owner roles=SysConfig

# lsauth -a roles aix.wpar.system.start
aix.wpar.system.start roles=SysConfig

# lsauth -a roles aix.wpar.system.stop
aix.wpar.system.stop roles=SysConfig
```

- Create the `wpar-operator` user-defined role that will include the authorizations previously determined as shown in Example 6-11’

*Example 6-11 Creating wpar-operator user-defined role*

```
# mkrole
authorizations="aix.wpar.owner,aix.wpar.system.start,aix.wpar.system.st
op" dfltmsg="WPAR-Op>

# lsrole wpar-operator
wpar-operator
authorizations=aix.wpar.owner,aix.wpar.system.start,aix.wpar.system.sto
p rolelist= groups= visibility=1 screens=* dfltmsg=WPAR-Operator
msgcat= auth_mode=INVOKER id=11
```

- As demonstrated in Example 6-12, create the `operator` user, assign it the `wpar-operator` role, and update the Kernel Security Tables. After this is complete, then user `operator` can activate the role `wpar-operator` and will be assigned the expected authorizations.

*Example 6-12 Creating the wpar-operator user and assigning the wpar-operator role*

```
# mkuser operator

# lsuser -a roles operator
operator roles=

# chuser roles=wpar-operator operator

# lsuser -a roles operator
operator roles=wpar-operator

# setkst
Successfully updated the Kernel Authorization Table.
Successfully updated the Kernel Role Table.
Successfully updated the Kernel Command Table.
Successfully updated the Kernel Device Table.

$ swrole wpar-operator
operator's Password:

$ rolelist -ea
wpar-operator   aix.wpar.owner
```

```
                aix.wpar.system.start
                aix.wpar.system.stop
```

- Example 6-13 illustrates that at this point, there are two roles that contain the three authorizations. Now, remove them from the `SysConfig` role.

  Note that this could have been done at any time in this scenario. However, performing this task at this stage ensures that at least one role having those three privileges is available at all times.

*Example 6-13 Removing the three authorizations from the Sys Config role*

```
# lsauth -a roles aix.wpar.owner
aix.wpar.owner roles=SysConfig,wpar-operator

# lsauth -a roles aix.wpar.owner
aix.wpar.owner roles=SysConfig,wpar-operator

# lsauth -a roles aix.wpar.system.start
aix.wpar.system.start roles=SysConfig,wpar-operator

# lsauth -a roles aix.wpar.system.stop
aix.wpar.system.stop roles=SysConfig,wpar-operator

# lsrole -a authorizations SysConfig
SysConfig
authorizations=aix.system.boot.create,aix.system.config.bindintcpu,aix.
system.config.console,aix.system.config.date,aix.system.config.diag,aix
.system.config.dlpar,aix.system.config.inittab,aix.system.config.io,aix
.system.config.kext,aix.system.config.mode,aix.system.config.perf,aix.s
ystem.config.rset,aix.system.config.uname,aix.system.config.write,aix.s
ystem.stat,aix.wpar

# chrole
authorizations=aix.system.boot.create,aix.system.config.bindintcpu,aix.
system.config.console,aix.system.config.date,aix.system.config.diag,aix
.system.config.dlpar,aix.system.config.inittab,aix.system.config.io,aix
.system.config.kext,aix.system.config.mode,aix.system.config.perf,aix.s
ystem.config.rset,aix.system.config.uname,aix.system.config.write,aix.s
ystem.stat SysConfig

# lsrole -a authorizations SysConfig
SysConfig
authorizations=aix.system.boot.create,aix.system.config.bindintcpu,aix.
```

```
system.config.console,aix.system.config.date,aix.system.config.diag,aix
.system.config.dlpar,aix.system.config.inittab,aix.system.config.io,aix
.system.config.kext,aix.system.config.mode,aix.system.config.perf,aix.s
ystem.config.rset,aix.system.config.uname,aix.system.config.write,aix.s
ystem.stat
```

- The final step shows that the `wpar-operator` role is the only role that includes `aix.wpar.owner`, `aix.wpar.system.start` and `aix.wpar.system.stop` authorizations. Only users that switched to this role can perform the WPAR-specific operations that required those authorizations.

  `user1` successfully switches to the `SysConfig` role. However, this role no longer contains the authorizations required to start a WPAR. Therefore, `user1` is denied the permission to execute the **`startwpar`** command.

*Example 6-14 wpar-operator is the only role that contains the three initial authorizations*

```
# lsauth -a roles aix.wpar.owner
aix.wpar.owner roles=wpar-operator

# lsauth -a roles aix.wpar.system.start
aix.wpar.system.start roles=wpar-operator

# lsauth -a roles aix.wpar.system.stop
aix.wpar.system.stop roles=wpar-operator

$ swrole SysConfig
user1's Password:

$ rolelist -ea
SysConfig       aix.system.boot.create
                aix.system.config.bindintcpu
                aix.system.config.console
                aix.system.config.date
                aix.system.config.diag
                aix.system.config.dlpar
                aix.system.config.inittab
                aix.system.config.io
                aix.system.config.kext
                aix.system.config.mode
                aix.system.config.perf
                aix.system.config.rset
                aix.system.config.uname
                aix.system.config.write
```

```
               aix.system.stat

$ startwpar
ksh: startwpar: 0403-006 Execute permission denied.
```

## 6.6 Network isolation

In highly-secured environments there may be situations where it is desirable to isolate WPAR network traffic due to specific network security requirements. For instance, traffic originating from a WPAR may be required to follow a certain secured route and pass through a firewall to enter a trusted network.

The key point in understanding network traffic isolation is to understand how network traffic flows between WPARs on the same global environment; between the global environment and one of its WPARs; or between a WPAR and an external system.

Each WPAR can use the routing table available in the global environment. However, the WPAR administrator can decide to enable a WPAR local routing table, add and delete routes as desired thereby deciding the routing path to be followed by the traffic originating from the WPAR.

### 6.6.1 Using the global environment routing table

When an WPAR interface address is configured during WPAR startup, a host route for the WPAR IP address is added to the global routing table. The subnet route and broadcast route that correspond to the WPAR IP address are computed and added to the global routing table.

Because the WPAR IP address and the global environment IP address belong to the same subnet, the subnet and broadcast route corresponding to the WPAR IP address are identical to those corresponding to the global environment IP address.

Example 6-15 illustrates an example of a routing table in a global environment that contains two WPARs. Notice that the host routes associated with the IP addresses of the two WPARs point to the loopback interface.

*Example 6-15 Example of routing table when WPAR-specific routing is disabled*

```
root@lpar13#lswpar
Name   State  Type  Hostname  Directory
```

```
-------------------------------------------
wpr03  A      S     wpr03     /wpars/wpr03
wpr04  A      S     wpr04     /wpars/wpr04

root@lpar13#lswpar -L wpr03|grep -i rout
WPAR-Specific Routing:   no
USER-SPECIFIED ROUTES

root@lpar13#lswpar -L wpr04|grep -i rout
WPAR-Specific Routing:   no
USER-SPECIFIED ROUTES

root@lpar13#lswpar -L wpr03|grep -ip address
NETWORK
Interface     Address(6)        Mask/Prefix       Broadcast
-----------------------------------------------------------------
en0           9.3.5.185         255.255.254.0     9.3.5.255

root@lpar13#lswpar -L wpr04|grep -ip address
NETWORK
Interface     Address(6)        Mask/Prefix       Broadcast
-----------------------------------------------------------------
en0           9.3.5.186         255.255.254.0     9.3.5.255

root@lpar13#netstat -rn
Routing tables
Destination        Gateway           Flags   Refs     Use  If   Exp  Groups

Route Tree for Protocol Family 2 (Internet):
default            9.3.4.1           UG        0       10 en0      -      -
9.3.4.0            9.3.5.8           UHSb      0        0 en0      -      -   =>
9.3.4/23           9.3.5.8           U         4     9868 en0      -      -
9.3.5.8            127.0.0.1         UGHS      0       11 lo0      -      -
9.3.5.185          127.0.0.1         UGHS      0        0 lo0      -      -
9.3.5.186          127.0.0.1         UGHS      0        0 lo0      -      -
9.3.5.255          9.3.5.8           UHSb      0        5 en0      -      -
127/8              127.0.0.1         U        23     3463 lo0      -      -

Route Tree for Protocol Family 24 (Internet v6):
::1                ::1               UH        0      182 lo0      -      -
```

The routing table on the global system is accessible from all WPARs, including the host routes associated with WPAR IP addresses.

When one WPAR intends to communicate with another WPAR, it looks up the corresponding destination address in the routing table. There, it determines that the longest match, which is 32 bits in length, points to loopback interface `lo0`. So all network traffic between any two WPARs is routed via the loopback interface.

When a WPAR intends to communicate with the global environment, the same logic applies so all traffic will also be routed via the loopback interface.

When a WPAR intends to communicate with an external system for which there is a specific (sub)network route, it will use the gateway associated with that route. The gateway will be accessible via one of the global environment interfaces which are, in turn, accessible via the loopback interface.

When a WPAR intends to communicate with an external system for which there is no specific route in the routing table, it tries to use a default gateway. The default gateway is accessible via one of the global environment interfaces which are, in turn, accessible via the loopback interface.

You can conclude that all WPAR traffic is routed via the loopback interface. Although this behavior facilitates faster communication, it does not provide any control over routing at the WPAR level. To overcome this issue, a new feature has been implemented that enables each WPAR to use its own routing table rather than share the global environment routing table.

### 6.6.2 Using the WPAR-specific routing table

To allow each WPAR to decide the routing path for its own traffic, two changes have been implemented:

- The WPAR ID has been added to the routes of a WPAR.
- The routing lookup algorithm has been enhanced to take advantage of this new feature.

To help you understand the effect of these changes, this section returns to the scenario displayed in Example 6-15 on page 183 and explains how to enable WPAR-specific routing.

Just as in that example, when an WPAR interface address is configured during WPAR startup a host route, a subnet route, and a broadcast route that correspond to the WPAR IP address are computed and added to the global routing table.

However, all these routes are marked as belonging to the WPAR and will represent different entries in the global environment routing table as shown in

Example 6-16. Routes belonging to the global environment will be marked as global.

*Example 6-16 Example of routing table when WPAR-specific routing is enabled*

```
root@lpar13#netstat -rn@
Routing tables
WPAR    Destination       Gateway          Flags   Refs     Use  If   Exp  Groups

Route Tree for Protocol Family 2 (Internet):
Global  default           9.3.4.1          UG       0        10 en0      -      -
Global  9.3.4.0          9.3.5.8          UHSb     0         0 en0     -      -
=>
Global  9.3.4/23         9.3.5.8          U        4     26915 en0     -      -
=>
wpr03   9.3.4/23         9.3.5.185        U        1      4154 en0     -      -
=>
wpr04   9.3.4/23          9.3.5.186        U        0         4 en0      -      -
Global  9.3.5.8           127.0.0.1        UGHS     2        17 lo0      -      -
wpr03   9.3.5.185         127.0.0.1        UGHS     0         0 lo0      -      -
wpr04   9.3.5.186         127.0.0.1        UGHS     0         0 lo0      -      -
Global  9.3.5.255        9.3.5.8          UHSb     0         5 en0     -      -
=>
wpr03   9.3.5.255        9.3.5.185        UHSb     0         0 en0     -      -
=>
wpr04   9.3.5.255         9.3.5.186        UHSb     0         0 en0      -      -
Global  127/8             127.0.0.1        U       17      5382 lo0      -      -

Route Tree for Protocol Family 24 (Internet v6):
Global  ::1               ::1              UH       0       426 lo0      -      -
```

When a WPAR intends to communicate with another WPAR, it resorts to the enhanced routing look-up algorithm. That is, it looks up the destination address longest match that has been marked as *belonging* to the WPAR.

First, it tries to find a host route. If a host route is not available, it tries to find a subnet/network route. If a subnet/network route is not available, it tries to find a default route. Finally, if a default route is not found, the enhanced lookup algorithm fails.

When WPAR-specific routing is enabled, the WPAR has access only to routes marked as belonging to that WPAR. All other routes associated either with the global environment or with a different WPAR are not accessible. In this case, each WPAR can be thought of as having its own routing table. The routing table of WPAR wpr03 can be displayed from the global environment, as illustrated in Example 6-17.

*Example 6-17 Routing table of a WPAR*

```
root@lpar13#netstat -rn@ wpr03
Routing tables
WPAR    Destination        Gateway           Flags    Refs     Use   If    Exp  Groups

Route Tree for Protocol Family 2 (Internet):
wpr03   9.3.4/23          9.3.5.185         U         1     14850 en0      -      -
=>
wpr03   9.3.5.185          127.0.0.1         UGHS      0         0 lo0      -      -
wpr03   9.3.5.255         9.3.5.185         UHSb      0         0 en0      -      -
=>

Route Tree for Protocol Family 24 (Internet v6):
```

Next, to understand how WPAR-specific routing affects network traffic flow, the results shown in Example 6-18 are analyzed in this section.

*Example 6-18 Understanding WPAR-specific routing*

```
root@wpr03#ping 9.3.5.186
PING 9.3.5.186: (9.3.5.186): 56 data bytes
64 bytes from 9.3.5.186: icmp_seq=0 ttl=255 time=0 ms
64 bytes from 9.3.5.186: icmp_seq=1 ttl=255 time=0 ms
^C
----9.3.5.186 PING Statistics----
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0/0/0 ms

root@wpr03#ping 9.3.5.8
PING 9.3.5.8: (9.3.5.8): 56 data bytes
64 bytes from 9.3.5.8: icmp_seq=0 ttl=255 time=0 ms
64 bytes from 9.3.5.8: icmp_seq=1 ttl=255 time=0 ms
^C
----9.3.5.8 PING Statistics----
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0/0/0 ms

root@wpr03#ping 9.9.9.9
```

```
PING 9.9.9.9: (9.9.9.9): 56 data bytes
0821-069 ping: sendto: Cannot reach the destination network.
ping: wrote 9.9.9.9 64 chars, ret=-1
0821-069 ping: sendto: Cannot reach the destination network.
ping: wrote 9.9.9.9 64 chars, ret=-1
^C
----9.9.9.9 PING Statistics----
2 packets transmitted, 0 packets received, 100% packet loss
```

When attempting to ping `9.3.5.186` from `wpr03`, the routing look-up algorithm finds that the longest match (32 bits) corresponds to the loopback interface. However, this route is marked as belonging to a different WPAR and is therefore skipped.

The next longest match is `9.3.4/23` (23 bits match). There are three instances of this match in the routing table: `wpr04 9.3.4/23`, `Global  9.3.4/23`, and `wpr03 9.3.4/23`. The first instance belongs to `wpr04` and is therefore skipped. The second instance belongs to `Global` and is also skipped. However, the last instance belongs to `wpr03`, so interface `en0` is selected.

When a WPAR intends to communicate with the global environment, the same logic applies. When attempting to ping an external system such as `9.3.5.186` from `wpr03`, the routing look-up algorithm looks up the longest match of the destination address that has been marked as *belonging* to the WPAR.

First it tries to find a host route. If a host route is not available, it tries to find a subnet/network route. If a subnet/network route is not available, it tries to find a default route. However, if there is no route, the enhanced lookup algorithm fails.

**Important:** Ensure that you always define a default route for a WPAR so that the enhanced route lookup algorithm will never fail.

Routes associated to a WPAR address are automatically added to the routing table when the WPAR is started.

When the WPAR is stopped, the IP address of the WPAR interface is deleted and all routes associated with that address will also be deleted. Thus, adding and removing all routes associated with a WPAR interface is performed automatically.

The global environment can access all routes either associated with the global system or with a WPAR. This implies that the network traffic originating from the global environment to any WPAR is routed via the loopback interface.

## 6.6.3 Managing WPAR-specific routing

This section explains how to add and delete routes to and from the routing table, and how to enable and disable WPAR-specific routing.

### Adding and deleting routes to and from the routing table

Example 6-18 on page 187 demonstrates the importance of having a default route in a WPAR so that the enhanced route lookup algorithm would never fail. WPAR management commands and SMIT interfaces have been enhanced to allow route addition/deletion to and from the WPAR routing table.

A new option -I has been added to support adding and deleting routes. The attributes supported by the -I option are:

| | |
|---|---|
| rttype=<net/host> | [whether this is a host route or a net route] |
| rtdest=<A.B.C.D> | [destination address] |
| rtgateway=<A.B.C.D> | [gateway address] |
| rtnetmask=<A.B.C.D> | [network mask] |
| rtinterface=<if> | [outgoing interface] |

The following command adds a default route (`0.0.0.0`) accessible via `1.1.1.10` while configuring `wpr03`.

```
# mkwpar -n wpr03 -N address=1.1.1.1 interface=en0
netmask=255.255.255.0 –I rtdest=0.0.0.0 rtgateway=1.1.1.10
```

The following command adds a subnet route for the network `1.1.0.0/16` accessible via `1.1.1.10` associated to the existing `wpr03`

```
# chwpar –I rttype=net rtdest=1.1.0.0 rtgateway=1.1.1.10
rtnetmask=255.255.0.0 rtinterface=en0 wpr03
```

The following command deletes from routing table the subnet route for the network `1.1.0.0/16` associated with `wpr03`

```
# chwpar –K –I rtdest=1.1.0.0 rtgateway=1.1.1.10 wpr03
```

As explained, regarding the enhanced route lookup algorithm, any change to a WPAR routing table does *not* affect the routing configurations of the global environment or other WPARs.

### Enabling and disabling WPAR-specific routing

WPAR management commands and SMIT interfaces have been enhanced to allow enabling and disabling WPAR-specific routing. A new flag **`-i`** has been added to **`mkwpar`**, **`wparexec`** and **`chwpar`** commands. The flag is Boolean and allows you to switch on and switch off WPAR-specific routing. Its default value is off, which means the WPAR uses the global routing table. This flag is equivalent to setting the `general.routing` attribute to `yes` or `no` in the specification/configuration file.

After enabling WPAR-specific routing and adding a subnet route using the **`chwpar`** command as shown, the WPAR settings look similar to Example 6-19.

*Example 6-19 Displaying WPAR-specific routing information*

```
root@lpar13#lswpar -L wpr03|grep -ip rout
=================================================================
wpr03 - Active
=================================================================
GENERAL
Type:                   S
Owner:                  root
Hostname:               wpr03
WPAR-Specific Routing:  yes
Directory:              /wpars/wpr03
Start/Stop Script:
Auto:                   no
Private /usr:           no
Checkpointable:         no
Application:

USER-SPECIFIED ROUTES
Type   Destination       Gateway       Interface
-----------------------------------------------------------------
net    1.1.0.0/16        1.1.1.10      en0
```

For more details about the use of these new flags with WPAR-related commands, refer to the corresponding manual pages.

## 6.7 Auditing and accounting in WPAR environments

The following sections discuss auditing and accounting in WPAR environments in detail.

## 6.7.1 Overview of auditing in WPAR environments

This section provides an overview of auditing subsystem in WPAR environments, along with practical examples of WPAR auditing and global-initiated WPAR auditing.

AIX 6 supports the auditing subsystem, which allows administrators to record various information regarding the system. This section refers to an *auditable* event as an event that occurs on the system and is relevant to system security.

The practice of system auditing involves detecting an auditable event, collecting information regarding that event into an audit event record, processing that information, creating audit reports, and generating security alerts.

There are two modes in which audit data can be collected: BIN mode and STREAM mode:

- When BIN mode is used, the audit information is alternatively logged against two temporary BIN files and then appended to a single audit trail file.
- When STREAM mode is used, the audit records are written into a circular buffer that can be read using a /dev/audit device file.

The two modes can be enabled and used simultaneously.

There are also two types of auditing: event auditing and object auditing, as explained next.

### Event auditing

Event auditing monitors certain *events* associated with user processes. There is a predefined set of approximately 300 different base events that are built into AIX. The complete list of these events can be found in the /etc/security/audit/events file.

An audit class can be defined using any subset of the base events set. One or more audit classes can be associated with an user ID. The associations are kept in the /etc/security/audit/config file. Each process is monitored against the audit class associated to its owner.

### Object auditing

The subject of object auditing is access to individual files. All three types of accessing a file (that is, read, write, and execute) can be audited. The files being audited must be defined in the /etc/security/audit/objects file.

## 6.7.2 Using auditing in WPAR environments

There are three ways in which auditing can be used in WPAR environments:

- Global system auditing

  Events are collected based on the global configuration files and recorded on the global environment.

- WPAR auditing

  Events are collected and recorded inside the WPAR, based on the local WPAR configuration files.

- Global-initiated WPAR auditing

  WPAR events are collected and recorded on the global environment based on settings kept in the global configuration files.

Enabling auditing in the global environment or within a system WPAR is very similar to using auditing on a traditional stand-alone AIX system.

**Note:** Auditing is disabled within an application WPAR. However, an application WPAR can be audited from the global environment.

For instance, if the WPAR administrator decides to use event auditing and collect `general` class associated to `root` and `files` class associated to `tmpuser`, then the WPAR configuration file would look similar to Example 6-20.

*Example 6-20 Sample event audit configuration file*

```
start:
        binmode = on
        streammode = off

bin:
        trail = /audit/trail
        bin1 = /audit/bin1
        bin2 = /audit/bin2
        binsize = 10240
        cmds = /etc/security/audit/bincmds
        freespace = 65536

stream:
        cmds = /etc/security/audit/streamcmds

classes:
```

```
general=USER_SU,PASSWORD_Change,FILE_Unlink,FILE_Link,FILE_Rename,FS_Ch
dir,FS_Mkdir,FS_Rmdir

files=FILE_Open,FILE_Read,FILE_Write,FILE_Close,FILE_Link,FILE_Unlink,FI
LE_Rename

users:
        root = general
        tmpuser = files
```

If the WPAR administrator decides to use object auditing to audit all read and write operations of the WPAR /etc/security/passwd file, the stanza shown in Example 6-21 should be present in the WPAR /etc/security/audit/objects file.

*Example 6-21 Auditing read and write access to /etc/security/passwd file*

```
/etc/security/passwd:
        r = "S_PASSWD_READ"
        w = "S_PASSWD_WRITE"
```

Example 6-22 illustrates an excerpt of the WPAR audit trail that includes events occurred on a WPAR when local root changed the password for `tmpuser`.

*Example 6-22 Excerpt from a WPAR audit trail file*

```
root@wpr03#auditpr < /audit/trail
event           login    status      time                     command
wpar name
--------------- -------- ----------- ------------------------
------------------------------- -------------------------
S_PASSWD_READ   root     OK          Fri Jun 27 16:19:28 2008 srcmstr
wpr03
S_PASSWD_READ   root     OK          Fri Jun 27 16:19:28 2008 srcmstr
wpr03
FS_Chdir        root     OK          Fri Jun 27 16:19:28 2008 srcmstr
wpr03
FILE_Unlink     root     FAIL        Fri Jun 27 16:19:28 2008 srcmstr
wpr03
```

```
FILE_Rename    root    OK         Fri Jun 27 16:19:28 2008 srcmstr
wpr03
FILE_Unlink    root    OK         Fri Jun 27 16:19:28 2008 srcmstr
wpr03
FILE_Unlink    root    OK         Fri Jun 27 16:19:28 2008 compress
wpr03
FILE_Unlink    root    OK         Fri Jun 27 16:19:33 2008 uncompress
wpr03
FILE_Unlink    root    OK         Fri Jun 27 16:19:33 2008 uncompress
wpr03
FILE_Unlink    root    OK         Fri Jun 27 16:19:33 2008 uncompress
wpr03
PASSWORD_Change root    OK         Fri Jun 27 16:20:35 2008 passwd
wpr03
S_PASSWD_READ   root    OK         Fri Jun 27 16:20:40 2008 telnetd
wpr03
S_PASSWD_READ   root    OK         Fri Jun 27 16:20:40 2008 telnetd
wpr03
PASSWORD_Change root    OK         Fri Jun 27 16:20:52 2008 tsm
wpr03
FILE_Open      tmpuser  OK         Fri Jun 27 16:20:52 2008 tsm
wpr03
FILE_Read      tmpuser  OK         Fri Jun 27 16:20:52 2008 tsm
wpr03
FILE_Close     tmpuser  OK         Fri Jun 27 16:20:52 2008 tsm
wpr03
FILE_Open      tmpuser  FAIL       Fri Jun 27 16:20:52 2008 tsm
wpr03
```

### 6.7.3 Global-initiated WPAR auditing

When global-initiated WPAR auditing is used, WPAR events are collected on the global system based on configurations kept on the global system. The global audit subsystem should be started *before* starting the WPAR audit subsystem.

When using event auditing, you must ensure the following:

- All classes that are to be audited on the WPARs have been added to the global configuration file /etc/security/audit/config
- All WPARs are added to the /etc/security/audit/config in a stanza named WPARS using the following format:

```
WPARS:
```

```
WPAR_ name = <class name1>, <class name2> …
```

The global system will collect the selected events occurring on the selected WPARs. Note that all data will be logged against the *global* system audit trail.

For instance, if the global administrator decides to use event auditing and collect the `general` class events for `wpr04`, the configuration file would look similar to Example 6-23;

*Example 6-23 Example of event audit configuration file for WPARs*

```
start:
        binmode = on
        streammode = off

bin:
        trail = /audit/trail
        bin1 = /audit/bin1
        bin2 = /audit/bin2
        binsize = 10240
        cmds = /etc/security/audit/bincmds
        freespace = 65536

stream:
        cmds = /etc/security/audit/streamcmds

classes:
general =
USER_SU,PASSWORD_Change,FILE_Unlink,FILE_Link,FILE_Rename,FS_Chdir

users:
root = general

WPARS:
wpr04 = general
```

When using object auditing, all objects to be audited should be added to the global configuration file /etc/security/audit/objects.

If the global system administrator decides to use object auditing to audit the file /etc/security/passwd on `wpr04`, the following stanza be appended to the global /etc/security/audit/objects file.

```
/wpars/wpr04/etc/security/passwd:
```

```
r = "S_WPAR_PASSWD"

w = "S_WPAR_PASSWD_WRITE"
```

**Notes:**

- Ensure that you use the absolute paths of the global environment.
- Ensure that stanzas referring to WPARs are the last stanzas present in the objects file.

Example 6-24 illustrates the audit events produced by a scenario that includes the following steps:

1. Auditing is started in the global environment.
2. Auditing is started on `wpr04`.
3. A **`clogin`** command to `wpr04` is followed by a command such as **`ls`**.
4. Auditing is stopped.
5. The audit log is displayed. Notice that the set of events, generated and recorded during **`clogin`** containing `S_WPAR_PASSWD`**,** `USER_SU` and `FS_Chdir`, is also a subset of the class `general`.

*Example 6-24   Example of WPAR-level audit events*

```
root@lpar13#audit start

root@lpar13#audit start -@ wpr04

root@lpar13#clogin wpr04 "ls"
.sh_history
admin
audit
bin
dev
etc
home
lib
lost+found
lpp
mnt
opt
proc
sbin
tftpboot
```

```
tmp
u
unix
usr
var
wpars
root@lpar13#audit shut
auditing reset

root@lpar13#auditpr </audit/trail
event          login    status     time                    command
wpar name
--------------- -------- ----------- ------------------------
------------------------------ -------------------------
S_WPAR_PASSWD   root     OK        Fri Jun 27 18:24:25 2008 su
wpr04
S_WPAR_PASSWD   root     OK        Fri Jun 27 18:24:25 2008 su
wpr04
USER_SU        root     OK        Fri Jun 27 18:24:25 2008 su
wpr04
FS_Chdir       root     OK        Fri Jun 27 18:24:25 2008 su
wpr04
```

## 6.7.4 Overview of accounting in WPAR environments

This section provides an overview of the accounting subsystem in WPAR environments, along with examples of WPAR accounting and global-initiated WPAR accounting.

Advanced Accounting in AIX 6 provides utilization information for a wide variety of system resources including disks, network interfaces, file systems, processors, and memory.

Workload partitions support Advanced Accounting for resources available within WPARs, along with accounting generic features such as interval accounting and data aggregation.

### Interval accounting

Interval accounting provides the ability to record accounting data periodically. Intervals can be defined separately for processes and system resources.

- *Process interval* refers to recording accounting data for all active processes including intermediate records of long-running processes.

- *System interval* refers to collecting accounting information related to system resources such as processors, memory, network interfaces, file systems and virtual devices. This can be used for load and performance monitoring.

#### Data aggregation

Data aggregation allows you to aggregate similar accounting records, thereby avoiding flooding the accounting repository with numerous individual accounting records.

Interval accounting should be enabled *prior to* enabling data aggregation, because aggregation accounting records are recorded periodically according to the intervals described in interval accounting.

## 6.7.5 Using accounting in WPAR environments

There are three ways you can use accounting in WPAR environments, as explained here:

- Global environment accounting

  Accounting data is collected based on global configuration settings and recorded in the global environment.
- WPAR accounting

  The WPAR administrator can configure the Advanced Accounting subsystem within a system WPAR similar to the global environment. A WPAR has its own configuration, which can be modified only by the WPAR administrator. A WPAR has its own accounting data. This accounting data contains transaction records specific to that WPAR.
- Global-initiated WPAR accounting

  The global environment administrator can enable Advanced Accounting for any active system WPAR. Accounting data will be collected into the global repository.

  Accounting in the global environment or within a system WPAR is very similar to using accounting on a traditional stand-alone AIX system.

**Note:** Advanced Accounting cannot be enabled within an application WPAR. Application WPAR accounting information can be obtained only from the global environment.

To enable WPAR accounting, a new set of WPAR-specific transactions has been defined for differentiating WPAR accounting records from global environment accounting records. Each WPAR-specific transaction record has an additional field named *wparname* which uniquely identifies specific WPAR records.

Table 6-1 lists and describes the new WPAR-specific transactions.

*Table 6-1 WPAR-specific transactions*

| Transaction ID | Description |
|---|---|
| 33 | WPAR Process record |
| 34 | WPAR Aggregated Process record |
| 35 | WPAR Aggregated Application record |
| 36 | WPAR processor and memory usage record |
| 38 | WPAR file system activity record |
| 39 | WPAR network interface I/O record |

Transactions 33, 34, and 35 are assigned to process records for application WPARs. These records can be generated only when process interval accounting is enabled. Process transactions for system WPARs cannot be completely recorded in the global environment because the set of users of a system WPAR can be different from the set of users of the global environment.

Transaction 36, 38, and 39 are defined relative to the CPUMEM, FILESYS, and NETIF LPAR accounting records of system. Accounting records can be generated for any WPAR that has system interval accounting enabled, provided that global-initiated WPAR accounting is also enabled.

Advanced Accounting requires a kernel extension being loaded in the shared kernel. By default, loading kernel extensions is allowed only in the global environment. Therefore, accounting can be enabled within a system WPAR only *after* loading the Advanced Accounting kernel extension in the global environment using, for instance, the **`acctctl on`** command.

If you decide to load the kernel extension directly from WPAR, then the WPAR should be granted the PV_DEV_LOAD privilege.

There are transactions that cannot be recorded within a WPAR. For instance, disk transactions (transaction id 8), vscsi_target transactions (transaction id 10), or vscsi_client transactions (transaction id 11) are not available within a WPAR.

When using interval accounting, WPAR administrators can set the interval of their choice only if there is no interval specified in the global environment. If the interval value in the global environment Advanced Accounting is already specified, then it cannot be overwritten at the WPAR level and the WPAR Advanced Accounting subsystem will use the same value.

## 6.7.6 Global-initiated WPAR accounting

When global-initiated WPAR accounting is used, WPAR records are collected on the global system based on configurations kept on the global system.

Example 6-25 illustrates the accounting records produced by a scenario that includes the following steps:

1. Two accounting data files, each 10 MB in size, are defined.
2. Global accounting is started.
3. Local accounting is started on `wpr03`.
4. System accounting is configured using a 1-minute interval.
5. Process accounting is configured using a 1-minute interval.
6. System level aggregation is enabled.
7. Global settings are verified.
8. Propagation of the global settings to the WPAR is verified.
9. The name of the active data file is displayed along with its utilization and time stamps of the first and last records.
10. LPAR accounting report for `wpr03` is displayed.

*Example 6-25 Example of WPAR-level audit events*

```
root@lpar13#acctctl fadd /var/aacct/aacct1.dat 10

root@lpar13#acctctl fadd /var/aacct/aacct2.dat 10

root@lpar13#acctctl on

root@lpar13#acctctl on -@ wpr03

root@lpar13#acctctl isystem 1

root@lpar13#acctctl iprocess 1

root@lpar13#acctctl agproc on

root@lpar13#acctctl
Advanced Accounting is running.
```

```
Email notification is off.
The current email address to be used is not set.
Process Interval Accounting every 1 minutes.
System Interval Accounting every 1 minutes.
System-wide aggregation of process data is on.
System-wide aggregation of third party kernel extension data is off.
System-wide aggregation of ARM transactions is off.
Recover CPU accounting time in turbo mode is False.
Files: 2 defined, 1 available.

root@lpar13#acctctl -@ wpr03
Advanced Accounting is running.
Process Interval Accounting every 1 minutes.
System Interval Accounting every 1 minutes.
System-wide aggregation of process data is off.
System-wide aggregation of third party kernel extension data is off.


root@lpar13#acctctl fquery
FILENAME
  STATE     | FIRST RECORD TIME            | LAST RECORD TIME             | UTIL
/var/aacct/aacct1.dat
  Active    | Sat Jun 28 17:05:36 CDT 2008 | Sat Jun 28 18:07:41 CDT 2008 |   5%

/var/aacct/aacct2.dat
  Available |                              |                              |   0%

root@lpar13#acctrpt -f /var/aacct/aacct1.dat -L ALL -@ wpr03

CPU and Memory Accounting Report
-----------------------------------------

                         (C) IDLE     IOWAIT   SPROC    UPROC    INTR    (sec)
 WPAR          CNT       (U) IO       PGSPIN   PGSPOUT  LGPGUTIL PGRATE
-----          ---       --- ----     ------   -------  -------- ------
wpr03           60        C:  0.0      0.0      36.2     54.9     0.0
                          U:  582015   0        0        0.0      0.0

File Systems Accounting Report
------------------------------

 WPAR           CNT    DEVNAME            MOUNTPT    FSTYPE RDWR    OPEN    CREATE
LOCKS   XFERS(MBs)
 ----           ---    -------            -------    ------ ----    ----    ------
-----   ----------
```

```
wpr03         60    /dev/fslv01       /wpars/wpr03/home 0     436764  16     16
0       1706.1
wpr03         60    /dev/fslv02       /wpars/wpr03/tmp 0      0       0      0
0       0.0
wpr03         60    pipefs            pipefs     16     773     0      0      0
0.1
wpr03         60    /dev/fslv03       /wpars/wpr03/var 0      348     112    52
28      0.1
wpr03         60    /dev/hd2          /usr       0      3399    1327    0      0
11.8
wpr03         60    /dev/fslv00       /wpars/wpr03 0      728769  2413    48
732     2842.1
wpr03         60    specfs            specfs     16     7043    368     0      0
0.4

Network Interfaces Accounting Report
------------------------------------

 WPAR          CNT        NETIFNAME       NUMIO      XFERS(MBs)
 ----          ---        ---------       -----      ----------
wpr03          60         lo0             15208      9.312749
wpr03          60         en0             18184      0.896768
```

7

# Advanced configuration features

This chapter describes and provides examples of more advanced configuration features and options in and for WPARs and their global environments. The following topics are discussed:

- WPAR administrative scalability
- Specification files
- NIM client support
- Advanced file system considerations
- Backup considerations
- Software maintenance
- Print spooling
- System environment notes
- Processes and subsystems
- WPAR resource limiting
- IPv6
- Network Name Mapped Interface support
- WPAR static settings resolution

## 7.1 WPAR administrative scalability

Administrative scalability can best be described as removing and or quickly adapting to conflicting demands on resource usage, management, or security policies in multiple, independent WPARs running in the same global environment.

Administrative scalability is a highly desirable feature today as people look to make their infrastructure as flexible as possible, to allow organizations to quickly cater for changing demands in their IT environments.

WPARs provide for this in two key ways:

- The ability to scale both laterally (in the size of the WPAR and the application or applications that exist within the system WPAR) and to scale up to 8192 WPARs running within a single global environment
- The WPAR mobility feature makes these partitions now portable from one AIX host to another without disruption to the applications running within the WPARs

With the creation of many WPARs within a single global environment, and as more demand is placed on resources as more WPARs are created, the effects of this growth can be minimized by the use of *specification files* when you create WPARs.

## 7.2 Specification files

WPAR specification files are an optional aid to creating, managing, and cloning WPARs based on an existing configuration or a collection of predetermined configuration settings.

WPARs provide for many options and choices when it comes to configuration settings for defining workload partitions. When you create a WPAR with the default settings, the **mkwpar** command or **smit** requires little input from the system administrator. However, the use of the **mkwpar** command or **smit** can be cumbersome when the system administrator needs to change many of the default settings or wants to create a large number of WPARs simultaneously.

The use of specification files can expedite these activities and remove a great deal of complexity, particularly if a working model of your WPAR settings has already been captured. A specification file can be used to create a WPAR by storing configuration settings in a predefined format that contains all the information required by the **mkwpar** command or **smit** to install numerous WPARs quickly and easily.

A specification file is a text file that can be edited using any text editor. The sample specification file that is provided in /usr/samples/wpars/sample.spec is shown in Example 7-1. This sample file contains examples of configuration stanzas, as well as their detailed descriptions.

*Example 7-1 Example specification file*

```
general:
        name = "gordon"
        checkpointable = "no"
        hostname = "gordon"
        directory = "/wpars/gordon"
        application = /usr/sbin/apachectl start privateusr = "no"
        privateusr = "no"
        devices = "/etc/wpars/devexports"
        auto = "yes"
        preserve = "no"
        routing = "yes"


network:
        broadcast = "9.3.5.255"
        interface = "gordon_if1"
        address = "9.3.5.188"
        netmask = "255.255.254.0"

resources:
        active = "yes"

device:
        devname = "/dev/null"
        devtype = "1"

device:
        devname = "/dev/tty"
        devtype = "1"

device:
        devname = "/dev/console"
        devtype = "1"

device:
        devname = "/dev/zero"
        devtype = "1"

device:
        devname = "/dev/clone"
        devtype = "1"

device:
        devname = "/dev/sad"
        devtype = "3"
mount:
        logname = "INLINE"
        directory = "/wsapp4"
        vfs = "nfs"
        vg = "rootvg"
        size = "2097152"

mount:
        logname = "INLINE"
        directory = "/var"
        vfs = "nfs"
        vg = "rootvg"
        size = "262144"

mount:
        logname = "INLINE"
        directory = "/tmp"
        vfs = "nfs"
        vg = "rootvg"
        size = "262144"

mount:
        logname = "INLINE"
        directory = "/home"
        vfs = "nfs"
        vg = "rootvg"
        size = "262144"

mount:
        dev = "/opt"
        directory = "/opt"
        vfs = "namefs"
        mountopts = "ro"

mount:
        logname = "INLINE"
        directory = "/"
        vfs = "nfs"
        vg = "rootvg"
        size = "262144"

mount:
        dev = "/usr"
directory = "/usr"
        vfs = "namefs"
        mountopts = "ro"

mount:
        logname = "INLINE"
        directory = "/wsapp5"
        vfs = "nfs"
        vg = "rootvg"
        size = "2097152"

security:
        privs =
"PV_AU_,PV_AU_ADD,PV_AU_ADMIN,PV_AU_PROC,PV_AU_READ,PV_AU_WRITE,PV_AZ_ADMIN,PV_AZ_CHE
CK,PV_AZ_READ,PV_AZ_ROOT
,PV_DAC_,PV_DAC_GID,PV_DAC_O,PV_DAC_R,PV_DAC_RID,PV_DAC_UID,PV_DAC_W,PV_DAC_X,PV_DEV_
CONFIG,PV_DEV_QUERY,PV_FS_CHOWN,PV_FS_CH
ROOT,PV_FS_CNTL,PV_FS_LINKDIR,PV_FS_MKNOD,PV_FS_MOUNT,PV_FS_PDMODE,PV_FS_QUOTA,PV_KER
_ACCT,PV_KER_CONF,PV_KER_DR,PV_KER_EWLM,
PV_KER_EXTCONF,PV_KER_IPC,PV_KER_IPC_O,PV_KER_IPC_R,PV_KER_IPC_W,PV_KER_NFS,PV_KER_RA
C,PV_KER_RAS_ERR,PV_KER_REBOOT,PV_NET_PO
RT,PV_PROC_CKPT,PV_PROC_CORE,PV_PROC_CRED,PV_PROC_ENV,PV_PROC_PRIO,PV_PROC_PDMODE,PV_
PROC_RAC,PV_PROC_RTCLK,PV_PROC_SIG,PV_PR
OC_TIMER,PV_PROC_VARS,PV_PROC_PRIV,PV_SU_UID,PV_TCB,PV_TP,PV_TP_SET,PV_MIC,PV_MIC_CL,
PV_LAB_,PV_LAB_CL,PV_LAB_CLTL,PV_LAB_LEF
,PV_LAB_SLDG,PV_LAB_SLDG_STR,PV_LAB_SL_FILE,PV_LAB_SL_PROC,PV_LAB_SL_SELF,PV_LAB_SLUG
,PV_LAB_SLUG_STR,PV_LAB_TL,PV_MAC_,PV_MA
C_CL,PV_MAC_R,PV_MAC_R_CL,PV_MAC_R_STR,PV_MAC_R_PROC,PV_MAC_W,PV_MAC_W_CL,PV_MAC_W_DN
,PV_MAC_W_UP,PV_MAC_W_PROC,PV_MAC_OVRRD,
PV_KER_SECCONFIG,PV_PROBEVUE_TRC_USER,PV_PROBEVUE_TRC_USER_SELF"
```

When a WPAR has already been defined, the system administrator can create a specification file from the existing WPAR using **`mkwpar -e <wparname> -w -o filename`**. It is then possible to make clones of this existing WPAR by renaming and editing this specification file. As shown in Example 7-2 on page 208, we created a WPAR named `harold` using a specification file generated from the `gordon` WPAR, after we edited some settings.

*Example 7-2 Making clone WPARs*

```
Create a Workload Partition from a Specification File

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

[TOP]                                                    [Entry Fields]
* Workload Partition Name                             [harold]
* Specification File Location
[/usr/sys/inst.images/gordon.spec]                                      /
  HOSTNAME                                            [harold]
  Base Directory                                      [/wpars/harold]
 Default Volume group name                [datavg]
+
 Copy Global Name Resolution Settings         no
+
  User Script                                         []
 Network INTERFACE                       harold_if1
+
      Internet ADDRESS (dotted decimal)               [9.3.5.198]
      Network MASK                                    [255.255.254.0]
    -OR-
      IPv6 ADDRESS (colon delimited)                  []
      Prefix Length                                   []
 WPAR-Specific Routing?                   yes
+
 Create the Workload Partition?             yes
+
 Specification Output File                []
/
 Checkpointable?                         no
+
 START Workload Partition?                 no
+
 Start at system boot?                   yes
+
 Automatically RESOLVE conflicts?            no
+

  Resource Control
    Active                              yes
+
      Resource Set                                    []
    CPU Shares                          []
#
      CPU Minimum (%)                                []
      CPU Maximum (%)                                []
      CPU Absolute Maximum (%)                       []
    Memory Shares                       []
#
      Memory Minimum (%)                             []
      Memory Maximum (%)                             []
      Memory Absolute Maximum (%)                    []
      Per-Process Virtual Memory Limit               []
```

After you have the specification file, you can make any edits that you require before you create your remaining system WPARs, parsing the specification file as an argument to the **`mkwpar`** command. (Note that this method applies to system WPARs only.)

The WPAR can then be created by passing the specification filename as an argument to the **`-f`** flag of the **`mkwpar`** or **`wparexec`** commands.

There are two other files that are also involved in creating a WPAR:

- The /etc/wpars/secattrs file describes the default privileges for the WPARs that are being created, and
- The /etc/wpars/devexports file describes the default device handling for the WPARs that are being created.

The user can specify alternate paths to files of the same format via command line options.

**Note:** Both the secattrs and devexports configuration files are only applicable to system WPARs.

Because specification files can be stored anywhere within your AIX file system directory structure and the /etc/wpars directory organization is subject to change, they should not be directly used by the system administrator. We recommend that you create a specific WPAR administration file system where all specification and critical files are centrally located. This will aid the system administrator as well as provide simplicity around the backup regime for these critical files.

## 7.3 NIM client support

With the release of AIX V6.1 TL2, Network Installation Manager (NIM) client support has been added to the WPAR offering. This has resulted in a number of added minor changes to the NIM environment to cater for the requirements and behavior of the WPARs. This provides system administrators with the capability of managing installations of AIX and optional software to a number of system WPARs from the same NIM masters that serve the LPARs and global environments within the overall IT environment.

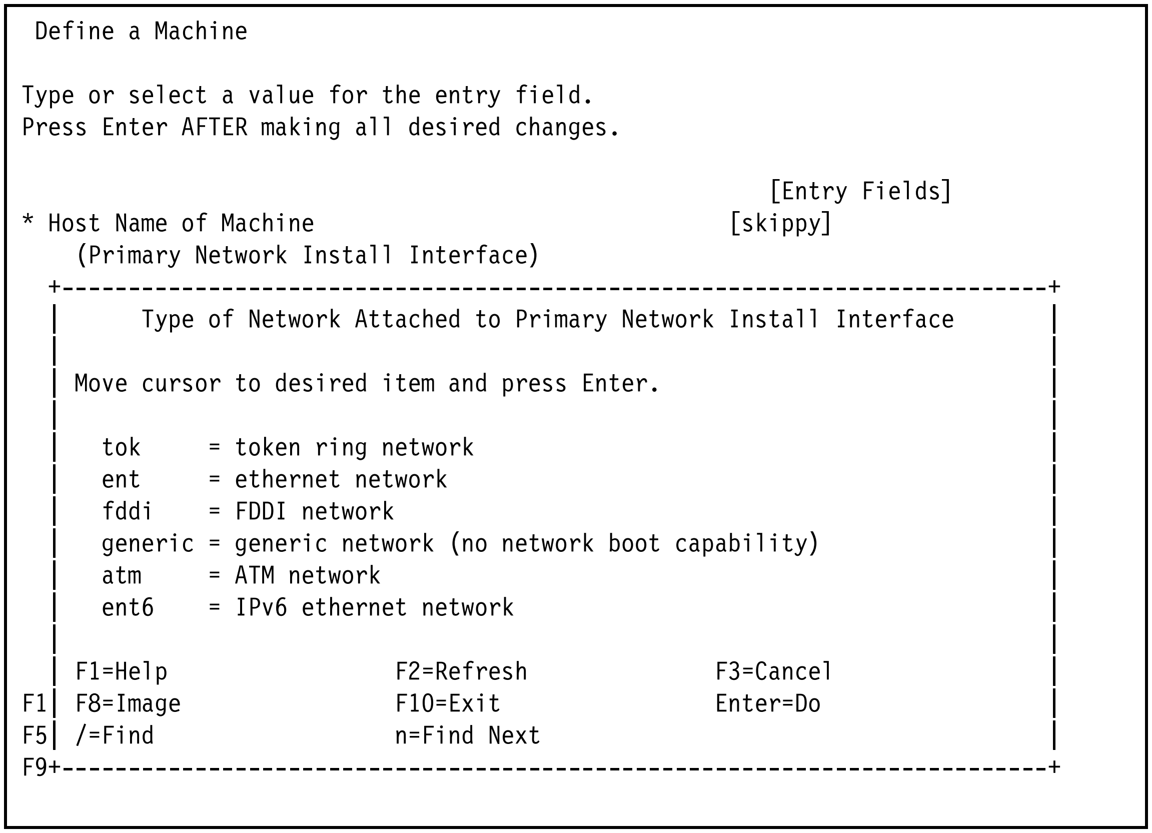
Because an application WPAR is a transient object, and therefore not suited to integration with NIM, application WPARs are not supported.

**Important:** The inclusion of NIM client support for WPARs is a new feature in AIX V6.1 TL2. Therefore, both the NIM Server and its clients in the global environment and the hosted WPARs need to be at least at this level in order to exploit this feature. Earlier versions of AIX, including AIX 6.1 TL1, do not support this feature.

The **`nimclient`** command allows NIM operations to be performed from the NIM client. Under normal circumstances within an LPAR, this would allow for the machine defined to NIM to initiate functions and operations using the **`nimclient`** command with the **`-o`** flag. However, for a WPAR, these functions and operations are further limited to those operations that do not involve the changing or any manipulation of the global environment or its relationship with the NIM master. With these exceptions, the **`nimclient`** command in a WPAR will function just the same as it would function if executed on any other client within the NIM environment.

### 7.3.1 Machine definition

An additional machine type, *wpar*-Workload Partition and a Machine Subclass, *wpar_res*-, a subset of resource types for use with WPAR, are shown in the new panels. Also in the new panels is the ability to utilize this feature in an IPv6 network; see Figure 7-1 on page 211.


```
 Define a Machine

Type or select a value for the entry field.
Press Enter AFTER making all desired changes.

                                                           [Entry Fields]
* Host Name of Machine                                  [skippy]
    (Primary Network Install Interface)
  +--------------------------------------------------------------------------+
  |      Type of Network Attached to Primary Network Install Interface       |
  |                                                                          |
  | Move cursor to desired item and press Enter.                             |
  |                                                                          |
  |   tok     = token ring network                                           |
  |   ent     = ethernet network                                             |
  |   fddi    = FDDI network                                                 |
  |   generic = generic network (no network boot capability)                 |
  |   atm     = ATM network                                                  |
  |   ent6    = IPv6 ethernet network                                        |
  |                                                                          |
  | F1=Help                 F2=Refresh              F3=Cancel                |
F1| F8=Image                F10=Exit                Enter=Do                 |
F5| /=Find                  n=Find Next                                      |
F9+--------------------------------------------------------------------------+
```

*Figure 7-1 Network options for the client machine definition*

Following on, Figure 7-2 on page 212 shows, in the next options screen, the addition of the WPAR machine type.

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

[TOP]                                                    [Entry Fields]
* NIM Machine Name                                    [skippy]
* Machine Type                                        [standalone]        +
* Hardware Platform Type                              [chrp]              +
  +---------------------------------------------------------------------------+
  |                               Machine Type                                |
  |                                                                           |
* | Move cursor to desired item and press Enter. Use arrow keys to scroll.    |
  |                                                                           |
  |   diskless         = all filesystems & resources remote                   |
* |   dataless         = local paging,dump; remote /,/usr; others remote or   |
* |   standalone       = local filesystems & resources                        |
* |   alternate_master = alternate machine which can control the NIM environ  |
* |   wpar             = file systems and resources hosted on managing syste  |
[M|                                                                           |
  | F1=Help                F2=Refresh             F3=Cancel                   |
F1| F8=Image               F10=Exit               Enter=Do                    |
F5| /=Find                 n=Find Next                                        |
F9+---------------------------------------------------------------------------+
```

*Figure 7-2 The addition of the WPAR machine definition on the NIM master*

Other inclusions in this screen concerning WPAR management by the NIM master are for information about the managing system or global environment and the WPAR groups to which the WPAR defined belongs.

### 7.3.2 NIM operations applicable to WPARs

The following NIM operations are applicable to WPARs being managed in a NIM environment. Under the covers they call the standard WPAR commands, but explanations of the NIM listed operations for WPARs and WPAR behavior are given here:

| | |
|---|---|
| **activate** | Start a managed machine. |
| **chwpar** | Change an instance of a managed machine. |
| **create** | Create an instance of a managed machine. |
| **deactivate** | Stop a managed machine. |

| | |
|---|---|
| **`destroy`** | Remove an instance of a managed machine. |
| **`lswpar`** | List workload partition characteristics. |
| **`syncwpar`** | Synchronize workload partition software. |

### 7.3.3 WPAR NIM states

The WPAR, when viewed from the managed system, will be in one of three states:

| | |
|---|---|
| **`Defined`** | The WPAR has not been created and not running. |
| **`Created`** | The WPAR has been created and is in a defined state, i.e not running. |
| **`Running`** | Created, started and operational. |

### 7.3.4 Additional resource types

The following additional resources have been added to NIM to handle functions and operations that are unique to WPARs and what they require to carry out certain operations within the NIM environment:

| | |
|---|---|
| **`devexports`** | Device handling file used during WPAR install. |
| **`savewpar`** | Back up the WPAR image. |
| **`secattrs`** | Security privileges file for WPARs. |
| **`wpar_spec`** | General specification file used by WPARs during installation. |

## 7.4 Advanced file system considerations

A WPAR, when created, can be configured using one of several types of file systems: namefs, jfs, jfs2, or nfs. By default, the system creates /, /home, /tmp, and /var as jfs2 and /usr, /opt, and /proc as namefs. If you require additional file systems to support your applications, you will need to create them.

Inside the WPAR, you can mount a file system via nfs but you cannot create, change, or delete file systems. These tasks need to be done by someone with privileged access to, and from within, the global environment.

### 7.4.1 Creating additional file systems for WPARs

The fastest, most simplistic way to create a file system to an existing WPAR is by using the CLI. To make it permanent to a WPAR, specify the mount group with the name of that WPAR. Example 7-3 demonstrates the command to create a 1 Gb file system that will be mounted on /wsapp5 of the `gordon` WPAR, with an absolute mount point of /wpars/gordon/wsapp5 and using the **`-u`** flag we assign this file system to the gordon mount group.

*Example 7-3 Creating additional file systems for WPARs*

```
root@sydney:/# crfs -v jfs2 -g rootvg -m /wpars/gordon/wsapp5 -u gordon -a
logname=INLINE -a size=1G
File system created successfully.
1040148 kilobytes total disk space.
New File System size is 2097152
```

**Important:** The mount point for the file system in the global environment must be under the base or root directory for the WPAR (**`/wpars/gordon`** in Example 7-3) for the newly created file system to be visible from within the WPAR.

### 7.4.2 Configuring writable shared file systems

By default, a system WPAR shares the global environments /usr and /opt file systems. As stated earlier, they are read-only, but if you have an application that has a requirement to write to a local directory under /usr or /opt, then you can use the shared /usr and be able to write. In this case, create a new file system in the global environment and mount it within the WPAR's filesystem. Then, from the global environment, create a soft link from /usr to that file system.

### 7.4.3 Network File System (NFS) considerations

To share a directory or an entire file system, you must do so from the global environment. There should be no attempt to share directories that belong to WPARs, because the WPARs can be relocated, removed, stopped, or rebooted.

The NFS client is supported in the global environment and inside the WPARs. NFS V4 is supported. If you need to share a directory or file system, then create and share it from the global environment. This will ensure that it exists regardless of the state of any WPARs that need to mount it. If you are relocating your WPARs, then this approach will ensure that the file system and content is also available at the target machine.

## 7.5 Backup considerations

Application WPARs do not have their own file systems; the file systems are provided by the global environment.

System WPARs have their own unique file system structure and will require backups. The following information about this topic refers to system WPARs only.

### 7.5.1 The mkwpardata command

The **mkwpardata** command creates a file that contains information about a WPAR for use by the **savewpar** and restwpar **commands**. It generates detail of logical volumes, file systems and their sizes, volume group information and the details unique to the WPAR such as hostname. The **mkwpardata** command, when used with the **-m** flag, generates a list of the logical-to-physical partition mappings for each logical volume within the WPAR being backed up. This mapping can be used to determine the same logical mapping during a restore.

As with all the commands in the backup/restore regime, this command when executed with the **-X** flag will increase the space in the /tmp file system if it is required.

One of the files generated by the **mkwpardata** command is image.data. This file can be manually edited by the system administrator, if required. Note that if the image.data file exists when the command is executed, the existing file is overwritten.

### 7.5.2 The savewpar command

The **savewpar** command to the workload partition is similar to the **savevg** command. The **savewpar** command saves the files and configuration details of the WPAR. Generally the size of the backup generated by this command is small, because it is not required to save the shared file systems of the global environment within the WPAR image it generates.

The **savewpar** command must be executed from within the global environment and not the WPAR. It can be accessed via smit using the **smit savewpar** fast path. The command makes use of the **mkcd** and **mkdvd** utilities already within AIX to write the WPAR images. Example 7-4 on page 216 demonstrates the use of the **savewpar** command from the CLI to back up the `skippy` WPAR.

*Example 7-4 Using savewpar for backup*

```
root@sydney:/# savewpar -f /usr/sys/inst.images/skippy_swpr -imX skippy

Creating information file for workload partition skippy.
Creating list of files to back up.
Backing up 2186 files......
2186 of 2186 files (100%)
0512-038 savewpar: Backup Completed Successfully.
```

The **savewpar** command has a **-N** flag that allows you to back up the writable NFS file systems associated with the WPAR. This may be an important option to use for relocating WPARs that are using NFS file systems.

## 7.5.3 The restwpar command

The administrator of the global environment can restore a WPAR from a backup image created by the **savewpar** command. The **restwpar** command creates a workload partition from a backup image that was created using the **savewpar**, **mkcd**, or **mkdvd** commands. For more details about this topic, refer to 4.4.4, "Backing up and restoring a system WPAR" on page 80 and 4.4.5, "Cloning a system WPAR" on page 83.

A workload partition backup image contains an *image.data* file and a *workload partition specification* file, used to recreate the configuration and characteristics of the workload partition. There are flags that you can use to override these defaults, if required.

If the command is used and no input device is specified, it will by default read from /dev/rmt0.

If you specify a value of `Yes` in the EXACT_FIT field of the logical_volume_policy stanza of the /tmp/wpardata/WparName/image.data file, the **restwpar** command uses the map files to preserve the placement of the physical partitions at the time the backup was made for each logical volume.

**Note:** To list the files in a backup image or to restore individual files from the image, use the **lssavewpar**, **restwparfiles** or **restore** command with the **-T** or **-x** flag.

## 7.6 Software maintenance

System WPARs present a unique issue when you install or update system software. Because they have read-only access to /usr and /opt in the global environment, this prevents the normal installation of software inside the WPAR. Also, the WPAR's root user cannot delete software from shared directories. To install software on a system WPAR, you need to take a different approach.

There are two ways in which you can install software:

- Shared install - from the global environment using normal installation procedures
- Non-shared install - inside the WPAR utilizing read and write file systems

Application WPARs are not subject to the same issues around software installation because they use the file systems of the global environment. Therefore, the same software that is installed in the global environment is always readily accessible and available for the WPAR.

### 7.6.1 Software availability

The global environment is a regular AIX operating system instance. Accessing the CD-ROM or DVD-ROM, an NFS mount, or NIM is the same as usual. However, for WPARs, there are differences. To have software available inside the WPAR, you need to have it in a file system form as:

- A software repository
- A mounted CD or DVD
- Or an NFS mount

If the software you want to install is in a CD format, copy it to a regular file system. You can use **`smit bffcreate`** to do that. Simply mounting the CD might be enough for the operation that you want. In this case, pass the mount point as a namefs parameter to the WPAR at creation time: **`-M directory=/cdrom dev=/cdrom`**.

If your WPAR is already running, the best way to temporarily mount file systems is under /wpars/<wparname>/<mount_point> in the global environment, as shown in Example 7-5 on page 218.

An alternative approach is to create a subdirectory in the global environment, inside the WPAR system (like /tmp), and then copy the files to that location from the CD.

*Example 7-5 Mounting the media*

```
root@sydney:/# mount
  node       mounted        mounted over    vfs       date        options
-------- ---------------  ---------------  ------ ------------ ---------------
         /dev/hd4         /                jfs2   Jun 18 20:11 rw,log=/dev/hd8
         /dev/hd2         /usr             jfs2   Jun 18 20:11 rw,log=/dev/hd8
         /dev/hd9var      /var             jfs2   Jun 18 20:11 rw,log=/dev/hd8
         /dev/hd3         /tmp             jfs2   Jun 18 20:11 rw,log=/dev/hd8
         /dev/hd1         /home            jfs2   Jun 18 20:11 rw,log=/dev/hd8
         /dev/hd11admin   /admin           jfs2   Jun 18 20:11 rw,log=/dev/hd8
         /proc            /proc            procfs Jun 18 20:11 rw
         /dev/hd10opt     /opt             jfs2   Jun 18 20:11 rw,log=/dev/hd8
         /dev/livedump    /var/adm/ras/livedump jfs2   Jun 18 20:11 rw,log=/dev/hd8
         /dev/fslv00      /wpars/skippy    jfs2   Jun 19 17:01 rw,log=INLINE
         /dev/fslv01      /wpars/skippy/home jfs2   Jun 19 17:01 rw,log=INLINE
         /opt             /wpars/skippy/opt namefs Jun 19 17:01 ro
         /proc            /wpars/skippy/proc namefs Jun 19 17:01 rw
         /dev/fslv02      /wpars/skippy/tmp jfs2   Jun 19 17:01 rw,log=INLINE
         /usr             /wpars/skippy/usr namefs Jun 19 17:01 ro
         /dev/fslv03      /wpars/skippy/var jfs2   Jun 19 17:01 rw,log=INLINE
         /dev/fslv04      /wpars/kenny     jfs2   Jun 19 17:10 rw,log=INLINE
         /dev/fslv05      /wpars/kenny/home jfs2   Jun 19 17:10 rw,log=INLINE
         /opt             /wpars/kenny/opt namefs Jun 19 17:10 ro
         /proc            /wpars/kenny/proc namefs Jun 19 17:10 rw
         /dev/fslv06      /wpars/kenny/tmp jfs2   Jun 19 17:10 rw,log=INLINE
         /usr             /wpars/kenny/usr namefs Jun 19 17:10 ro
         /dev/fslv07      /wpars/kenny/var jfs2   Jun 19 17:10 rw,log=INLINE
         /dev/backuplv    /backups         jfs2   Jun 29 12:37 rw,log=INLINE
         /dev/imagslv     /usr/sys/inst.images jfs2   Jul 02 15:48 rw,log=INLINE
root@sydney:/# crfs -v jfs2 -g datavg -m /wpars/skippy/media -a size=5G -a
logname=INLINE
File system created successfully.
5222036 kilobytes total disk space.
New File System size is 10485760
root@sydney:/# mount /wpars/skippy/media
root@sydney:/# cp -R /mnt/cdrom /wpars/skippy/media
root@sydney:/# tn skippy
Trying...
Connected to skippy.
Last login: Sun Jul  6 14:08:48 EET 2008 on /dev/pts/0 from sydney
root@skippy:/# mount
  node       mounted        mounted over    vfs       date        options
-------- ---------------  ---------------  ------ ------------ ---------------
         /dev/fslv00      /                jfs2   Jun 19 17:01 rw,log=INLINE
```

```
        /dev/fslv01      /home            jfs2   Jun 19 17:01 rw,log=INLINE
        /opt             /opt             namefs Jun 19 17:01 ro
        /proc            /proc            namefs Jun 19 17:01 rw
        /dev/fslv02      /tmp             jfs2   Jun 19 17:01 rw,log=INLINE
        /usr             /usr             namefs Jun 19 17:01 ro
        /dev/fslv03      /var             jfs2   Jun 19 17:01 rw,log=INLINE
        /dev/fslv14      /media           jfs2   Jul 03 12:18 rw,log=INLINE
```

To use NFS for the media file system, you can use the **-M** flag when you create it. Or, if the WPAR is active, you can also simply mount it inside the WPAR as shown in Example 7-6.

**Note:** If you need to install the same software in many WPARs, a convenient way to accomplish this is to store all the installation files in a common NFS file system. Then mount it within the WPAR whenever you need it. This saves time, because you only need to manage one copy of the files.

*Example 7-6 NFS mounts inside an active WPAR*

```
root@skippy:/# mount newcastle:/usr/sys/inst.images /mnt/media
root@skippy:/# mount
  node       mounted        mounted over    vfs       date        options
-------- ---------------  ---------------  ------ ------------ ---------------
        /dev/fslv00      /                jfs2   Jun 19 17:01 rw,log=INLINE
        /dev/fslv01      /home            jfs2   Jun 19 17:01 rw,log=INLINE
        /opt             /opt             namefs Jun 19 17:01 ro
        /proc            /proc            namefs Jun 19 17:01 rw
        /dev/fslv02      /tmp             jfs2   Jun 19 17:01 rw,log=INLINE
        /usr             /usr             namefs Jun 19 17:01 ro
        /dev/fslv03      /var             jfs2   Jun 19 17:01 rw,log=INLINE
        /dev/fslv14      /media           jfs2   Jul 03 12:18 rw,log=INLINE
newcastle /usr/sys/inst.images /mnt/media       nfs3   Jul 03 12:36
```

Software installed in the global environment is not always instantly available to system WPARs for use. The administrator can use the **syncwpar** command or, from within the WPAR, run the **syncroot** command.

The **syncwpar** command is used to ensure that the important WPAR AIX files are synchronized with the global AIX files. If using private /usr and /opt, this is important to keep the private WPAR files updated to match the underlying AIX kernel.

The advantage in using **syncwpar** from the global environment is that you can make the software in question available to more than one WPAR using the one command and the WPAR names as arguments.

Alternatively, if you have a file that contains the working collection of WPAR hostnames, you can specify the command to synchronize the WPARs in that file, by hostname, with the newly installed software in the shared file system, generally /opt and /usr. For example, to make software recently installed or updated available to the system WPARs `skippy` and `gordon`, run one of the following commands:

- **syncwpar skippy gordon**
- **syncwpar -f** <filename>

## 7.6.2 Shared install

A shared install is one that will install software into the global environment that will allow several WPARs to use the same version of the software installed. This has the added advantage of saving space on the systems.

Start by installing the software in the global environment using the **installp** command or by using smit. Both methods have the ability to install software in detached or defined WPARs. After this is complete, synchronize the installation with the WPAR. This installs the root part of the newly installed software onto the WPAR and updates the ODM within the WPAR.

**Note:** Synchronize the WPAR after the shared software installation so that you prevent inconsistencies in the WPARs using the software products.

Example 7-7 illustrates the installation of the bos.net.mobip6.rte fileset, which provides for WPAR mobility in IPv6 into the global environment as a shared install for all WPARs to use.

*Example 7-7 Shared install in the global environment*

```
root@sydney:/# lslpp -L bos.net.mobip6.rte
  Fileset                      Level  State  Type  Description (Uninstaller)
  ----------------------------------------------------------------------------
lslpp: 0504-132  Fileset bos.net.mobip6.rte not installed.

root@sydney:# installp -ac -d /usr/sys/inst.images/ bos.net.mobip6.rte
+-----------------------------------------------------------------------------+
                    Pre-installation Verification...
+-----------------------------------------------------------------------------+
Verifying selections...done
Verifying requisites...done
Results...

SUCCESSES
---------
  Filesets listed in this section passed pre-installation verification
  and will be installed.

  Selected Filesets
  -----------------
  bos.net.mobip6.rte 6.1.2.0                  # IPv6 Mobility

  << End of Success Section >>

+-----------------------------------------------------------------------------+
                   BUILDDATE Verification ...
+-----------------------------------------------------------------------------+
Verifying build dates...done
FILESET STATISTICS
------------------
    1  Selected to be installed, of which:
        1  Passed pre-installation verification
  ----
    1  Total to be installed

+-----------------------------------------------------------------------------+
                         Installing Software...
+-----------------------------------------------------------------------------+

installp:  APPLYING software for:
        bos.net.mobip6.rte 6.1.2.0


. . . . . << Copyright notice for bos.net >> . . . . . . .
 Licensed Materials - Property of IBM

 5765G6200
   Copyright International Business Machines Corp. 2001, 2008.
   Copyright BULL 2001, 2008.

 All rights reserved.
 US Government Users Restricted Rights - Use, duplication or disclosure
 restricted by GSA ADP Schedule Contract with IBM Corp.
. . . . . << End of copyright notice for bos.net >>. . . .
```

```
0513-071 The mobip6reqd Subsystem has been added.
Finished processing all filesets.  (Total time:  11 secs).

+-----------------------------------------------------------------------------+
                                Summaries:
+-----------------------------------------------------------------------------+

Installation Summary
--------------------
Name                        Level           Part        Event       Result
-------------------------------------------------------------------------------
bos.net.mobip6.rte          6.1.2.0         USR         APPLY       SUCCESS
bos.net.mobip6.rte          6.1.2.0         ROOT        APPLY       SUCCESS
```

### 7.6.3 Non-shared install

Non-shared install refers to an installation of software that takes place exclusively to a WPAR and only for that WPAR, with its own writable file systems, which are not shared with the global environment. This can be done using the normal install commands or, if the WPAR is detached, by using the **inuwpar** command.

The **inuwpar** command performs software and installation maintenance tasks on all or selected detached WPARS.

## 7.7 Print spooling

Printing from inside a WPAR is supported, although it has some restrictions. Because there are no physical devices inside a system WPAR, printers cannot be local. You cannot connect parallel or serial printers directly to the WPAR.

Because the global environment can manage local printers, and remote printers work inside the WPARs, you can use them. If you only have a parallel or serial printer, then perform the following tasks.

1. In the global environment:
   a. Connect the printer to the appropriate port (parallel or serial).
   b. Install the appropriate software and device drivers.
   c. Create a print queue.
   d. Add the WPAR's host name to the print server: Add Print Access for a Remote Client.
   e. Start the print server.

2. Inside the WPAR:
   a. Create a remote print queue where:

      **Name of QUEUE to add**

      The WPAR's queue name

      **HOSTNAME of remote server**

      The host name of the global environment

      **Name of QUEUE on remote server**

      The global environment's queue name

After the queue has been created within the WPAR, you can view its status by using the **lpstat** command, as shown in Example 7-8.

*Example 7-8 Print queue status from within the WPAR*

```
root@skippy:/# lpstat
Queue   Dev   Status    Job Files              User         PP %   Blks  Cp Rnk
------- ----- --------- --- ------------------ ---------- ---- -- ----- --- ---
wpr1    @sydn READY
```

For standard system WPARs with shared read-only /usr and /opt, install the print software and drivers in the global environment. Then synchronize it with the WPARs using the **syncwpar** command. For system WPARs with writable /usr and /opt file systems, you must install the software and drivers inside the WPAR.

Managing queues inside a WPAR is otherwise the same as that of a standard AIX system. Application WPARs use the global environment's print queues and the required software and drivers, and are managed through the global environment.

## 7.8 System environment notes

Some of the usual configuration settings made in AIX during the initial install of a system, such as changing the date, language or the locale, pose new challenges when considering WPARs and their implementations:

- The console of a WPAR is considered the global environment in which the WPAR is operating. If the proper daemons are running in order to support the required connections, you can log in to a system WPAR using clogin or remotely via ssh, telnet, rlogin, or rsh.

  You cannot change the date or the time inside a WPAR, because there is only one system time and this is governed by the global environments settings.

You can, however, have different time zone settings within each WPAR by changing the TZ variable in the /etc/environment file.

You can have the time zone defined in the global environment for America/Chicago (Central time), and in a WPAR have the variable setting a different time zone, in this case Australia/Canberra for the WPAR named skippy, as seen in Example 7-9.

*Example 7-9 WPAR time zone*

```
root@sydney:/# date
Tue Jul  8 23:22:30 CDT 2008
root@sydney:/# tn skippy
Connected to skippy.
AIX Version 6
Copyright IBM Corporation, 1982, 2008.
login: root
root@skippy:/# date
Wed Jul  9 14:23:17 EET 2008
```

- WPARs support different languages. Install the corresponding language file sets in the global environment.

  If the WPARs have privately writable file systems, then the language file sets will need to be installed to these file systems rather than the global environments.

- Changing any of the characteristics of the operating system from inside the WPAR, such as the number of processes allowed per user or the maximum number of pages in the block I/O buffer cache, is not allowed and an attempt to do so will fail.

  These characteristics are inherited from the global environment and must be changed from the global environment.

- Dump devices do not exist within the WPARs.

  It is not possible to generate a dump inside a WPAR. Dump devices are created and changed in the global environment and any dumps will need to be generated from the global environment.

- The AIX Security Expert tool assists the system administrator to increasing the level of security for the operating system by running scripts that modify several files.

  These files reside in several file systems, some of which in a default configuration will be read-only.

## 7.9 Processes and subsystems

The system administrator can view the system processes using the normal **ps** command or by using **smit process**. Inside a WPAR, the WPAR system administrator can only see the processes that belong to that WPAR.

Example 7-10 shows which processes are visible from within the WPAR. Note that we have an init process /etc/init has PID=1 and PPID=0 inside the WPAR. This is the parent process for all other processes that run or are running in the WPAR.

**Note:** If you created the WPAR with the **-c** option, then **ps** command output in either the global environment or within the WPAR itself will show different process ID values.

*Example 7-10 View the process listing from within the WPAR*

```
root@skippy:/# ps -ef
     UID    PID   PPID   C    STIME    TTY  TIME CMD
    root  90230 393464   0 04:12:35      -  0:00 telnetd -a
    root 221368 282712   0 03:09:20      -  0:00 /usr/sbin/biod 6
    root 282712      1   0 03:09:13      -  0:00 /usr/sbin/srcmstr
    root 352502 282712   0 03:09:20      -  0:00 /usr/sbin/nfsrgyd
    root 372892 282712   0 03:09:19      -  0:00 /usr/sbin/portmap
    root 393464 282712   0 03:09:20      -  0:00 /usr/sbin/inetd
    root 409818      1   0 03:09:22      -  0:00 /usr/sbin/cron
    root 417886      1   0 03:09:13      -  0:00 /usr/lib/errdemon
  daemon 438362 282712   0 03:09:21      -  0:00 /usr/sbin/rpc.statd -d 0 -t 50
    root 458808 282712   0 03:09:19      -  0:00 /usr/sbin/syslogd
    root 487596 282712   0 03:09:23      -  0:00
/usr/sbin/rsct/bin/vac8/IBM.CSMAgentRMd
    root 499886  90230   0 04:12:35  pts/0  0:00 -ksh
    root 614506 499886   0 07:07:25  pts/0  0:00 ps -ef
    root 626886 282712   0 03:09:19      -  0:00 sendmail: accepting connections
    root      1      0   0 03:09:12      -  0:00 /etc/init
    root 639018 282712   0 03:09:22      -  0:00 /usr/sbin/qdaemon
    root 647242 282712   0 03:09:22      -  0:00 /usr/sbin/writesrv
    root 692382 282712   0 03:09:21      -  0:00 /usr/sbin/rpc.lockd -d 0
    root 831632 282712   0 03:09:22      -  0:00 /usr/sbin/rsct/bin/rmcd -a
IBM.LPCommands
```

```
    root 835764 282712   0 03:09:23      -  0:00 /usr/sbin/rsct/bin/IBM.ServiceRMd
```

In the global environment, the same **ps** command run by the system administrator results in a normal view of all processes, including those that belong to the WPARs.

System administrators who want to distinguish between the processes for each WPAR can use the command with the **-@** flag, which will show a new column with the corresponding name of the WPAR.

Example 7-11 shows that the /etc/init directory of the WPAR has a different PID in the global environment, and that the global environment has its own /etc/init.

*Example 7-11 Viewing processes in the global environment*

```
root@sydney:/# ps -ef -@
   WPAR      UID    PID   PPID   C    STIME    TTY  TIME CMD
 Global     root      1      0   0   Jun 18      -  0:17 /etc/init
 Global     root  86190      1   0   Jun 18      -  0:00 /usr/ccs/bin/shlap64
 skippy     root  90258 282712   0   Jul 05      -  0:00 /usr/sbin/lpd
 Global     root  98428      1  17   Jun 18      -  5:50 /usr/sbin/syncd 60
 Global     root 110802 122966   0   Jun 18      -  0:00 /usr/sbin/rpc.lockd -d 0
 Global     root 114824      1   0   Jun 18      -  0:01 /usr/sbin/slp_srvreg -D
 Global     root 122966      1   0   Jun 18      -  0:00 /usr/sbin/srcmstr
 Global   daemon 131174 122966   0   Jun 18      -  0:08 /usr/sbin/rpc.statd -d 0 -t
50
 Global     root 151660      1   0   Jun 18      -  0:13 /usr/lib/errdemon
 Global     root 180394 122966   0   Jun 18      -  0:00 /usr/sbin/nfsrgyd
 Global     root 188572 122966   0   Jun 18      -  0:11 /usr/sbin/aixmibd
 Global pconsole 192650 278722   1   Jun 18      -  5:33 /usr/java5/bin/java -Xmx512m
-Xms20m -Xs
 Global     root 196766 122966   0   Jun 18      -  0:00 /usr/sbin/portmap
 Global     root 200844 122966   0   Jun 18      -  0:00 /usr/sbin/inetd
 Global     root 209018      1   0   Jun 18      -  0:10 /usr/sbin/cron
 Global     root 217338 295146   0   Jun 18      - 85:06 java -cp
/opt/pconsole/pda/daemon/pda_ba
 skippy     root 221368 282712   0   Jul 05      -  0:00 /usr/sbin/biod 6
 Global     root 229512      1   0   Jun 18      -  0:00 /usr/sbin/uprintfd
 Global     root 241818 122966   1   Jun 18      -  2:39 /usr/java5/bin/java
-Xbootclasspath/a:/u
 Global pconsole 245926 192650   0   Jun 18      -  0:00
/opt/pconsole/bin/pconsole_exec
 Global     root 249998 122966   0   Jun 18      -  1:02 /usr/sbin/rsct/bin/rmcd -a
IBM.LPCommand
 Global     root 254160 122966   0   Jun 18      -  0:01 /usr/sbin/syslogd
 Global     root 258212 122966   0   Jun 18      -  0:00 /usr/sbin/snmpd
 Global     root 266420 122966   0   Jun 18      -  0:00 /usr/sbin/rpc.mountd
 Global     root 270506 122966   0   Jun 18      -  0:00 /usr/sbin/nfsd 3891
 Global     root 274636 122966   0   Jun 18      -  0:27 sendmail: accepting
connections
 Global pconsole 278722 311500   0   Jun 18      -  0:00 /bin/ksh
/pconsole/lwi/bin/lwistart_src.
 skippy     root 282712 630974   0   Jul 05      -  0:01 /usr/sbin/srcmstr
 Global     root 291028      1   0   Jun 19   vty0 18:31 -ksh
 Global     root 295146 122966   0   Jun 18      -  0:00 /bin/ksh
/opt/pconsole/bin/pda_daemon
 Global     root 303282      1   0   Jun 18      -  4:10 /usr/bin/xmwlm -L
 Global     root 307394 122966   0   Jun 18      -  0:00 /usr/sbin/biod 6
 Global     root 311500 122966   0   Jun 18      -  0:00 /bin/ksh
/pconsole/lwi/bin/lwistart_src.
 Global     root 315586 122966   0   Jun 18      -  0:00
/usr/sbin/rsct/bin/IBM.ServiceRMd
 Global     root 319672 122966   0   Jun 18      -  0:01 /usr/sbin/snmpmibd
 Global     root 323832 122966   0   Jun 18      -  0:23
/usr/sbin/rsct/bin/vac8/IBM.CSMAgentRMd
 Global     root 327922 122966   0   Jun 18      -  0:00 /usr/sbin/gssd
 Global     root 331980 122966   0   Jun 18      -  0:01 /usr/sbin/hostmibd
 gordon     root 336078 749628   0   Jul 03      -  0:00 /usr/sbin/syslogd
 Global     root 339992 122966   0   Jun 18      -  0:00 /usr/sbin/rsct/bin/IBM.DRMd
 gordon     root 348244 749628   0   Jul 03      -  0:00 /usr/sbin/writesrv
 skippy     root 352502 282712   0   Jul 05      -  0:00 /usr/sbin/nfsrgyd
  kenny     root 356408 430096   0   Jun 19      -  0:01 /usr/sbin/cron
  kenny     root 360488 430096   0   Jun 19      -  0:00 /usr/sbin/srcmstr
 skippy     root 372892 282712   0   Jul 05      -  0:00 /usr/sbin/portmap
  kenny     root 377042 360488   0   Jun 19      -  0:00 /usr/sbin/rpc.lockd -d 0
 skippy     root 393464 282712   0   Jul 05      -  0:00 /usr/sbin/inetd
  kenny     root 401560 430096   0   Jun 19      -  0:00 /usr/lib/errdemon
 skippy     root 409818 630974   0   Jul 05      -  0:00 /usr/sbin/cron
 skippy     root 417886 630974   0   Jul 05      -  0:00 /usr/lib/errdemon
  kenny     root 430096 122966   0   Jun 19      -  0:01 /etc/init
 skippy   daemon 438362 282712   0   Jul 05      -  0:00 /usr/sbin/rpc.statd -d 0 -t
50
  kenny     root 450802 360488   0   Jun 19      -  0:00
/usr/sbin/rsct/bin/IBM.ServiceRMd
 skippy     root 458808 282712   0   Jul 05      -  0:00 /usr/sbin/syslogd
 gordon     root 483394 749628   0   Jul 03      -  0:00 /usr/sbin/inetd
 skippy     root 487596 282712   0   Jul 05      -  0:01
/usr/sbin/rsct/bin/vac8/IBM.CSMAgentRMd
 Global     root 499932 655554  17 10:35:43  pts/0  0:00 ps -ef -@
  kenny     root 516278 360488   0   Jun 19      -  0:00 /usr/sbin/syslogd
 gordon     root 524476 749628   0   Jul 03      -  0:04 sendmail: accepting
connections
  kenny     root 532712 360488   0   Jun 19      -  0:00 /usr/sbin/portmap
  kenny     root 536768 360488   0   Jun 19      -  0:26 sendmail: accepting
connections
  kenny     root 548870 360488   0   Jun 19      -  0:15
/usr/sbin/rsct/bin/vac8/IBM.CSMAgentRMd
  kenny     root 553202 360488   0   Jun 19      -  0:00 /usr/sbin/writesrv
 Global     root 557072      1   0   Jun 20      -  0:00 ./lparmon_v2_aix53 3499
 gordon     root 573572 819336   0   Jul 03      -  0:03 /usr/lib/errdemon
 gordon     root 577588 749628   0 10:32:47      -  0:00 /usr/sbin/nfsrgyd
 gordon     root 618510 749628   0   Jul 03      -  0:00 /usr/sbin/qdaemon
 skippy     root 626886 282712   0   Jul 05      -  0:01 sendmail: accepting
connections
 skippy     root 630974 122966   0   Jul 05      -  0:00 /etc/init
 skippy     root 639110 282712   0   Jul 05      -  0:00 /usr/sbin/writesrv
 gordon     root 643316 819336   0   Jul 03      -  0:00 /usr/sbin/cron
 skippy     root 647352 282712   0   Jul 05      -  0:00 /usr/sbin/qdaemon
  kenny     root 651334 360488   0   Jun 19      -  0:02 /usr/sbin/rsct/bin/rmcd -a
IBM.LPCommand
 Global     root 655554 803022   1 10:34:44  pts/0  0:00 -ksh
  kenny     root 676032 360488   0   Jun 19      -  0:00 /usr/sbin/inetd
  kenny     root 688186 360488   0   Jun 19      -  0:00 /usr/sbin/nfsrgyd
 skippy     root 692382 282712   0   Jul 05      -  0:00 /usr/sbin/rpc.lockd -d 0
  kenny     root 696444 360488   0   Jun 19      -  0:01 /usr/sbin/qdaemon
  kenny     root 708700 360488   0   Jun 19      -  0:00 /usr/sbin/biod 6
 gordon     root 720966 749628   0   Jul 03      -  0:00
/usr/sbin/rsct/bin/IBM.ServiceRMd
 gordon     root 729202 749628   0   Jul 03      -  0:16
/usr/sbin/rsct/bin/vac8/IBM.CSMAgentRMd
 gordon     root 745664 749628   0 10:30:56      -  0:00 /usr/sbin/rpc.statd -d 0 -t
50
 gordon     root 749628 819336   0   Jul 03      -  3:26 /usr/sbin/srcmstr
 Global     root 762092 291028   0   Jul 01   vty0  0:00 telnet 9.3.5.197
 gordon     root 790634 749628   0   Jul 03      -  0:00 /usr/sbin/portmap
 gordon     root 794834 749628   0   Jul 03      -  0:02 /usr/sbin/rsct/bin/rmcd -a
IBM.LPCommand
 Global     root 803022 200844   0 10:34:44      -  0:00 telnetd -a
 gordon     root 815334 749628   0   Jul 03      -  0:00
/usr/sbin/rsct/bin/IBM.AuditRMd
 gordon     root 819336 122966   0   Jul 03      -  0:00 /etc/init
 skippy     root 831632 282712   0   Jul 05      -  0:00 /usr/sbin/rsct/bin/rmcd -a
IBM.LPCommand
 skippy     root 835764 282712   0   Jul 05      -  0:00
/usr/sbin/rsct/bin/IBM.ServiceRMd
```

```
 gordon     root 839684 749628   0   Jul 03      -  0:00 /usr/sbin/biod 6
```

To view processes within a WPAR from the global environment, you can use the **ps** command with the **-@** flag and the WPAR name as an argument, as shown in Example 7-12.

*Example 7-12 Viewing WPAR processes from the global environment*

```
root@sydney:/# ps -ef -@ skippy
   WPAR      UID    PID   PPID   C    STIME    TTY  TIME CMD
 skippy     root 221368 282712   0 12:09:20      -  0:00 /usr/sbin/biod 6
 skippy     root 282712 630974   0 12:09:13      -  0:00 /usr/sbin/srcmstr
 skippy     root 352502 282712   0 12:09:20      -  0:00 /usr/sbin/nfsrgyd
 skippy     root 372892 282712   0 12:09:19      -  0:00 /usr/sbin/portmap
 skippy     root 393464 282712   0 12:09:20      -  0:00 /usr/sbin/inetd
 skippy     root 409818 630974   0 12:09:22      -  0:00 /usr/sbin/cron
 skippy     root 417886 630974   0 12:09:13      -  0:00 /usr/lib/errdemon
 skippy   daemon 438362 282712   0 12:09:21      -  0:00 /usr/sbin/rpc.statd -d 0 -t
50
 skippy     root 458808 282712   0 12:09:19      -  0:00 /usr/sbin/syslogd
 skippy     root 487596 282712   0 12:09:23      -  0:00
/usr/sbin/rsct/bin/vac8/IBM.CSMAge
 skippy     root 626886 282712   0 12:09:19      -  0:00 sendmail: accepting
connections
 skippy     root 630974 122966   0 12:09:12      -  0:00 /etc/init
 skippy     root 639018 282712   0 12:09:22      -  0:00 /usr/sbin/qdaemon
 skippy     root 647242 282712   0 12:09:22      -  0:00 /usr/sbin/writesrv
 skippy     root 692382 282712   0 12:09:21      -  0:00 /usr/sbin/rpc.lockd -d 0
 skippy     root 831632 282712   0 12:09:22      -  0:00 /usr/sbin/rsct/bin/rmcd -a
IBM.LPC
 skippy     root 835764 282712   0 12:09:23      -  0:00
/usr/sbin/rsct/bin/IBM.ServiceRMd
```

The system resource controller (SRC) in a system WPAR works independently of the SRC in the global environment. Therefore, a system administrator in the WPAR can start, stop, list or refresh the SRCs inside the WPAR without any issues.

As shown in Example 7-13 on page 230, after stopping the resource controller group *spooler* in both the WPAR and the global environment, we start it inside the WPAR. As illustrated, the spooler in the global environment is inoperative. Note that in Example 7-13 on page 230, `sydney` is the global environment.

The system administrator can also run a trace on a subsystem within a WPAR.

*Example 7-13 Stopping and starting the different subsystems*

```
root@sydney:/# lssrc -g spooler
Subsystem         Group            PID          Status
 qdaemon          spooler          176230       active
 writesrv         spooler          237698       active
 lpd              spooler                       inoperative
root@sydney:/# stopsrc -g spooler
0513-044 The qdaemon Subsystem was requested to stop.
0513-044 The writesrv Subsystem was requested to stop.
root@sydney:/# lssrc -g spooler
Subsystem         Group            PID          Status
 qdaemon          spooler                       inoperative
 writesrv         spooler                       inoperative
 lpd              spooler                       inoperative
root@sydney:/# tn skippy
Trying...
Connected to skippy.
AIX Version 6

...

root@skippy:/# lssrc -g spooler
Subsystem         Group            PID          Status
 qdaemon          spooler          639018       active
 writesrv         spooler          647242       active
 lpd              spooler                       inoperative
root@skippy:/# stopsrc -g spooler
0513-044 The qdaemon Subsystem was requested to stop.
0513-044 The writesrv Subsystem was requested to stop.
root@skippy:/# lssrc -g spooler
Subsystem         Group            PID          Status
 qdaemon          spooler                       inoperative
 writesrv         spooler                       inoperative
 lpd              spooler                       inoperative
root@skippy:/# startsrc -g spooler
0513-059 The qdaemon Subsystem has been started. Subsystem PID is 647352.
0513-059 The writesrv Subsystem has been started. Subsystem PID is 639110.
0513-059 The lpd Subsystem has been started. Subsystem PID is 90258.
root@skippy:/# lssrc -g spooler
Subsystem         Group            PID          Status
 qdaemon          spooler          647352       active
 writesrv         spooler          639110       active
 lpd              spooler          90258        active
```

**Note:** Because the SRC subsystems in a WPAR operates independently, the subsystems in a WPAR are not viewable or changeable from the global environment.

## 7.10 WPAR resource limiting

An AIX 6.1 global environment can theoretically support up to 8192 resource-controlled workload partitions. The number of non-resource-controlled WPARs that can be operational is restricted solely by the availability of resources available within the global environment, such as memory, adapters and disk space.

The system administrator of the global environment can change the resource controls for a WPAR dynamically by using the **`chwpar`** command.

**Note:** If the limits for processes and threads are reduced on an active WPAR, processes or threads are not terminated. However, creation of new threads or processes will not take place until the total count falls below the new maximum specified limits.

For example, if you have a WPAR with 150 active processes and the administrator changes the total maximum process limit on the WPAR to 130, the the system will not terminate 20 processes. It will, however, prevent the creation of new processes within the WPAR until the process count falls below 130.

You can enable resource controls for a WPAR when it is created. You can also change the controls on an existing WPAR by using SMIT or using the **`chwpar`** command with the **`-R active=yes`** option.

In contrast, you can disable the resource control settings by using the **`chwpar -R active=no`** option. This will disable the resource control but maintain the settings for later use. After the settings are no longer active for a WPAR, the WPAR uses the default settings, which in the case of both processes and threads are unlimited.

## 7.11 IPv6

With the industry starting to adopt IPv6 to alleviate the issues around address availability surrounding IPv4, and the requirement to move to the 128-bit address space, workload partitioning now supports IPv6 addressing. The most notable change is the expansion of the IP address space from 32 bits to 128 bits, enabling virtually unlimited IP addresses. This addressing capability, and other new functions that enable end-to-end security, improved mobility and support around mobility, simplified address configuration and management, make IPv6 a critical component in the evolution of the IT landscape.

For compliance with IPv6 addressing on the part of any of the IBM software product offerings you intend to run in the environment, refer to:

http://www.ibm.com/software/info/ipv6/index.jsp

Workload partitions now have the ability to fully utilize a IPv6 socket connection. Because network connectivity for a WPAR is managed using an alias on a physical adapter attached to the global environment, see 7.12, “Network Name Mapped Interface Support” on page 232 for more detail about this feature.

## 7.12 Network Name Mapped Interface Support

Name Mapped Interface Support is a new feature that allows a system administrator to specify a pseudo network interface name. That name is then used as an argument to the **chwpar** or **mkwpar** commands, and used to bring that network interface up when the WPAR is activated.

This greatly aids the mobility of the WPAR. For example, if a WPAR to be moved has the interface en1 as part of its network configuration, and the if1 interface is not up and available on the original system, then the WPAR will not function correctly. However, now with a defined name mapped interface that will be made available as part of the WPAR on the target system, that issue is removed.

These network interface name mappings are contained in the configuration file /etc/wpars/devmap.

Each line of the devmap file contains a name and a corresponding interface. Example 7-14 shows the entry for the name mapped interface for the `gordon` system WPAR that we have added for this example.

*Example 7-14 The entry for the pseudo interface in /etc/wpars/devmap*

```
root@sydney:/# grep gordon /etc/wpars/devmap
gordon_if1      en0
```

The entries in the /etc/wpars/devmap file can be separated by either spaces or tabs. It makes sense for all global environments that are used for relocation to map the same set of network interface names.

Example 7-15 demonstrates, using the **chwpar** command, how we configure the name mapped interface. It also demonstrates using the **lswpar** command with the **-N** flag to see the detailed network configuration for the WPAR in the global environment.

*Example 7-15 Change the characteristics of the WPAR to include the Name Mapped Interface*

```
root@sydney:/# chwpar -N address=9.3.5.188 interface=gordon_if1 gordon
root@sydney:/# lswpar -N gordon
Name    Interface   Address(6)  Mask/Prefix    Broadcast
---------------------------------------------------------
gordon  gordon_if1  9.3.5.188   255.255.254.0  9.3.5.255
```

This new feature also aids in mobility by making WPAR communications more flexible.

**Note:** If the WPAR is running, then the interface actually in use is displayed. The mapped interface name will only show up when the WPAR is in the Defined state.

When you extract a specification file from an existing WPAR using the **mkwpar** command and the **-e** flag to retrieve and make use of specification settings for the new WPAR that will be created or cloned from an existing WPAR, the specification file will contain the name mapped interface information for that workload partition.

**Note:** It is possible to create a name mapped interface on a defined or inactive system which does not map to a valid system interface. However, if the WPAR in question is in an Active state, an attempt to change the WPAR using the **chwpar** command will fail if the interface is not a valid device on the system.

## 7.13 WPAR static settings resolution

When creating a workload partition from a specification file or from an existing workload partition, certain static settings may conflict. Some of these settings could be name, directory, hostname, network addresses, and network interfaces. The advantage of this feature is to resolve conflicts through implementation of additional flags in **mkwpar**, **wparexec**, **startwpar** and **chwpar**.

## 7.13.1 Sample case

The following use case addressed the potential conflicting settings of name, directory, hostname, and network address(es). The subsequent actions are taken, depending on the specified static settings resolution:

- By default, commands will issue an error message if conflicts are detected without resolving any conflicting settings.
- Commands auto-resolve conflicting static settings, if the auto-resolve flag **`-a`** is used, *except* in the following cases:
  - Name settings conflict when processing a system workload partition.
  - Network address cannot be resolved and hostname does not resolve into an available one.
  - Interface cannot be detected.

To resolve network settings, the WPAR should make use of DNS. The network address and interface will be resolved based on the hostname of WPAR. If the WPAR is network-less (that is, a non-DNS registered one), it will not resolve the network settings.

Example 7-16 shows the creation of system wpar using an existing WPAR without the **`-a`** flag (without resolving conflict settings).

Assume the WPAR named `skippy` exists on the system.

*Example 7-16 Creating a WPAR without resolving conflicts*

```
root@sydney:/# lswpar
Name     State  Type  Hostname  Directory
-----------------------------------------------
skippy  A      S     skippy   /wpars/skippy
```

*The specification file for skippy wpar is:*

```
root@sydney:/# cat /etc/wpars/skippy.cf

general:
        name = "skippy"
        checkpointable = "no"
        hostname = "skippy"
        privateusr = "no"
        directory = "/wpars/skippy"
        auto = "no"
        routing = "no"

network:
        broadcast = "9.3.63.255"
        interface = "en0"
        netmask = "255.255.255.0"
        address = "9.3.63.145"

resources:
        active = "yes"
         :
         :
         :
```

*Try to create the 'kenny' wpar without 'a' flag (without support of static settings resolution). It should return a FAILURE because it is unable to resolve the static settings.*

```
root@sydney:/# mkwpar -n kenny -e skippy
mkwpar: 0960-002 /wpars/skippy already exists.
mkwpar: 0960-318 Workload partition directory /wpars/skippy is not empty.  Quitting.
```

Example 7-17 shows the creation of system wpar using an existing WPAR with the **-a** flag (with resolving conflict settings).

*Example 7-17 Resolving conflicts with WPAR creation*

```
root@sydney:/# mkwpar -n kenny -e skippy -a
mkwpar: 0960-002 /wpars/skippy already exists.
**********************************************************************
RESOLVED
mkwpar: 0960-105 general.hostname must be unique across all workload partitions.
        hostname = skippy also found in the following files:
        /etc/wpars/skippy.cf

**********************************************************************
**********************************************************************
RESOLVED
mkwpar: 0960-105 network.address must be unique across all workload partitions.
        address = 9.3.63.145 also found in the following files:
        /etc/wpars/skippy.cf

**********************************************************************
mkwpar: Creating file systems...
 /
 /home
   :
   :
```

```
Workload partition kenny created successfully.
```

*It resolved the name, directory and hostname attributes along with network address.The specification file for the kenny wpar after creation as below:*

```
root@sydney:/# cat /etc/wpars/kenny.cf

general:
        name = "kenny"
        checkpointable = "no"
        hostname = "kenny"
        privateusr = "no"
        directory = "/wpars/kenny"
        auto = "no"
        routing = "no"

network:
        broadcast = "9.3.63.255"
        interface = "en0"
        netmask = "255.255.255.0"
        address = "9.3.63.26"

resources:
        active = "yes"
            :
            :
```

Example 7-18 shows resolving conflict settings using the **chwpar** command. The WPARs `kenny` and `skippy` exist on the same system. We then try and change the name of the `skippy` wpar to `kenny`, this time by using the **-a** flag to automatically resolve the conflicting static settings.

*Example 7-18 Changing the WPAR to resolve conflicts*

```
root@sydney:/# lswpar
Name     State  Type  Hostname  Directory
-----------------------------------------------
gordon  A      S     gordon   /wpars/gordon
skippy  D      S     skippy   /wpars/skippy
kenny    D      S     kenny     /wpars/kenny
```

*Change the wpar name with already existing wpar name on the system. (After using* **stopwpar -skippy; chwpar -n kenny skippy***)*

```
root@sydney:# lswpar
Name    State  Type  Hostname  Directory
---------------------------------------------
gordon  A      S     gordon    /wpars/gordon
kenny   D      S     kenny     /wpars/kenny
kenny2  D      S     skippy    /wpars/skippy
```

The wpar `kenny` exists, so the **`chwpar`** command with the **`-a`** flag has automatically resolved the name of wpar to `kenny2`.

## 7.13.2 Static settings resolution flow

The implementation flow diagram for the WPAR static settings resolution is illustrated in Figure 7-3 on page 238. This diagram represents the complete description of resolving the conflict settings for the commands **`mkwpar`**, **`wparexec`**, **`chwpar`**, or **`startwpar`**.

The objective of this feature is to address the conflicts of key WPAR properties. When the creation of a WPAR matches with an existing WPAR in the same global environment, this feature will resolve these conflicts automatically. It also resolves the name of the WPAR by incrementing the name of the existing WPAR as shown in Figure 7-3 on page 238.

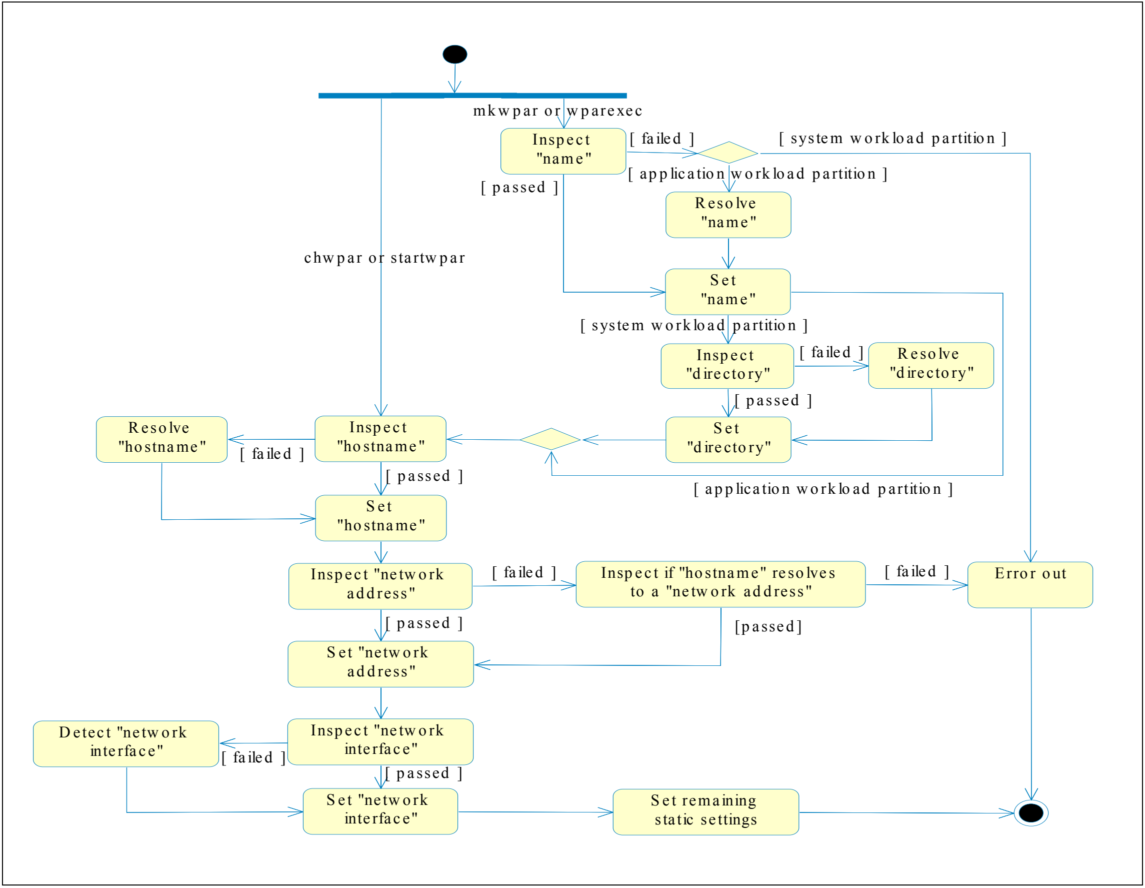


*Figure 7-3 Static settings resolution path*

## 7.14 PowerHA (formerly HACMP) and WPARs

The WPAR offering is supported with IBM PowerHA Cluster Manager (formerly HACMP) v5.4.1. However, particularly in the planning phase, care must be taken because the combination of WPARs and PowerHA in an environment has the potential to introduce new single points of failure (SPOFs) into the environment, such as the NFS server and the WPAR Manager and the networks between these and the WPARs.

**Note:** PowerHA does not manage or monitor the WPAR. It only manages and monitors the applications that run within the WPAR.

## 7.14.1 Planning for High Availability

It is important to plan for your WPARs to be created for mobility, by using the WPAR mobility feature, so that they are not fixed to one particular node. This will involve the introduction of the NFS server required for mobility and the networks around it to be carefully considered in the cluster planning. Note the following points:

- Rather than using service addresses defined in PowerHA, the name-mapped interfaces for the WPARs are defined for the WPAR using the WPAR tools.
- Create and test WPAR mobility before integrating it with PowerHA.
- PowerHA controls the operations of the WPARs such as startup, shutdown, reboot and any mobility of the WPARs between the cluster nodes.
- PowerHA monitors and manages the applications running inside the WPAR, using custom applications monitoring.

We recommend that you plan for and configure NFS cross-mounts for all WPAR file systems. This will force all the WPAR required file systems to be mounted via NFS regardless of which machine is hosting the WPAR at any given point in time. It will also expedite any WPAR movement operations. The failover of a WPAR-based resource group will be faster as a result.

**Important:** If implementing NFS cross-mounts as recommended, ensure that you set the cluster option `File systems mounted before IP is configured` to true.

We also recommend that you make use of the WPAR mobility feature in preference to the default PowerHA resource group integration. As with any highly available implementation, the keys to success are in meticulous planning, implementation, and thorough and ongoing testing.

8

# Resource control

This chapter covers advanced topics related to resource control, tracing, and logging.

The following topics are discussed:

- Resource control
- Workload partition resource control overview
- Workload partition resource control attributes
- Default and recommended values of resource control attributes
- Using resource allocation
- Using WPARs instead of WLM classes
- WPAR resource control changes to WLM
- Frequently Asked Questions (FAQs) regarding WLM and WPAR resource control

## 8.1 Resource control

This section discusses WPAR resource control and presents methods that can be used to allocate processing power and memory resources to WPARs. Proper usage of these methods can prevent potential performance issues in environments where multiple WPARs contend for a shared pool of resources.

The workload partition resource control is based on the Workload Manager (WLM) technology which has been incorporated in the AIX kernel since version 4.3.3. Because the workload partition resource control commands encapsulate and hide WLM details, the system administrator does not need to have in-depth knowledge of WLM in order to use workload partition resource control.

## 8.2 Workload partition resource control overview

In the context of workload partitions, the term *resource control* can be used to specify one of the following:

- The amount of CPU and memory resources allocated to a workload partition
- The number of processes and threads that are allowed to run in a workload partition
- The amount of virtual memory that a single process running in a workload partition can use
- Which *resource set* (rset) to use

**Note:** Resource allocation control for each WPAR is performed at the global environment level by the global administrator. Commands related to resource control are not available within a workload partition.

You can specify resource control attributes using the **`-R`** flag of the **`mkwpar`**, **`chwpar`**, **`wparexec`** and **`lswpar`** commands.

### 8.2.1 CPU and memory allocation to a WPAR

There are two approaches to specifying CPU and memory allocation:

- Share-based

  In this approach, each workload partition receives its part of the specified resource according to the ratio of its own share to the sum of shares of all *currently active* workload partitions.

Note that in this case, the amount of a resource received by a WPAR can vary significantly depending not only on its own shares, but also on *running status* and shares of all other WPARs.

- Percentage-based

  In this approach, there are three parameters that should be specified:

  – Minimum percentage - this refers to the minimum amount of a resource that a WPAR is guaranteed to have available at all times.

    If the WPAR uses less than this amount, WLM will raise the priority of WPAR processes and distribute the unused amount of resource to the global environment. You must specify a value between 0 and 100.

  – Soft maximum percentage - this refers to the maximum amount of a resource that a WPAR can have when multiple WPARs contend for that type of resource. If resource contention does not occur, the WPAR *can* exceed this limit. You must specify a value between 0 and 100.

  – Hard maximum percentage - this refers to the maximum amount of a resource that a WPAR can *ever* have. Even if there is a sufficient amount of that type of resource available and resource contention does not occur, the WPAR *cannot* exceed this limit. You must specify a value between 0 and 100.

**Important:** Keep in mind that when you specify the minimum, soft maximum, and hard maximum percentages, the following rules apply:

- The sum of minimum percentage for all running WPARs must not exceed 100.
- For each WPAR, the minimum percentage must not exceed the soft maximum percentage.
- For each WPAR, the soft maximum percentage must not exceed the hard maximum percentage.

## 8.2.2 Processes and threads in a WPAR

You can specify the maximum number of processes and the maximum number of threads for a WPAR as described in 8.5.4, “Processes and threads” on page 252.

## 8.2.3 Virtual memory of a single process within a WPAR

You can specify the maximum amount of virtual memory that a process may have in a WPAR as shown in Section 8.5.5, “Process virtual memory” on page 253.

### 8.2.4 Resource sets

A *resource set* is used to define a subset of processors in the system. If a resource set is specified for a WPAR, it may use only the processors specified within the resource set.

**Note:** Resource sets are normally used by compute-intensive applications in order to achieve processor affinity. They can also be used for license control in a situation where you are licensed for only a subset of the total CPUs in the system.

## 8.3 Workload partition resource control attributes

Table 8-1 lists the attributes that can specified with the `-R` option when using `mkwpar`, `chwpar`, `wparexec` or `lswpar` commands.

*Table 8-1 Attributes for the -R option*

| **Attribute=value** | **Description** |
|---|---|
| active={yes\|no} | Using active=yes activates resource control for a WPAR.<br>Using active=no saves the rest of attributes specified in the command, but does not activate the resource control. |
| rset=*<rset>* | Instructs the WPAR to use only the specified resource set. |
| shares_CPU=*<n>* | Specifies the number of processor shares available for the WPAR. Shares value can be from 1 to 65535. |
| CPU=*<min>*%-*<sMax>*%,*<hMax>*% | Specifies *<min>* as the minimum CPU percentage, *<sMax>* as the soft maximum CPU percentage and *<hMax>* as the hard maximum CPU percentage for the WPAR. |
| shares_memory=*<n>* | Specifies the number of memory shares available for the WPAR. Shares value can be from 1 to 65535. |

| Attribute=value | Description |
|---|---|
| memory=*<min>*%-*<sMax>*%,*<hMax>*% | Specifies *<min>* as the minimum memory percentage, *<sMax>* as the memory soft maximum percentage, and *<hMax>* as the memory hard maximum percentage for the WPAR. |
| totalProcesses=*<n>* | Specifies the maximum number of processes allowed in the WPAR. |
| totalThreads=*<n>* | Specifies the maximum number of threads allowed in the WPAR. The maximum number of threads must be greater or equal to the maximum number of processes. |
| procVirtMem=*<n>*[MB \| GB \| TB ] | Specifies the maximum amount of virtual memory that a single process in the WPAR may use. |

## 8.4 Default and recommended values of resource control attributes

Because WPAR resource control is based on Workload Manager technology, the default values for resource control attributes are inherited from WLM.

There are cases in which some of these default values may need to be adjusted. For instance, if a CPU-intensive application and an I/O-intensive application are running in WPARs defined on the same LPAR, it may be necessary to define a minimum CPU percentage for the WPAR hosting the I/O-intensive application to ensure that a guaranteed amount of computing power is available even when the CPU-intensive application reaches its peak.

Table 8-2 on page 246 lists the default and the recommended value for each attribute.

*Table 8-2 Default value and recommendation*

| Attribute | Default value | Recommendation |
|---|---|---|
| active | yes | Use the default value for *all* WPARs. If a WPAR uses the `no` value, it is difficult for WLM to have effective control over all resources. |

| Attribute | Default value | Recommendation |
|---|---|---|
| rset | none | Use the default value unless your workload includes computing-intensive applications. |
| shares_CPU | unlimited | Specify for *all* WPARs a value that reflects the importance and weight a WPAR should have among all other WPARs. |
| CPU | 0%-100%,100% | ► Specify an adequate value for the minimum percentage so the WPAR is guaranteed to receive enough CPU resource even when CPU contention conditions occur.<br>► Use the default values for soft/hard maximum. |
| shares_memory | unlimited | Specify for *all* WPARs a value that reflects the importance and weight a WPAR should have among all other WPARs. |
| memory | 0%-100%,100% | ► Specify an adequate value for the minimum percentage so the WPAR is guaranteed to get enough memory resource even when memory contention conditions occur.<br>► Use the default value for soft/hard maximum. |
| totalProcesses | unlimited | Use the default value. |
| totalThreads | unlimited | Use the default value. |
| procVirtMem | unlimited | Use the default value unless you want to restrict the amount of virtual memory that any of your processes may use. |

**Tip:** It is usual to start by specifying only the minimum values for `shares_CPU` and `shares_memory`. After this initial step, the administrator can use the **`wlmstat`** command from the global environment to monitor the utilization for each WPAR and adjust its parameters accordingly. Because the resources can be controlled dynamically, applications running in the WPARs do not experience any downtime.

## 8.5 Using resource allocation

WPAR resource control settings can be modified dynamically at any time. Changes are effective immediately.

### 8.5.1 Resource control command line interface

Values for resource control attributes can be specified with the **-R** option when creating system WPARs using the **mkwpar** command, or application WPARs using the **wparexec** command.

Values for resource control attributes can be modified at runtime by using the **-R** flag of the **mkwpar** command for both types of WPARs.

Values for resource control attributes can be displayed at any runtime by using the **-R** flag of the **lswpar** command for both types of WPARs.

WPAR resource control settings can also be modified or displayed by using the SMIT interface. The fast path is **smitty wpar**.

### 8.5.2 Using CPU resource control

This section discusses and demonstrates how to use CPU resource control features.

#### Using share-based resource control

The share-based approach implies allocating available CPU resources to WPARs based on their importance. The total amount of processing power is conceptually divided in shares. Each workload partition eventually receives a percentage of the total processing power that equals the ratio of WPAR own share to the sum of shares of all *currently active* workload partitions. This implies that the amount of a CPU resources received by a WPAR can vary significantly and depends on the running status and shares of all other WPARs as shown in this simple example:

- Assume there are two WPARs: WPAR A is assigned 5 shares and WPAR B is assigned 10 shares.
  - The total number of shares is 5+10=15.
  - The WPAR A percentage is 5/15=33%.
  - The WPAR B percentage is 10/15=66%.

  Therefore, WLM assigns 33% of the existing CPU resources to WPAR A, and 66% of existing CPU resources to WPAR B.

- A new WPAR named WPAR C is added to the system and assigned 15 shares.
  - The total number of shares is now 5+10+15=30.
  - The WPAR A new percentage is 5/30=17%.
  - The WPAR B new percentage is 10/30=33%.
  - The WPAR C new percentage is=15/30=50%.

It is obvious from this scenario that using share-based resource control requires careful planning. For instance, the impact of adding a third WPAR reduced the CPU resources available to WPAR B to a half of the initial amount; that is, from 66% of the total processing power to 33%.

By default, the number of shares for each WPAR is unlimited, which means that each WPAR is allowed to use as much CPU power as possible. The recommendation is to explicitly set the number of shares that would reflect the importance and weight a WPAR should have among all other WPARs.

Example 8-1 demonstrates how to modify the number of shares for a WPAR by using the **`chwpar -R shares_CPU`** command.

*Example 8-1 Changing CPU shares for a WPAR*

```
# lswpar -R MySysWpar
=================================================================
MySysWpar - Active
=================================================================
Active:                            yes
RSet:
CPU Shares:                        unlimited
CPU Limits:                        0%-100%,100%
Memory Shares:                     unlimited
Memory Limits:                     0%-100%,100%
Per-Process Virtual Memory Limit:  unlimited
Total Processes:                   unlimited
Total Threads:                     unlimited
# chwpar -R shares_CPU=10 MySysWpar
# lswpar -R MySysWpar
=================================================================
MySysWpar - Active
=================================================================
Active:                            yes
RSet:
CPU Shares:                        10
CPU Limits:                        0%-100%,100%
```

```
Memory Shares:                     unlimited
Memory Limits:                     0%-100%,100%
Per-Process Virtual Memory Limit:  unlimited
Total Processes:                   unlimited
Total Threads:                     unlimited
```

**Tip:** We recommend that you start with the share-based approach to control the resources. In almost all cases, your requirements will be met.

## Using percentage-based resource control

Consider using percentage-based controls in these cases:

- When you want to guarantee at all times that a minimum amount of CPU resources is available for a WPAR
- When you want to limit the maximum amount of CPU resources a WPAR can use

The minimum percentage value reflects the amount of CPU resource capacity allocated to a WPAR. If the WPAR uses less than this amount, depending on the WLM setup, one of the following conditions will occur:

- The amount of CPU resources will be reserved for the WPAR's exclusive usage and will not be redistributed to the global environment.
- WLM will raise the priority of WPAR processes and redistribute the unused amount to the global environment. If at any time later the WPAR needs the unused resource it can reclaim it, and because it has the highest priority, it will get it.

The soft maximum percentage reflects the maximum amount of CPU resources a WPAR may ever get when contention conditions for CPU resources occur. If the system has enough CPU power and contention does not occur, then the WPAR may go over this limit.

The hard maximum percentage is the absolute amount of CPU resources that a WPAR may ever have. This limit can *never* be exceeded.

Example 8-2 shows how to modify the percentage-based limits for a WPAR by using the **chwpar -R** command.

*Example 8-2 Changing CPU percentage-based limits*

```
# chwpar -R CPU=6%-30%,45% MySysWpar
# lswpar -R MySysWpar
=================================================================
MySysWpar - Active
```

```
=================================================================
Active:                            yes
RSet:
CPU Shares:                        10
CPU Limits:                        6%-30%,45%
Memory Shares:                     unlimited
Memory Limits:                     0%-100%,100%
Per-Process Virtual Memory Limit:  unlimited
Total Processes:                   unlimited
Total Threads:                     unlimited
```

**Important:** Resource control configuration can be performed using the share-based approach, the percentage-based approach, or both. When using both methods, the percentage-based control has precedence over the shared-based control.

In Example 8-2, if `MySysWpar` were the only WPAR and the shared-based approach were used, then it would get all CPU resources as 10/10=100%. However, because we have also used the percentage-based approach to limit the CPU usage, it will not get more than 45% (hard limit) of the CPU resources.

## 8.5.3 Using memory resource control

Similar to CPU, memory resource controls can be defined both in shares or percentages. The attributes that can be specified have a similar significance.

**Important:** Consider controlling only memory if you have a problem to fix. Otherwise, it is safer to allow AIX to manage memory as normal with its regular techniques of least recently used algorithm and virtual memory. If you are not careful, it is easy to create a memory usage problem that will cause unnecessary paging and performance degradation.

Example 8-3 demonstrates how to modify the number of shares and percentage-based limits for a WPAR by using the **`chwpar -R`** command.

*Example 8-3 Change memory shares and limits*

```
# chwpar -R shares_memory=20 MySysWpar
# lswpar -R MySysWpar
=================================================================
MySysWpar - Active
=================================================================
Active:                            yes
```

```
RSet:
CPU Shares:                        10
CPU Limits:                        6%-30%,45%
Memory Shares:                     20
Memory Limits:                     0%-100%,100%
Per-Process Virtual Memory Limit:  unlimited
Total Processes:                   unlimited
Total Threads:                     unlimited

# chwpar -R memory=5%-100%,100% MySysWpar
# lswpar -R MySysWpar
=================================================================
MySysWpar - Active
=================================================================
Active:                            yes
RSet:
CPU Shares:                        10
CPU Limits:                        6%-30%,45%
Memory Shares:                     20
Memory Limits:                     5%-100%,100%
Per-Process Virtual Memory Limit:  unlimited
Total Processes:                   unlimited
Total Threads:                     unlimited
```

**Tip:** Consider using the percentage-based approach *only* for cases that cannot be addressed via the share-based approach.

- Use the minimum limit to ensure that enough memory is available for an application that uses a low amount of memory, but requires a fast response time. The application may show a slow response if its memory pages have been paged out and need to be paged in.
- Use the maximum limit to contain lower-priority but resource-intensive applications such as batch jobs. Prevent such applications from using memory that may be otherwise required for other applications having higher priority.
- Analyze carefully the benefits and consequences of specifying multiple parameters simultaneously.

## 8.5.4 Processes and threads

You can specify the maximum number of processes and the maximum number of threads for a WPAR as shown in Example 8-4.

If the maximum number of process is not specifically set for a WPAR, the value of the *maxuproc* attribute of the `sys0` object of the global environment will be used for non-root users. The value for root will be unlimited.

*Example 8-4 Setting the maximum number of processes and threads*

```
# chwpar -R totalProcesses=128 MySysWpar
# lswpar -R MySysWpar
=================================================================
MySysWpar - Active
=================================================================
Active:                            yes
RSet:
CPU Shares:                        10
CPU Limits:                        6%-30%,45%
Memory Shares:                     20
Memory Limits:                     5%-10%,20%
Per-Process Virtual Memory Limit:  unlimited
Total Processes:                   128
Total Threads:                     unlimited

# chwpar -R totalThreads=64 MySysWpar
**********************************************************************
**ERROR: resources.totalThreads (64) must be >= totalProcesses (128)
**********************************************************************

# chwpar -R totalThreads=512 MySysWpar
# lswpar -R MySysWpar
=================================================================
MySysWpar - Active
=================================================================
Active:                            yes
RSet:
CPU Shares:                        10
CPU Limits:                        6%-30%,45%
Memory Shares:                     20
Memory Limits:                     5%-10%,20%
Per-Process Virtual Memory Limit:  unlimited
Total Processes:                   128
Total Threads:                     512
```

In this example, the error message shows that the system enforces that the maximum number of threads must be greater or equal to the maximum number of processes.

## 8.5.5 Process virtual memory

You can specify the maximum amount of virtual memory that a process may have in a WPAR as shown in Example 8-5.

*Example 8-5 Setting the maximum amount of virtual memory for a process*

```
# lswpar -R MySysWpar
=================================================================
MySysWpar - Active
=================================================================
Active:                            yes
RSet:
CPU Shares:                        unlimited
CPU Limits:                        0%-100%,100%
Memory Shares:                     unlimited
Memory Limits:                     0%-100%,100%
Per-Process Virtual Memory Limit:  unlimited
Total Processes:                   unlimited
Total Threads:                     unlimited
# chwpar -R procVirtMem=1TB MySysWpar
# lswpar -R MySysWpar
=================================================================
MySysWpar - Active
=================================================================
Active:                            yes
RSet:
CPU Shares:                        unlimited
CPU Limits:                        0%-100%,100%
Memory Shares:                     unlimited
Memory Limits:                     0%-100%,100%
Per-Process Virtual Memory Limit:  1048576MB
Total Processes:                   unlimited
Total Threads:                     unlimited
```

## 8.6 Using WPARs instead of WLM classes

There are situations in which both WLM and WPAR resource control can be used for assigning system resources.

For example, imagine a WebSphere Application Server environment with two running instances called Development and Production in the same LPAR. We can chose to use WLM to assign each application server enough resources so that it could always run in a satisfactory manner.

However, we can also chose a much simpler method. We can define two application WPARs and specify the values for resource control variables as arguments of the **wparexec** command. Assume that we use /home/script/startserver.sh to start the application server. Also assume that we want to allocate 30% and 70% of the existing resources to the Development instance and the Production respectively. We could simply use the following commands:

```
wparexec -R shares_CPU=30 shares_memory=30 /home/script/startserver.sh
devServer

wparexec -R shares_CPU=70 shares_memory=70 /home/script/startserver.sh
prodServer
```

Note that because we use share-based control only when one of the two instances is not running, the other can use all resources available.

## 8.7 WPAR resource control changes to WLM

WLM functions in AIX have been enhanced to support resource control for WPARs.

All WLM concepts addressed in this section are described in more detail in *AIX 5L Workload Manager (WLM)*, SG24-5977.

### 8.7.1 Number of user-defined superclasses

Prior to AIX 6, WLM supported only 69 superclasses, of which 5 were predefined and 64 were user-defined.

WLM implementation in AIX 6 has been extended to support 8192 user-defined superclasses in order to support the maximum number of active WPARs (8192).

**Important:** You may find that commands such as **wlmstat** still work inside the WPAR. But WLM is officially not supported inside WPARs.

## 8.7.2 Resource limit granularity

Because the number of superclasses is above 100, it is necessary to be able to specify resource limits with a granularity finer than an integer percentage number could provide.

WPARs resource control parameters such as *<min>*%, *<sMax>*%, or *<hMax>*% allow now two decimal digits when using the percentage-based approach.

## 8.7.3 Changes to WLM commands

The **wlmstat** command can change its scope and the output depends on whether the command is run in the global environment or in a WPAR:

- The **wlmstat** command run without any parameter displays the information about all classes that are *not* associated with WPARs.
- You can use **wlmstat** command the **-@** option to display information about classes associated with a WPAR. Displaying information for the WPAR having ID 2 is shown in Example 8-6.

*Example 8-6 Using wlmstat command to display WPAR-specific statistics*

```
# wlmstat -@ 2
            CLASS    CPU    MEM   DKIO
        MySysWpar   0.00   0.04   0.00
  MyCompetingWpar   0.24   0.04   0.00
            TOTAL   0.24   0.08   0.00

            CLASS    CPU    MEM   DKIO
        MySysWpar   0.00   0.04   0.00
  MyCompetingWpar   0.44   0.04   0.00
            TOTAL   0.44   0.08   0.00
...
```

## 8.8 Frequently Asked Questions (FAQs) regarding WLM and WPAR resource control

When comparing WLM concepts and WPAR resource control, the following questions are frequently asked.

### 8.8.1 The tier

**Question:**
Since the tier is not specified with the **-R** attribute, how is it used by WLM?
**Answer:**
All WPARs are included by default in tier 0.

**Question:**
Can I change the tier for a WPAR?
**Answer:**
No, this feature currently this is not supported.

### 8.8.2 Superclass and subclass

**Question:**
Since the superclass and subclass are not specified with the **-R** attribute, how are they used by WLM?
**Answer:**
A superclass with the same name as the WPAR (from the **-n** option) is created for each WPAR. There is no subclass explicitly defined, so all processes in a WPAR are in the Default subclass.

### 8.8.3 Class assignment rule

**Question:**
The class assignment rules are not specified with **the -R** attribute, so how can WLM assign a process to a superclass?
**Answer:**
The first process in each WPAR is the init process. It is assigned by the kernel to a superclass that has the same name as the WPAR. All children of the init process and all processes within the WPAR will inherit this superclass classification. As a consequence, all processes in a WPAR are assigned to the same superclass, and processes running in different WPARs are assigned different superclasses.

### 8.8.4 Enabling resource control when creating a WPAR

**Question:**
Do I need to enable resource control when I create a WPAR?
**Answer:**
Resource control is enabled by default. We highly recommend that you keep it enabled. If you disable resource control by specifying **-R active=no**, then WLM will not be able to control and adjust the amount of resources used by processes

in a WPAR. This may lead to processes having no limitations and using as much resource as possible. It is very likely that resource contention conditions will occur. Resource control does *not* add a performance penalty.

### 8.8.5 Enabling resource control for selected WPARs

**Question:**
Can I enable resource control only for certain WPARs?
**Answer:**
We highly recommend that you keep the default setting of **`-R active=yes`** for *all* WPARs, because WLM can control and manage resources most effectively when it controls all WPARs.

### 8.8.6 Using WLM commands directly for resource control

**Question:**
Can I use WLM commands directly to set or change resource settings for a WPAR?
**Answer:**
No. You should use either WPAR Manager GUI or **`mkwpar`**, **`wparexec,`** or **`chwpar`** commands.

### 8.8.7 Using WLM to manage other workloads

**Question:**
Can I use WLM to manage other workloads in the global environment?
**Answer:**
Yes, WLM in the global environment is supported. The **`wlmstat`** command has been enhanced so that resource utilization for WPARs can be monitored using the same interface used for user-defined WLM superclasses and subclasses.

9

# Tracing and logging

This chapter presents the following advanced topics:

- Trace support for WPARs
- Error logging
- System logging

## 9.1 Trace support for WPARs

The trace facility helps you to isolate system problems by monitoring selected system events. Events that can be monitored include entry/exit to/from 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, as well as for 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.

### 9.1.1 System trace enablement for WPARs

System trace is a powerful utility for both administrators and performance tools. The ability to collect selected system activities to report them is beneficial to users and performance tools. This is the main trace facility on AIX that provides Second Failure Data Capture (SFDC) and supports tracing both applications and the kernel. 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. Trace data consists of the values of up to five variables (or expressions involving variables or constants) or the address of a buffer and its length, which are passed as parameters to the trace API. These are directed either to a single system-wide buffer or to per-CPU buffers, which are later copied to the file system.System trace is supported in a standard AIX system, as well as in WPARs.

When system trace is initiated, it dumps the process table with all process IDs, thread IDs and the WPAR ID (CID) to whom the process belongs. The trace header will contain the dump of the currently active WPAR's CID and the name of that WPAR. This will allow trace tools to display the WPAR name in any reports generated.

Whenever a new WPAR is started, its name and CID are stored in the system trace. Also, if a process associates itself to a WPAR (brand()), then the PID and CID get recorded in the system trace. This ensures that the correlation from the TID to the PID, PID to the CID, and then CID to WPAR name is always there to generate the trace report.

The new event trace group (wpar) is added to correlate the trace entry to WPAR ID. This “wpar” event group contains the hookids such as 0x134, 0x139, 0x5D8, 0x210, and 0x465, as explained here:

| | |
|---|---|
| 0x134 | exec() system call |
| 0x139 | fork() system call |
| 0x5D8 | WPAR brand() and config call |
| 0x465 | kthread_create call |
| 0x210 | init process |

Trace subsystem can be initiated in the following ways:

- Global-initiated trace

  The Global administrator initiates the trace for the Global system and collects the trace report.

- WPAR-initiated trace

  The WPAR administrator initiates the trace inside WPAR and collects the trace report based on selected events. This is the same as global-initiated trace.

- Global-initiated WPAR trace

  The Global administrator initiates the trace for WPARs and collects the trace report for selected WPARs.

## Global-initiated trace

The global-initiated system trace is same as in a traditional AIX system. In addition, it will dump the active WPARs list at the top of the trace header along with the WPAR ID and root_path. The root_path shows 0000, which indicates that the trace is initiated from that WPAR.

Inside mobile WPARs, the processes are assigned with a virtual pid and virtual tid. These PIDs and TIDs can be seen as real pids and real tids on the Global system. By ensuring that all PIDs and vPIDs are stored in the trace, along with knowing a trace was run from a WPAR or the Global, the trace library can determine the appropriate PID. It should be able to figure out the PID to return based on the context of where the trace was run. If trace is initiated from the global, real pids will get recorded.

Example 9-1 on page 262 illustrates the global-initiated trace for the Global system only. By default, you do not receive the active WPAR information and process table dump in the trace report.

To view the trace header information with active WPARs and process table dump, you must to comment the line $NOPRINT under the TRACE_UTIL_SECTION from /etc/trcfmt file. If $NOPRINT is not commented, the trace report will not print the header with WPAR name and process table dump.

*Example 9-1 Tracing an application by Global*

```
Start the system trace on global system
# trace -a -J wpar

Run ps command

# ps

Stop the trace
# trcstop

View the trace report using trcrpt command
# trcrpt | more
ID      ELAPSED_SEC     DELTA_MSEC   APPL    SYSCALL KERNEL  INTERRUPT

00A     0.000000000*
00A     0.000000000*
00A     0.000000000*
00A     0.000000000*
00A     0.000000000*

Thu Jul  3 15:32:03 2008
System: AIX 6.1 Node: boss02
Machine: 00CB9CAE4C00
Internet Address: 09034148 9.3.65.72
At trace startup, the system contained 16 cpus, of which 16 were
traced.
Buffering: Kernel Heap
This is from a 64-bit kernel.
Tracing only these hooks, 1340,1390,2100,4650,5d80



trace -a -J wpar

00A     0.000000000* cid=0 name=Global root_path=0000
00A     0.000000000*                               cid=5
name=orca02app01 root_path=/wpars/orca02app01
00A     0.000000000*                              pid=0 ppid=0 tid=3
string=swapper CID=0 vTID=0 vPID=0
```

```
00A     0.000000000*                                pid=1 ppid=0
tid=4099 string=init CID=0 vTID=0 vPID=0
           :
           :
00A     0.000000000*                                pid=98374
ppid=98920 tid=471129 string=dfpd CID=5 vTID=471129 vPID=98374
00A     0.000000000*                                pid=98374
ppid=98920 tid=119221 string=dfpd CID=5 vTID=119221 vPID=98374
00A     0.000000000*                                pid=98374
ppid=98920 tid=328701 string=dfpd CID=5 vTID=328701 vPID=98374

Get the ps command trace hook entry.
# trcrpt | grep ps
00A     0.000000000*                                pid=82860
ppid=107030 tid=197399 string=ps CID=0 vTID=0 vPID=0
134     2.038622466      1.002537          exec:   cmd=ps pid=78002
tid=442487
```

In Example 9-1 on page 262, the trace header contains the active WPAR name `orca02app01`, along with the WPAR ID. The global WPAR entry contains root_path as `0000` to indicate the trace is started from the Global system, along with the WPAR name and cid as zero (0).

The process table contains the process entries with both real and virtual PIDs and TIDs. The trace report contains global processes with a virtual PID/virtual TID as 0. The WPAR process entry has the real process ID and virtual process ID. The **ps** command exec() entry is recorded with the corresponding process ID. The fork() trace hook entry contains the WPAR ID, along with the process ID and thread ID.

## WPAR-initiated trace

WPAR-initiated trace helps WPAR administrators to run trace and collect trace reports based on its WPAR activities. By default, it is not possible to run the trace inside the WPAR until the WPAR has the PV_KER_RAS privilege. The PV_KER_RAS privilege is used to manage RAS-related activities. This ensures the global administrator has control over the authority for the WPAR to run trace.

**Note:** WPAR-initiated trace is supported for system WPARs. Tracing from an application WPAR is not a requirement because application WPARs run in the same environment and potentially share the file system space of the Global system.

Tracing from an application WPAR is not supported.

From within a WPAR, a user only knows about vPIDs. In contrast, within the global environment, the user only knows about the real PID.

When the kernel is processing, it processes based on the real PID. Therefore, most trace hook calls write their trace entries with real PIDs. If trace is run from the global, then all PIDs should be displayed as real PIDs. If trace is run from a WPAR, then all PIDs should be displayed as vPIDs.

Example 9-2 shows an example of running a trace inside a system WPAR. This example shows the trace header contains only this WPAR ID and root_path as 0000 to indicate the trace is started from the global, along with the process table dump.

**Important:** One WPAR's trace entries should not record in another WPAR's trace report. The PIDs/TIDs visible inside WPAR are *virtual* PIDs/virtual TIDs. The WPAR trace records the virtual PID/virtual TIDs for each process hook entry.

*Example 9-2 WPAR-initiated trace*

```
Start the trace by commenting line $NOPRINT under the
TRACE_UTIL_SECTION section in the file /etc/trcfmt.

# trace -a -J wpar

trace the df command with wpar event group
# df
Filesystem    512-blocks      Free %Used    Iused %Iused Mounted on
/dev/fslv00       262144    207304   21%     1614     7% /
/dev/fslv01       131072    128312    3%        5     1% /home
/opt              262144     30472   89%     1977    34% /opt
/proc                  -         -    -         -     -  /proc
/dev/fslv02       262144    257248    2%        8     1% /tmp
/usr             3670016     61904   99%    36906    78% /usr
/dev/fslv03       262144    233176   12%      362     2% /var

Stop the trace
```

```
# trcstop

Print the trace information using trcrpt command.

# trcrpt | more
ID      ELAPSED_SEC     DELTA_MSEC   APPL    SYSCALL KERNEL  INTERRUPT

00A     0.000000000*
            :
Fri Jul  4 14:09:50 2008
System: AIX 6.1 Node: test
Machine: 000C285E4C00
Internet Address: 00000000 0.0.0.0
At trace startup, the system contained 1 cpus, of which 1 were traced.
Buffering: Kernel Heap
This is from a 64-bit kernel.
Tracing only these hooks, 1340,1390,2100,4650,5d80



trace -a -J wpar



00A     0.000000000* cid=3 name=Global root_path=0000
00A     0.000000000*                               pid=0 ppid=0 tid=3
string=swapper CID=0 vTID=0 vPID=0
00A     0.000000000*                               pid=196608
ppid=18874410 tid=1544225 string=ksh CID=3 vTID=41418787 vPID=
19398696
00A     0.000000000*                               pid=209104
ppid=26738688 tid=1040533 string=nfsrgyd CID=3 vTID=57671711 v
PID=58720282
                     :

001     0.000000000  1022648592.560828                   TRACE ON
channel 0
                                                   Fri Jul  4
14:42:18 2008
210     0.000790284      0.790284                  initp: pid=565360
tid=794773 name=trclogio vpid=0 vtid=0 cid=0
139     3.254868058    3254.077774         fork:   pid=458826
tid=1204265 kcid=3
134     3.257337495      2.469437          exec:   cmd=df
pid=24117286 tid=46137435
```

```
139     6.980184390    3722.846895          fork:   pid=458828
tid=978953 kcid=3
134     6.982676126       2.491736          exec:   cmd=trcstop
pid=24641574 tid=46661723
002     7.001649851      18.973725                  TRACE OFF channel
0000 Fri Jul  4 14:42:25 2008
```

The fork (139) trace entry for the **df** command has the cid along with the process id(PID) and thread id (TID).The exec hook entry is recorded with the corresponding PID/TID.

When you start the trace with the WPAR event group, the trace hook (134, 210,139, 5D8, 465) entries are recorded based on the type of application execution. In Example 9-2 on page 264, hook entries such as 134, 210, and 139 are recorded.

As previously mentioned, the WPAR requires the PV_KER_RAS privilege to run the trace. This privilege is checked when trace is started to determine whether the user has been granted the authority to run trace. When a WPAR is started, load the trace kernel extension(trcdd) to start the trace.

**Notes:**

- You can start the trace in a maximum of eight WPARs, including Global, at the same time.
- It is not possible to relocate a WPAR running trace.

## Global-initiated WPAR trace

The Global administrator has the ability to trace specific WPARs, as opposed to the entire system. This reduces the amount of trace entries recorded during a trace period.

To support global-initiated specific WPAR trace, the trace command has a flag **-@**. The **trace** command supports a **-@ [Global,][<WPARnames>]** option that will only collect trace entries from the indicated WPARs.

- If **-@** is not specified, then no filtering will occur based on WPARs.
- If **-@** is specified, you must specify the keyword `Global` to trace the Global system. Otherwise, the Global system's trace entries will not be collected.

If you do not specify **-@**, then all WPARs and the Global system will be traced.

Additionally, the **trace** command has a flag **-W** that is used to store the WPAR ID in each trace entry. Without using this option, the correlation of CID to trace entry will have to occur at report time. It will be based on the utility hook that dumped the PIDs and TIDs into the trace header (now including the CID), and keep track of which TID belongs with which CID throughout the trace entries. This capability only works if the following trace events are being collected: 134, 139, 465, and 5D8.

**Note:** **Trace** command filtering, and storing the additional CID word in each entry, is only supported when the **trace** command is issued from the Global.

Example 9-3 shows the trace filtering on the Global system. The trace is initiated for two WPARs from the Global system. It traces only those two WPARs even if you run applications on the Global and inside other WPARs.

*Example 9-3 Tracing two WPARs from the Global system*

```
Start the trace in two WPARs (shaper, test1)
# trace -a -J wpar -@ shaper,test1
Run applications in three WPARs along with global

# clogin shaper "sleep 1"
# clogin test1 "ps"
    PID    TTY  TIME CMD
# clogin test "sleep 20"
# sleep 3

Stop the trace
#trcstop
The trace report is as below
# trcrpt

Fri Jul  4 16:37:11 2008
System: AIX 6.1 Node: top
Machine: 000C285E4C00
Internet Address: 09033F58 9.3.63.88
At trace startup, the system contained 1 cpus, of which 1 were traced.
Buffering: Kernel Heap
This is from a 64-bit kernel.
Tracing only these hooks, 1340,1390,2100,4650,5d80



trace -a -J wpar -@ shaper,test1
```

```
ID      ELAPSED_SEC     DELTA_MSEC   APPL    SYSCALL KERNEL  INTERRUPT

5D8     0.000000000       0.000000                   brand:  pid=475152
kcid=2
139     0.000462639       0.462639           fork:   pid=548938
tid=1228905 kcid=2
134     0.002192459       1.729820           exec:   cmd=su - root -c
sleep 1    pid=548938 tid=1228905
134     0.053552561      51.360102           exec:   cmd=-ksh -c sleep
1    pid=548938 tid=1228905
139     0.072050914      18.498353           fork:   pid=315432
tid=1904701 kcid=2
134     0.074422147       2.371233           exec:   cmd=termdef
pid=315432 tid=1904701
134     0.083126473       8.704326           exec:   cmd=sleep 1
pid=548938 tid=1228905
465     1.767297356    1684.170883           thread_create:
pid=602154  tid=540865  priority=39
                                                               policy=0
465     1.767298336       0.000980             - thread_create pid
=602154 tid =540865 vtid=540865 vpid=602154 cid=4
465     1.767378445       0.080109           thread_create:
pid=602154  tid=589979  priority=39
                                                               policy=0
465     1.767378771       0.000326             - thread_create pid
=602154 tid =589979 vtid=589979 vpid=602154 cid=4
5D8     9.226914476    7459.535705                   brand:  pid=475154
kcid=4
139     9.227350085       0.435609           fork:   pid=548940
tid=2248779 kcid=4
134     9.228742687       1.392602           exec:   cmd=su - root -c
ps    pid=548940 tid=2248779
134     9.276167557      47.424870           exec:   cmd=-ksh -c ps
pid=548940 tid=2248779
139     9.295378857      19.211300           fork:   pid=315434
tid=2252877 kcid=4
134     9.298169290       2.790433           exec:   cmd=termdef
pid=315434 tid=2252877
134     9.307126309       8.957019           exec:   cmd=ps pid=548940
tid=2248779
```

Notice in Example 9-3 on page 267, the trace report contains only trace entries from two WPARs (shaper,test1). It does not contain the Global and other WPAR (test) entries.

### *Trace report filtering*

With global-initiated WPAR trace, the advantage is to filter the trace report based on a specific WPAR. The **trcrpt** command has a **-@ <WPARList>** option that 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 zero (0) or a WPAR name of Global will display the trace entries for the Global system. The **trcrpt** command also supports the **-O** options of **cid=on** and **wparname=on** to display the WPAR IDs and WPAR names for each entry.

**Note:** Trace report filtering is only supported when the **trace** command is started from the Global.

Example 9-4 shows the trace report filtering along with the WPAR name and cid as on. The filtered trace report shows only the shaper WPAR trace entries instead of all the WPAR entries along with WPAR name and WPAR id(cid).

*Example 9-4 Trace report filtering based on WPAR names*

```
# trace -a -J wpar

# clogin shaper "sleep 2"
# ps
    PID    TTY  TIME CMD
 323712  pts/1  0:00 trace -a -J wpar
 466972  pts/1  0:00 ps
 655482  pts/1  0:00 -ksh

# trcstop

# trcrpt -@ shaper -O cid=on,wparname=on

Fri Jul  4 16:59:33 2008
System: AIX 6.1 Node: top
Machine: 000C285E4C00
Internet Address: 09033F58 9.3.63.88
At trace startup, the system contained 1 cpus, of which 1 were traced.
Buffering: Kernel Heap
This is from a 64-bit kernel.
Tracing only these hooks, 1340,1390,2100,4650,5d80
```

```
trace -a -J wpar


WPAR       CID  ID      ELAPSED_SEC     DELTA_MSEC   APPL    SYSCALL
KERNEL  INTERRUPT

Global     0    001     0.000000000       0.000000
TRACE ON channel 0

Fri Jul  4 16:59:34 2008
shaper     2    5D8     9.124653207    9124.653207
brand:  pid=393402 kcid=2
shaper     2    139     9.125087777       0.434570           fork:
pid=475242 tid=1339555 kcid=2
shaper     2    134     9.127015398       1.927621           exec:
cmd=su - root -c sleep 2    pid=475242 tid=1339555
shaper     2    134     9.171914068      44.898670           exec:
cmd=-ksh -c sleep 2    pid=475242 tid=1339555
shaper     2    139     9.182846984      10.932916           fork:
pid=548956 tid=950341 kcid=2
shaper     2    134     9.185116401       2.269417           exec:
cmd=termdef pid=548956 tid=950341
shaper     2    134     9.193689771       8.573370           exec:
cmd=sleep 2 pid=475242 tid=1339555
Global     0    002    21.487392008   12293.702237
TRACE OFF channel 0000 Fri Jul  4 16:59:55 2008
```

## Basic trace commands

Table 9-1 lists basic trace commands, along with brief descriptions.

*Table 9-1 Basic trace commands*

| Command | Description |
|---|---|
| **trace** | Start the trace |
| **trcstop** | Stop the trace |
| **trcrpt** | Trace report collection |

For more information about trace commands options, refer to the manual pages.

## 9.1.2 Dynamic trace (`probevue`) enablement for WPARs

ProbeVue is a dynamic tracing facility that allows users to dynamically specify trace points and dynamically provide the actions to be executed at the specified trace points. The tracing specifications are written in the VUE programming language, which is the primary interface for users to specify or control the dynamic tracing on AIX. The VUE programming language shares several similarities with the C programming language and the shell script, but also has some unique features.

By default, WPAR has only privileges (PV_PROBEVUE_TRC_USER and PV_PROBEVUE_TRC_USER_SELF) to trace user functions.

To start the dynamic trace on Global/WPAR, you must run the **`probevctrl`** command on the Global system to initiate the dynamic trace, before executing the **`probevue`** command on either GLOBAL or WPAR.

Example 9-5Example 9-5 shows dynamic tracing for user-level functions inside WPAR using the **`probevue`** command.

*Example 9-5 Dynamic trace for use-level functions*

```
Assume the user level program is main.c as listed below. This program
has function foo().
# cat main.c
foo()
{
int t;
t=20;
}

main()
{

while (1)
foo();

}

Assume probevctrl -c trace=on ran on GLOBAL system.
The following VUE structured program prints “Hello, I am testing the
probevue inside WPAR” when the traced process makes foo() function
call. The VUE structured program is shown below
# cat 1.e
@@uft:$1:*:foo:entry
{
```

```
printf("Hello, I am testing the probevue inside WPAR\n");
exit();
}

Tracing the main.c program with probevue

# ./main &
[1]     27787306 --->process id of prg which calls foo() function.

# probevue 1.e 27787306
Hello, I am testing the probevue inside WPAR
```

The Global administrator must provide the corresponding privileges for WPAR to start the dynamic trace for the system calls and kernel. These privileges are PV_PROBEVUE_TRC_SYSCALL, PV_PROBEVUE_TRC_KERNEL, PV_PROBEVUE_TRC_SYSCALL_SELF.

**Important:** The **`probevctrl`** command can be used to start and stop the dynamic trace. This command is allowed only on the Global system.

Example 9-6 demonstrates the tracing open system call from inside WPAR.

*Example 9-6 Dynamic tracing for a open system call*

```
The creation of wpar with the priviliges as below
#mkwpar -n test -S ‘privs+=PV_PROBEVUE_TRC_SYSCALL,
PV_PROBEVUE_TRC_KERNEL, PV_PROBEVUE_TRC_SYSCALL_SELF’ -s

The VUE programming structure is as below

# cat 4.e
@@syscall:$1:open:entry
{
String s[40];
s="IBM\nISL";
printf("%s===>%d\n",s,__pid );
exit();
}

Running the program with probvue command
# probevue 4.e 422082 -->Pid of open system call process
IBM
ISL===>422082
```

In Example 9-6 on page 272, IBM ISL is printed when the process runs the open() system call.

**Note:** Dynamic tracing on WPAR is same as on GLOBAL after the WPAR obtains all the privileges needed to run the dynamic trace.

## 9.2 Error logging

Error logging provides administrators with unparalleled monitoring and reporting on the health and general welfare of system components. It often provides early warning of impending problems, thus providing you with ample time to take action before the problem causes unscheduled downtime or data loss.

The error-logging facility records hardware and software failures in the error log for informational purposes or for fault detection and corrective action. Examples of these errors are file systems running out of space, applications dumping core, and Ethernet adapter failures.

The error logging daemon is normally started during system initialization. Stopping the error logging daemon can cause any error data that is temporarily stored in internal buffers to be overwritten before it can be recorded in the error log file.

Any error written to the AIX error log usually contains more information about probable cause and possible recovery or diagnostic actions. Error logging uses errdaemon to record the errors.The error logging daemon reads error records from the /dev/error file and creates error log entries in the system error log /var/adm/ras/errlog. In addition to s writing an entry to the system error log, the error logging daemon performs error notification as specified in the error notification database /etc/objrepos/errnotify.

The error reporting is enabled on both GLOBAL and WPAR. The errors that belong to WPAR are recorded on both GLOBAL and WPAR. All error reports generated due to an asynchronous event or hardware errors will get logged only on GLOBAL.

WPAR administrators can collect errors based on the configuration they set in the error notification database /etc/objrepos/errnotify. Every WPAR has its own error logging daemon. That daemon becomes active when a WPAR is in the Active state.

Only software errors are recorded in WPARS. Hardware errors are recorded on Global. The Global system will record Global and WPARs errors (both software and hardware).

The error reporting commands **`errlogger`**, **`errpt`**, and **`errclear`** are used for logging errors, generating error reports, and clearing the errors, respectively.

## Error logging commands

Table 9-2 covers the brief description of error logging commands. The **`errpt`** command used to generate the error report. This command supports the new option **`-@ <WPAR list>`** to generate an error report for specific WPARs in which the Global administrator is interested. Note that the flag -@ is supported only on GLOBAL.

*Table 9-2 Error logging commands*

| command | Description |
|---|---|
| **`errlogger`** | Logs operator message |
| **`errpt`** | Generates report of logged errors |
| **`errupdate`** | Updates Error Record Template Repository |
| **`errclear`** | Clears error entries from error log |
| **`errtmplt`** | Contains Error Template Repository |
| **`errstop`** | Terminates error logging daemon. |
| **`errinstall`** | Installs messages in error logging message sets |
| **`errmsg`** | Adds a message to error log message catalog |
| **`errdead`** | Extracts error records from system dump or live dump |

For more detailed information about error logging commands, refer to the manual pages.

## Example of error reporting inside WPAR

Example 9-7 illustrates error reporting inside WPAR. The errdaemon records the core dump error. This error log shows the complete description of the probable cause.

*Example 9-7 Error reporting for coredump- generated application*

```
Assume the progam a.out generates the core dump error.
# ./a.out
Segmentation fault(coredump)
```

The error reporting shows the probable cause in detail as follows.

```
# errpt -a
-----------------------------------------------------------------------
LABEL:          CORE_DUMP
IDENTIFIER:     A924A5FC
Date/Time:       Sun Jul  6 12:36:41 CDT 2008
Sequence Number: 8
Machine Id:      000C285E4C00
Node Id:         shaper
Class:           S
Type:            PERM
WPAR:            shaper
Resource Name:   SYSPROC
Description
SOFTWARE PROGRAM ABNORMALLY TERMINATED

Probable Causes
SOFTWARE PROGRAM

User Causes
USER GENERATED SIGNAL
      Recommended Actions
        CORRECT THEN RETRY

Failure Causes
SOFTWARE PROGRAM

        Recommended Actions
        RERUN THE APPLICATION PROGRAM
        IF PROBLEM PERSISTS THEN DO THE FOLLOWING
        CONTACT APPROPRIATE SERVICE REPRESENTATIVE

Detail Data
SIGNAL NUMBER
          11
USER'S PROCESS ID:
                503996
FILE SYSTEM SERIAL NUMBER
         220
INODE NUMBER
           2
CORE FILE NAME
//core
PROGRAM NAME
```

```
a.out
STACK EXECUTION DISABLED
           0
COME FROM ADDRESS REGISTER

PROCESSOR ID
  hw_fru_id: N/A
  hw_cpu_id: 0

ADDITIONAL INFORMATION
main 14
__start 6C
__start 50

Symptom Data
REPORTABLE
1
INTERNAL ERROR
0
SYMPTOM CODE
PCSS/SPI2 FLDS/a.out SIG/11 FLDS/main VALU/14 FLDS/__start
```

### Error logging implementation flow

Figure 9-1 on page 277 illustrates the implementation flow of the error reporting facility. The handler2 handles hardware-related errors and writes hardware errors only in the Global system buffer.

The WPAR-aware handler1 writes WPAR-related software errors in both the Global and WPAR error buffers. This handler also writes Global system software errors in the Global system error buffer. Subsequently, these Global and WPAR error buffers are written to a file by the error logger.

**Important:** As previously mentioned, WPAR software errors are recorded on both GLOBAL and WPAR. WPAR hardware errors are recorded only on GLOBAL.

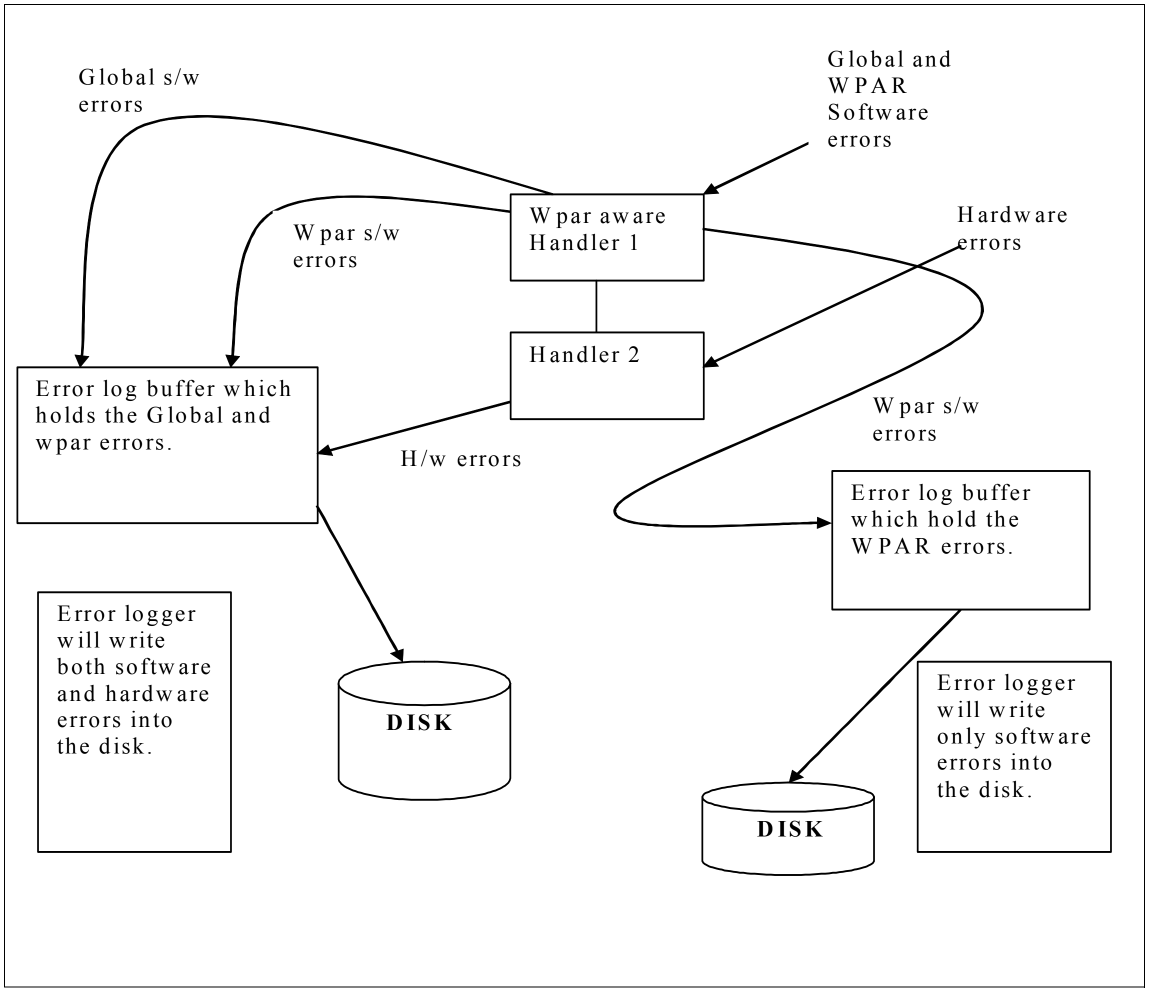

Global s/w errors
Global and WPAR Software errors
Wpar aware Handler 1
Wpar s/w errors
Hardware errors
Handler 2
Error log buffer which holds the Global and wpar errors.
H/w errors
Wpar s/w errors
Error log buffer which hold the WPAR errors.
Error logger will write both software and hardware errors into the disk.
DISK
Error logger will write only software errors into the disk.
DISK

*Figure 9-1 Error logging flow*

## 9.3 System logging

Syslog is a host-configurable, uniform system logging facility. The system uses a centralized system logging process that runs the program /etc/syslogd or /etc/syslog.

Individual programs that need to have information logged send the information to syslog. The messages can then be logged to various files, devices, or computers, depending on the sender of the message and its severity.

This log file can collect a variety of useful information, including panic conditions, data corruption, and hardware errors, as well as warnings and tracking information.

Syslog enables all sources to report to a central location. After the information is centralized, it can be used to build an alerting system or even carry out corrective actions.

When syslogd starts up, it reads its configuration file (usually /etc/syslog.conf) to determine what kinds of events it should log and where they should be logged. syslogd then listens for log messages from three sources: /dev/klog, /dev/log, and UDP port.

Table 9-3 lists system log sources, along with brief descriptions.

*Table 9-3 System log sources*

| Source | Description |
|---|---|
| /dev/klog | A special device used to read messages generated by the kernel |
| /dev/log | A UNIX domain socket used to read messages generated by processes running on the local machine |
| UDP port 514 | An Internet domain socket used to read messages generated over the local area network from other machines |

System logging is enabled on both GLOBAL and WPAR. The messages that belong to WPAR are recorded only on WPAR (unless WPAR sets up remote logging to the GLOBAL system). The messages that belong to Global are recorded in only Global, and not in the WPAR syslog.

The system logging inside WPAR is the same as the system logging on GLOBAL.

The implementation flow for system logging is illustrated in Figure 9-2 on page 279.

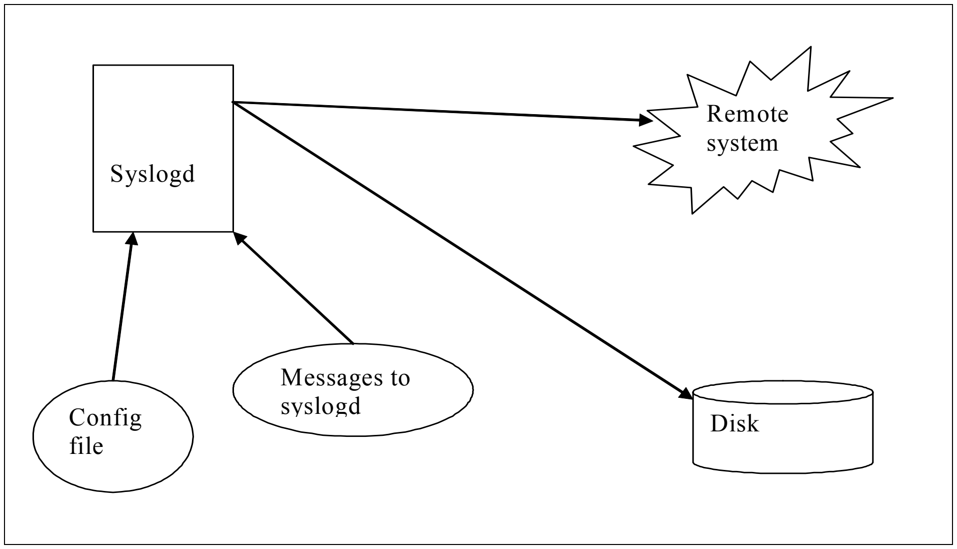


*Figure 9-2 System logging flow*

10

# Developer considerations

This chapter highlights the areas that need particular consideration when you develop your own WPAR applications. The information here is also useful if you need to discuss WPAR application availability with third-party software vendors.

The following topics are discussed:

- Device management
- AIX IOCP API and mobility of I/O completion reports
- Application licensing and compliance support
- WPAR messaging and logging
- XTISO mobility
- NFS client mobility support
- System V and POSIX IPC in WPAR

## 10.1 Device management

This section discusses which devices are available to WPARs and how you can add or remove them from the WPARs. It also covers the fine granularity logical volume (LV) support for the WPARs.

Most applications do not make direct use of devices (that is, files in the /dev directory). Instead, they only deal with files in mounted file systems and communicate over network connections. For those that do access devices directly, the information in this section is relevant.

#### Background

Workload partitions (WPARs) provide isolation of services, applications, and administration by utilizing flexible, software-defined boundaries within a single instance of the AIX operating system. As previously mentioned, there are two types of workload partitions:

- System WPARs are workload partitions that act as complete systems, containing their own file systems and devices.
- Application WPARs are workload partitions that contain a single application; file systems and devices are shared with the global environment.

### 10.1.1 Device visibility in a workload partition

One major difference between operating in the global environment and operating in the workload partition is device visibility. The /etc/wpars/devexports file describes the available devices to WPARs, and also describes which devices are exported by default. That is, this file defines which devices go into a default WPAR.

Example 10-1 shows partial contents of the /etc/wpars/devexports file, and you can see that some devices are commented out.

*Example 10-1 Part of the /etc/wpars/devexports file*

```
...
# The black hole device that throws away any input
device:
  devname = "/dev/null"

# Device that generates zero output
device:
  devname = "/dev/zero"
```

```
# STREAMS:
#device:
#  devname = "/dev/echo"
...
```

You must edit this file if you have different needs. By default, all devices that are *not* commented out with the hash (#) sign character are exported with the **mkwpar** command. Each stanza starts with **device:** and continues with three parameters:

| | |
|---|---|
| **device name** | Required. Examples are /dev/zero, /dev/null (attribute name: "devname") |
| **device type** | Clone or pseudo (attribute name: "devtype") |
| **automatic export option** | Yes or no. If **export** is not specified, **yes** is assumed. (attribute name: "export") |

Editing the /etc/wpars/devexports file will affect all WPARs that you create by default. Alternatively, you can pass different devices to the **mkwpar** command with the **-D** flag.

Example 10-2 demonstrates how to *prevent* the creation, by default, of the /dev/zero device.

*Example 10-2 Editing the /etc/wpars/devexports file*

```
...
# Device that generates zero output
device:
  devname = "/dev/zero"
  export = no
...
```

After the creation of a WPAR, the **mkwpar** command no longer helps you to add new devices to the WPAR. Instead, use the **chwpar** command, as shown in Example 10-3, to change the available devices inside the WPAR.

*Example 10-3 Changing the WPAR's devices*

```
# lswpar
Name       State  Type  Hostname   Directory
---------------------------------------------------
MySysWpar  A      S     MySysWpar  /wpars/MySysWpar
# clogin MySysWpar
# ls -l /dev/zero
ls: 0653-341 The file /dev/zero does not exist.
```

```
# exit
# chwpar -D devname=/dev/zero export=yes MySysWpar
# clogin MySysWpar
# /usr/lib/methods/wpar_cfgpseudo
Importing device /dev/null, type = 1
Importing device /dev/tty, type = 1
Importing device /dev/console, type = 1
Importing device /dev/clone, type = 1
Importing device /dev/sad, type = 3
Importing device /dev/xti/tcp, type = 3
Importing device /dev/xti/tcp6, type = 3
Importing device /dev/xti/udp, type = 3
Importing device /dev/xti/udp6, type = 3
Importing device /dev/xti/unixdg, type = 3
Importing device /dev/xti/unixst, type = 3
Importing device /dev/error, type = 1
Importing device /dev/errorctl, type = 1
Importing device /dev/audit, type = 1
Importing device /dev/nvram, type = 1
Importing device /dev/zero, type = 1
# ls -l /dev/zero
crw-rw-rw-    1 root     system        2,  3 May 11 14:37 /dev/zero
```

The **chwpar** command writes into /etc/wpars/<*name of the WPAR*>.cf, thus making it permanent. When you reboot or stop and start the WPAR, those devices will start with the WPAR.

Example 10-4 shows the output of device-related commands running in the system workload partition.

*Example 10-4 lsdev -C command output in a system workload partition*

```
# lsdev -C
inet0  Available  Internet Network Extension
iscsi0 Defined    iSCSI Protocol Device
lo0    Defined    Loopback Network Interface
lvdd   Defined    LVM Device Driver
pty0   Available  Asynchronous Pseudo-Terminal
rcm0   Defined    Rendering Context Manager Subsystem
sys0   Available  System Object
# ls -la /dev
total 56
drwxrwxr-x    5 root     system         4096 May 03 16:48 .
drwxr-xr-x   17 root     system         4096 May 03 16:48 ..
drwxrwx---    2 root     system         4096 May 03 16:47 .SRC-unix
```

```
-rw--w--w-    1 root     system           0 May 03 16:48 Global
srwxrwxrwx    1 root     system           0 May 03 16:46 SRC
cr--r----T    1 root     system       8,  0 May 03 16:46 audit
crw-rw-rw-    1 root     system      12,  0 May 03 16:46 clone
crw--w--w-    1 root     system       4,  0 May 03 16:06 console
crw--w--w-    1 root     system       6,  0 May 03 16:46 error
crw-------    1 root     system       6,  1 May 03 16:46 errorctl
srw-rw-rw-    1 root     system           0 May 03 16:46 log
crw-rw-rw-    1 root     system       2,  2 May 03 16:48 null
crw-r--r-T    1 root     system       3,  0 May 03 16:46 nvram
crw-rw-rw-    1 root     system      12, 20 May 03 16:46 ptc
drwxr-xr-x    2 root     system       16384 May 03 16:46 pts
crw-rw-rw-    1 root     system      22,  0 May 03 16:46 ptyp0
...
crw-rw-rw-    1 root     system      22, 15 May 03 16:46 ptypf
crw-r--r--    1 root     system      11,  0 May 03 16:46 random
crw-rw-rw-    1 root     system      12, 13 May 03 16:46 sad
crw-rw-rw-    1 root     system       1,  0 May 03 16:46 tty
crw-rw-rw-    1 root     system      23,  0 May 03 16:46 ttyp0
...
crw-rw-rw-    1 root     system      23, 15 May 03 16:46 ttypf
crw-r--r--    1 root     system      11,  1 May 03 16:46 urandom
drwxr-xr-x    2 root     system         256 May 03 16:46 xti
crw-rw-rw-    1 root     system       2,  3 May 03 16:46 zero
```

## 10.1.2 Supported device categories in a WPAR environment

There are five classes of devices in a WPAR environment:

**Global** Globally exported devices are configured in the global environment and are made available for use in the workload partition. Examples of global devices are /dev/zero and /dev/null.

**Global environment only**
Global environment-only devices, such as /dev/kmem, /dev/bus, or /dev/sysdump, are not available inside a WPAR. These types of devices are reserved for the global environment only.

**Physical devices** Physical devices, such as disks, CD-ROMs, or devices dependent on physical devices, such as logical volumes, are only available in the global environment.

**Clone**
Clone devices and those that use the streams subsystem are devices that depend on minor and major numbers, hash keys and, in some cases, a channel number. These devices are created as special files that access the global clone device and return the usable device handle. Devices that provide a transport layer interface to socket-based protocols are an example of clone devices.

**Minor number dependent**
There are several types of devices that are minor number dependent: specific or finite devices and local or buffer-based devices. Some devices have a unique minor number inside the WPAR but not in the global environment. The WPAR accesses them through a dictionary created when the device is created. Errors and traces are examples of local or buffer-based devices. These are basically two-dimensional arrays of buffers with the buffer and the WPAR keyed in.

### 10.1.3 Fine granularity logical volume control

AIX V6.1 TL2 supports the creation of WPARs with additional user-defined logical volume and file system specifications.The advantage of this feature is to create WPARs with low-level control over logical volume attributes (stripe, pps, stale pps, and so on) and file system attributes (inline log, dmapi and so on).

When customers need WPAR file systems with a specific number of physical partitions (PP), PP_SIZE, Schedule_policy, an allocation policy like inter/intra, a specific logical volume name, and a required number of physical volumes, then administrators have an option to create WPARs with all these attributes specified in the image.data file.

The image.data file added by the `mkwpar -L` command utilizes only lv_data and fs_data specifications. Other AIX system install utilities (such as mksysb and mkszfile) use a specification template called image.data to set LVM and file system specifications at installation time. The image.data does not support namefs or remote mounts.

#### AIX install utilities image.data file

As mentioned, AIX system install utilities use a specification template called image.data to set LVM and file system specifications at installation time. This file contains information on the image installed during the Base Operating System installation process. The file is part of System Backup and BOS Install Utilities, and is created by either the `mksysb` or `mkszfile` command.

The image.data file is arranged in stanza format. Each stanza contains one or more fields:

- image_data: contains information about the image format, image type (product image/mksysb).
- logical_volume_policy: instructs the BOS install routines to create file systems specified in the image.datafile or to create the smallest file systems required to contain all the data in the file system.
- ils_data: sets the language used by the BOS install program.
- vg_data: contains information about the volume group from where we took the image. The image.data file contains information about the one volume's group data.
- source_disk_data: contains information about the physical volume. The image.data file contains one source_disk_data stanza for each disk in the root volume group.
- lv_data: contains the information about the logical volume. The image.data file contains one lv_data stanza for each logical volume created on the system.
- fs_data: contains the information about the file system properties. The image.data file contains one fs_data stanza for each logical volume created on the system.
- post_install_data: provides the full path name of a file or command to execute after BOS installation completes.
- post_restvg: specifies the full path name of the file or command to execute after the restvg process completes.

## WPAR image.data file

You can create WPARs having specific logical volume attributes with specific file system attributes by providing the image.data file to the **mkwpar** command with the **-L** flag.

The WPARs will utilize only logical volume and file system stanzas from the image.data file. Other stanza types will be ignored.

The **mkwpar** command should support all logical options supported by image.data specifications, and also support all file system creation options supported by image.data specifications. The supported stanzas (WPAR image.data file) are shown in Example 10-5 on page 288 and Example 10-6 on page 288.

Example 10-5 shows a sample logical volume entry.

*Example 10-5 Sample logical volume entry*

```
lv_data:

        VOLUME_GROUP= rootvg
        LV_SOURCE_DISK_LIST= hdisk0
        LV_IDENTIFIER= 000f88dc00004c00000001076f82f428.5
        LOGICAL_VOLUME= hd2
        VG_STAT= active/complete
        TYPE= jfs
        MAX_LPS= 32512
        COPIES= 1
        LPs= 14
        STALE_PPs= 0
        INTER_POLICY= minimum
        INTRA_POLICY= center
        MOUNT_POINT= /usr
        MIRROR_WRITE_CONSISTENCY= on/ACTIVE
        LV_SEPARATE_PV= yes
        PERMISSION= read/write
        LV_STATE= opened/syncd
        WRITE_VERIFY= off
        PP_SIZE= 64
        SCHED_POLICY= parallel
        PP= 14
        BB_POLICY= relocatable
        RELOCATABLE= yes
        UPPER_BOUND= 32
        LABEL= /usr
        MAPFILE=
        LV_MIN_LPS= 13
        STRIPE_WIDTH=
        STRIPE_SIZE=
        SERIALIZE_IO= no
        FS_TAG=
```

Example 10-6 shows a sample file system entry.

*Example 10-6 Sample file system entry*

```
fs_data:

        FS_NAME= /test6
        FS_SIZE= 131072
        FS_MIN_SIZE= 32768
        FS_LV= /dev/fslv01
        FS_JFS2_BS= 4096
        FS_JFS2_SPARSE= yes
        FS_JFS2_INLINELOG= no
        FS_JFS2_SIZEINLINELOG= 0
        FS_JFS2_EAFORMAT= v1
        FS_JFS2_QUOTA= no
        FS_JFS2_DMAPI= no
        FS_JFS2_VIX= no
```

The attributes listed in these examples are described in the image.data file man pages.

## WPAR logical volume support

The **mkwpar** command supports the creation and customization of file systems with the **–M** and **–f** flags. The **–M** flag supports several file system configurables, including the ability to specify direct options to the **crfs** command. However, it does not support the creation of file systems with advanced logical volume options.

WPAR logical volume support enables you to create file systems with low-level file system options (such as Quotas, Dmapi, inlinelog, EAformat, and sparse) and with specific low-level logical volume attributes (such as stripe, stale pps, pps, lv state and so on). These attributes are covered in WPAR image.data sample LV and FS stanzas.

You may continue to specify the **–M** and **–f** flag with **mkwpar** along with the –i <image.data> functionality.

Prioritization of file system specifications will be as follows (in order of descending priority) for any given file system:

- Mount settings command line arguments (**–M** flag)
- image.data File (**-L** flag)
- Specification File (**-f** flag)
- Default specifications

If the same file system is specified by two or more methods (that is, **–L**, **-M**, **–f**, and default), then the highest priority method will be used and all other specifications will be ignored.

The image.data specifications are not supported for mobile WPARs, which are expected to use remote file systems.

The following guidelines are for stanzas describing WPARs in the image.data file:

- WPARs will only utilize the logical volumes lv_data and file system fs_data specifications in image.data. Other stanza types will be ignored.
- LVs without associated file systems in image.data (that is, raw LVs) will be ignored and a warning will be issued.
- The LV device name specified in image.data is not guaranteed unless it is unique to the given system. If the LV device name matches an existing one, a unique name will be generated and a warning will be issued during processing.
- File system paths (FS_NAME attribute) correspond to how they would be viewed in the WPAR. For example, the root file system is / and the home file system is /home. The global base directory should not be included in image.data paths.
- All file systems must have associated LVs in the image.data file
- All LVs must have associated file systems in the image.data file.

Assume the customer needs a WPAR where the "/" file system will be created with LVname =WPAR_slash, PP size=64, sched policy = parallel and the /home file system should be created with LVNA.

Example 10-7 shows the image.data file where we specify ME=WPAR_home,relocatable=yes. The WPAR using the image.data file shown in Example 10-7 was created by the following command:

```
mkwpar -n banshee -L image_data=/tmp/image.data
```

*Example 10-7 The image.data file for creating a WPAR*

```
lv_data:
        VOLUME_GROUP= rootvg
        LV_SOURCE_DISK_LIST= hdisk0
        LOGICAL_VOLUME= nitro_slash
        VG_STAT= active/complete
        TYPE= jfs2
        MAX_LPS= 512
        COPIES= 1
        LPs= 1
        STALE_PPs= 0
        INTER_POLICY= minimum
```

```
                INTRA_POLICY= middle
                MOUNT_POINT= /
                MIRROR_WRITE_CONSISTENCY= on/ACTIVE
                LV_SEPARATE_PV= yes
                PERMISSION= read/write
                LV_STATE= opened/syncd
                WRITE_VERIFY= off
                PP_SIZE= 64
                SCHED_POLICY= parallel
                PP= 1
                BB_POLICY= relocatable
                RELOCATABLE= no
                UPPER_BOUND= 32
                LABEL= /
                MAPFILE=
                LV_MIN_LPS= 1
                SERIALIZE_IO= no
                FS_TAG= vfs=jfs2:log=INLINE:type=test_wpar1:account=false

lv_data:
                VOLUME_GROUP= rootvg
                LV_SOURCE_DISK_LIST= hdisk0
                LOGICAL_VOLUME= nitro_home
                VG_STAT= active/complete
                TYPE= jfs2
                MAX_LPS= 512
                COPIES= 1
                LPs= 1
                STALE_PPs= 0
                INTER_POLICY= minimum
                INTRA_POLICY= middle
                MOUNT_POINT= /home
                MIRROR_WRITE_CONSISTENCY= on/ACTIVE
                LV_SEPARATE_PV= yes
                PERMISSION= read/write
                LV_STATE= opened/syncd
                WRITE_VERIFY= off
                PP_SIZE= 32
                SCHED_POLICY= parallel
                PP= 1
                BB_POLICY= relocatable
                RELOCATABLE= yes
                UPPER_BOUND= 32
                LABEL= /home
                LV_MIN_LPS= 1
```

```
        SERIALIZE_IO= no
        FS_TAG= vfs=jfs2:log=INLINE:type=test_wpar1:account=false

fs_data:
        FS_NAME= /
        FS_SIZE= 131072
        FS_MIN_SIZE= 35812
        FS_LV= /dev/nitro_slash
        FS_JFS2_BS= 4096
        FS_JFS2_SPARSE= yes
        FS_JFS2_INLINELOG= yes
        FS_JFS2_SIZEINLINELOG= 1
        FS_JFS2_EAFORMAT= v1
        FS_JFS2_QUOTA= no
        FS_JFS2_DMAPI= no
        FS_JFS2_VIX= no
        FS_JFS2_EFS= no

fs_data:
        FS_NAME= /home
        FS_SIZE= 131072
        FS_MIN_SIZE= 32768
        FS_LV= /dev/nitro_home
        FS_JFS2_BS= 4096
        FS_JFS2_SPARSE= yes
        FS_JFS2_INLINELOG= yes
        FS_JFS2_SIZEINLINELOG= 1
        FS_JFS2_EAFORMAT= v1
        FS_JFS2_QUOTA= no
        FS_JFS2_DMAPI= no
        FS_JFS2_VIX= no
        FS_JFS2_EFS= no
```

The command output is displayed in Example 10-8.

*Example 10-8 WPAR creation using image.data file*

```
mkwpar: Creating file systems...
 /
Creating logical volume 'nitro_slash' specified in image.data
Creating file system '/' specified in image.data
 /home
Creating logical volume 'nitro_home' specified in image.data
Creating file system '/home' specified in image.data
```

```
 /opt
 /proc
 /tmp
 /usr
 /var
Mounting all workload partition file systems.
      :
      :
      :
      :
Workload partition nitro created successfully.
mkwpar: 0960-390 To start the workload partition, execute the following
as root: startwpar [-v] nitro
```

The WPAR file systems / and /home are created with the specifications mentioned in the image.data file, as shown in Example 10-9.

*Example 10-9 WPAR file systems created from image.data*

```
# lslv nitro_slash
LOGICAL VOLUME:     nitro_slash            VOLUME GROUP:   rootvg
LV IDENTIFIER:     000c285e00004c000000011a9d22d608.12 PERMISSION:
read/write
VG STATE:           active/complete        LV STATE:       closed/syncd
TYPE:               jfs2                   WRITE VERIFY:   off
MAX LPs:            512                    PP SIZE:        64
megabyte(s)
COPIES:             1                      SCHED POLICY:   parallel
LPs:                1                      PPs:            1
STALE PPs:          0                      BB POLICY:      relocatable
INTER-POLICY:       minimum                RELOCATABLE:    no
INTRA-POLICY:       middle                 UPPER BOUND:    32
MOUNT POINT:        /wpars/nitro           LABEL:          /wpars/nitro
MIRROR WRITE CONSISTENCY: on/ACTIVE
EACH LP COPY ON A SEPARATE PV ?: yes
Serialize IO ?:     NO

#  lslv nitro_home
LOGICAL VOLUME:     nitro_home             VOLUME GROUP:   rootvg
LV IDENTIFIER:     000c285e00004c000000011a9d22d608.13 PERMISSION:
read/write
VG STATE:           active/complete        LV STATE:       closed/syncd
TYPE:               jfs2                   WRITE VERIFY:   off
```

```
MAX LPs:            512                     PP SIZE:        32
megabyte(s)
COPIES:             1                       SCHED POLICY:   parallel
LPs:                1                       PPs:            1
STALE PPs:          0                       BB POLICY:      relocatable
INTER-POLICY:       minimum                 RELOCATABLE:    yes
INTRA-POLICY:       edge                    UPPER BOUND:    32
MOUNT POINT:        /wpars/nitro/home       LABEL:
/wpars/nitro/home
MIRROR WRITE CONSISTENCY: on/ACTIVE
EACH LP COPY ON A SEPARATE PV ?: yes
Serialize IO ?:     NO
```

As the preceding examples show, you can create WPAR file systems with specific user-defined low-level logical volume options.

## 10.2 AIX IOCP API and mobility of I/O completion reports

A number of important applications such as Lotus® Notes® and WebSphere use the AIX IOCP API as an alternative to either select(2) or poll(2). The purpose of this feature is to add support to the AIX kernel and IOCP kernel extension as needed by Metacluster Checkpoint and Restart (MCR) to be able to checkpoint and restart the IOCP state for these applications.

The select(2)/poll(2) system calls can be used with sockets to provide a synchronous multiplexing mechanism.

The AIX IOCP API fdinfo() system call with new commands can be used with sockets to provide the asynchronous I/O communication mechanism.

### Metacluster Checkpoint and Restart

Metacluster Checkpoint and Restart (MCR) is the software that provides the capability to *checkpoint* (capture the entire state of running applications to be relocated) in a workload partition to a statefile. You can then restart the applications using the information in that statefile on another logical partition or managed system. MCR enables the WPAR mobility on the system.

The following section briefly describes I/O completion ports. Refer to 10.2.1, “IOCP mobility requirements” on page 296 for an explanation of the AIX IOCP API.

## I/O completion ports

Input/Output completion ports offer an asynchronous model of communication. The asynchronous nature of the model offers parallelism, and provides a performance advantage in a multithreaded application environment.

I/O completion ports provide an efficient threading model for processing multiple asynchronous I/O requests on a multiprocessor system. When a process creates an I/O completion port, the system creates an associated queue object for requests whose sole purpose is to service these requests.

Processes that handle many concurrent asynchronous I/O requests can do so more quickly and efficiently by using I/O completion ports in conjunction with a preallocated thread pool than by creating threads at the time they receive an I/O request.

## Synchronous I/O versus asynchronous I/O

In synchronous input/output (I/O), a thread starts an I/O operation and immediately enters a wait state until the I/O request has completed. Application processing cannot continue until the I/O operation is complete.

In contrast, asynchronous I/O (AIO) operations run in the background and do not block user applications. This improves performance because I/O operations and application processing can run simultaneously.

Figure 10-1 on page 296 illustrates the difference between synchronous I/O and asynchronous I/O.

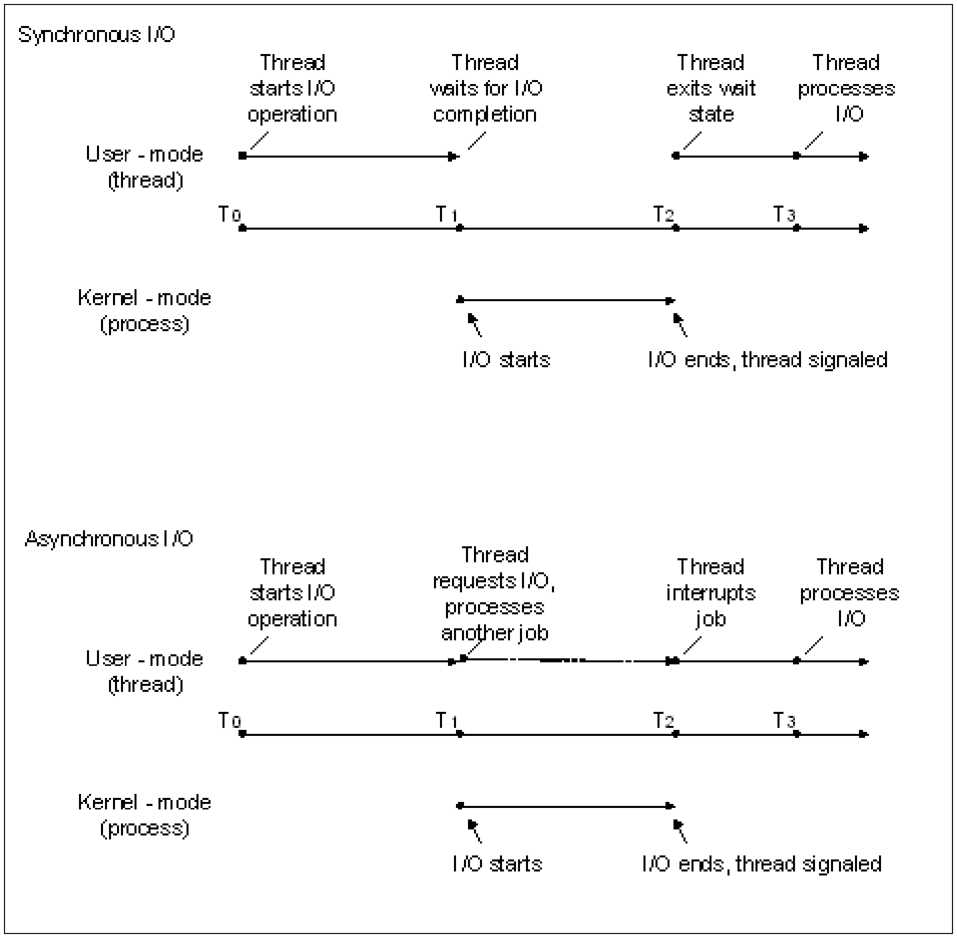


*Figure 10-1 Synchronous I/O versus asynchronous I/O*

### 10.2.1 IOCP mobility requirements

AIX must provide a way for MCR to capture the state of any IOCP completion port and its associated file descriptors so that the state can be saved on checkpoint, to be restored on restart. This functionality requires the addition of new commands (FDINFO_GET_IOCPFDS, FDINFO_GET_IOCPPKTS, FDINFO_PUT_IOCPPKTS) to an existing fdinfo system call. There are three requirements to support this feature, as explained in the following sections.

#### IOCP state capture and restore

AIX currently provides a facility to gather state data about open file descriptors in the current process during a checkpoint operation, and then to restore this data on restart. The fdinfo system call supports a number of specific commands that are used to do this. Support for IOCP will necessitate adding additional commands to the fdinfo system call to support its unique requirements.

During a checkpoint operation, the new interface must support the following requirements:

- Retrieve a list of all file descriptors that have been added to the IOCP completion port file descriptor.
- Retrieve the queued I/O completion and request packets and data for all file descriptors attached to an IOCP I/O Completion Port.
- During a restart operation, the new interface must provide a way to requeue the saved IOCP I/O completion and request packets to the IOCP completion port for each file descriptor.

#### Inherited file descriptors

To support inherited file descriptors for files that are attached to an IOCP Completion Port that have been inherited but are not allocated in the context of the current process, you need to allocate a file descriptor for this process and mark it to be closed in the file descriptor list that is output for MCR. This is so that the MCR code will know to close these file descriptors when the checkpoint is complete. Also, upon restart, MCR will be able to reopen these file descriptors and reattach them to the completion port before closing them.

#### Enable sleeping IOCP processes

Processes that are sleeping while waiting for IOCP completion status must be marked as interruptable by the checkpoint signal.

## 10.2.2 IOCP mobility APIs

The application programming interface fdinfo() with newly introduced commands is used for IOCP mobile WPARs. This section briefly describes fdinfo() and wpar_iocp_oper().

#### *fdinfo system call*

The fdinfo() system call is used to obtain and set information about open file descriptors in the current process. This section explains the fdinfo() commands (as specified via the **cmd** parameter) that were added to provide support for WPARs that use the AIX IOCP functionality.

```
#include <sys/fdinfo.h>
#include <sys/mcr_api.h>
int   fdinfo(pid_t pid, int cmd, void *arg)
```

Following are the parameters:

|  |  |
|---|---|
| **pid** | The process ID. As currently implemented, this must be for the current process. |
| **cmd** | One of the following new fdinfo() commands: |
|  | FDINFO_GET_IOCPFDS |
|  | Given an IOCP Completion Port file descriptor, return the list of file descriptors that are attached to this IOCP Completion Port. |
|  | FDINFO_GET_IOCPPKTS |
|  | Save the IOCP completion packets and requests that are associated with the specified IOCP Completion Port. |
|  | FDINFO_PUT_IOCPPKTS |
|  | Restore the IOCP completion packets and requests to the specified IOCP Completion Port. |
| **arg** | The address of a fdinfo_io_iocp structure that specifies the IOCP Completion Port file descriptor, the address and size of the out-put buffer and, for the FDINFO_PUT_IOCPPKTS command, the address of a the fdinfo_fd2key structure used to describe the mappings between the file's fkey (reported by the FDINFO_GET_IOCPFDS command) and the process file descriptors. |

## Return values

On success, a value of zero (0) is returned. On failure, a value of -1 is returned and the global variable errno is set to identify the error.

### *wpar_iocp_oper function*

The wpar_iocp_oper function serves as “glue” for the three WPAR IOCP functions. It is responsible for copying the fdinfo() parameters between the user space and the kernel space. Where necessary, it also converts them between 32-bit and 64-bit values.

Based upon the fdinfo() cmd value, it calls the appropriate WPAR function to interface to the IOCP kernel extension.

```
int wpar_iocp_oper(pid_t pid,int cmd, void *arg);
```

Parameters:

| | |
|---|---|
| **pid** | Process ID. As presently implemented, this must be for the current process. |
| **cmd** | One of the new fdinfo commands as previously described. |
| **arg** | Pointer to the fdinfo_io_iocp structure, as previously described, containing the input arguments. |

## Return values

On success, a value of zero (0) is returned. On failure, a value of -1 is returned and the global variable errno is set to identify the error.

These APIs are communicated to the kernel functions, and they perform the actions needed to provide IOCP mobility for WPARs.

Example 10-10 shows the data structures used by the fdinfo() IOCP support APIs.

*Example 10-10 fdinfo() IOCP support API data structures*

```
/* File descriptor to key mapping */
     typedef struct fdinfo_fd2key
     {
             int             fd2key_nbr;     /* Number of entries */
             struct fd2key_map {
                     long long       fkey;   /* key to identify the file     */
                     int             fd;     /* File descriptor              */
             } fd2key_list[0];
     } fdinfo_fd2key_t;
typedef struct fdinfo_io_iocp
     {
             int fd_iocp; /* file descriptor (HANDLE) for IOCP Completion port
*/

             fdinfo_fd2key_t *fd_fd2key;     /* mappings from fd to fkey, used
                                                with FDINFO_PUT_IOCPPKTS only*/
             void            *fd_iobuf;      /* i/o buffer                   */
             size_t          fd_iosize;      /* Size of fd_iobuf             */
     } fdinfo_io_iocp_t;
typedef struct fdinfo_iocp
     {
             int             fd;             /* file descriptor              */
             long long       fkey            /* file key to identify file    */
             uint64_t        fd_iocp_flags;  /* iocp related flags */
     #define FDINFO_IOCP_CLOSEFD     1
   /* fd opened by fdinfo and should be closed after c/r */
             uint64_t        fd_iocp_key;    /* CompletionKey value          */
```

```
        uint64_t        fd_iocp_cthrds; /* ConcurrentThreads value      */
    } fdinfo_iocp_t;
```

## 10.3 Application licensing and compliance support

AIX provides facilities through which an application may present and register acceptance of licensing agreements associated with installed software. These facilities must function properly within a WPAR. There are no changes required to support software license agreements in a WPAR. Note there are many software applications that run on AIX and do not use the standard AIX licensing features, so verify directly with the software vendor in these instances.

Each WPAR is seen as a separate instance of the operating system. Configuration and administration of IBM License Use Management (LUM) licenses should work the same on WPAR.

Global environment administrators may, at their discretion, grant WPAR administrators the necessary licenses to run certain applications. In order for the WPAR to access these licenses, the licenses must be stored in a shared location. For other applications, the WPAR administrator may obtain specific licenses and store them in a non-shared location that only that WPAR can access.

Refer to 10.3.1, “Application licensing support for WPAR” on page 300, for an overview of IBM license use management and IBM Tivoli compliance.

### 10.3.1 Application licensing support for WPAR

This section explains the role of the License Use Management product in supporting WPARs.

#### IBM License Use Management

The IBM License Use Management (LUM) product allows software products to be *license-enabled*. A software vendor license enables an application that checks at runtime to see whether the product-specific license is available.

The licenses come in two basic forms: nodelocked licenses and network licenses:

- A nodelocked license is stored on the system where the product is installed for use on that system.

- A network license is granted by a server to any client in the network.

The applications running within WPAR must be able to acquire the licenses in the same manner as those running on global environment via nodelocked and networked licenses.

The license server listens on a port for license requests from applications. Because a WPAR shares the machine identifier from the global environment, running a license server on a WPAR could result in a server distributing more licenses than it would normally be authorized to distribute.

LUM uses the CPU ID in several important ways:

1. For network-based licenses, the licenses are created to be served by a system with a particular TargetID, of which one of the components is the CPU ID. If the license is moved to a server with a different CPU ID, the license is invalid.
2. For nodelocked licenses (stored in /var on the system where the application is running), a license may be tied to a system with a particular TargetID.

### *Implications*

- Because the CPU ID on a WPAR is the same as in the global environment, existing LUM network servers would compute the same TargetID in a WPAR as on a global system. This would result in overserving licenses if the license server database was copied into the WPAR and the license server started.

  Consequently, old versions of LUM must be prevented from starting in a WPAR, and new versions must either be prevented from starting in a WPAR or some mechanism must be added to LUM to make it WPAR-aware (it could compute a TargetID or a similar identifier for a particular WPAR under option, for example).
- License servers that were moved to a new system via WPAR mobility or LPAR partition migration would fail to function on the new system because the TargetIDs would not match.
- Applications that are running under a nodelocked license (or any type of license tied to a particular CPU on the system where the process is executing) would need to have a valid license on the system to which the application is moved. (This no different than the requirements for moving offline applications to a new system.)

To resolve these implications, LUM uses a unique and stable WPAR ID for its operations. The stable identifier for a WPAR is its LPAR ID combined with its WPAR key.

The WPAR key is available through the newly supported **`W`** flag for **`uname`**. **`uname -W`** internally calls the corral_getckey() system call.

## 10.3.2 Tivoli License Compliance Manager for WPARs

This section explains the role of the Tivoli License Compliance Manager product in supporting WPARs.

### IBM Tivoli License Compliance Manager

The IBM Tivoli License Compliance Manager product provides software inventory, use metering, and license allocation services. Information about installed software and software use is collected from monitored computers by an agent, and is reconciled with user-defined license information.

The Tivoli License Compliance Manager can perform scans of installed software on monitored computers using Common Inventory Technology (CIT) software. The agent is able to assemble information about the level and duration of use of a product. The level of use can be metered by the number of users and number of processors.

Tivoli License Compliance Manager provides software inventory, use metering, and license allocation services. The license manager also provides an agency to create and assign the licenses to instances of software products. This licensing technology includes capabilities to license on various partitioning technologies, such as LPAR, DLPAR, shared pools, and micropartitioning. In order for Tivoli License Compliance Manager to function properly in a WPAR, that level of system information must be made available within a WPAR.

The CIT software currently collects the following system processor information for use by Tivoli licensing products:

**ComponentID**

| | |
|---|---|
| Product | uname -m |
| Serialnumber | uname -u |
| Manufacturer | uname -M |
| Type | odm_get_first(…, "attribute='modelname",…) |
| ModelClass | getconf MODEL_CLASS |
| WPAR key | uname -W |

**LPAR**

| | |
|---|---|
| LparID | sysconfig(SYS_GETLPAR_INFO,...) |
| SharedPoolID | lpar_get_info |
| NodeCapacity | lpar_get_info or _system_configuration.ncpus or "/usr/lib/boot/bin/dmpdt_chrp \| awk |

| | |
|---|---|
| | '/ibm,lrdr-capacity/{getline; getline; print $1}'" or sysconf(_SC_NPROCESSORS_CONF) |
| LparCapacity | lpar_get_info or sysconfig(SYS_GETLPAR_INFO,...,...) |
| SharedPoolCapacity | lpar_get_info |
| SerialNumber | lscfg_sysser |

**Processor**

| | |
|---|---|
| Number of processors | _system_configuration.ncpus |
| ID | MD5(type,speed) or hard coded or lscfg -vl cpucard0 |
| CpuEnabled | PAL_libcfg() or lscfg -vl cpucard0 or hard-coded |
| Family | PAL_libcfg() or lscfg -vl cpucard0 or hard-coded |
| MaxClockSpeed | PAL_libcfg() or lscfg -vl cpucard0 or hard-coded |
| CurrentClockSpeed | PAL_libcfg() or lscfg -vl cpucard0 or hard-coded |

**Physical processor**

| | |
|---|---|
| CorePerPackageCount | hard-coded (based on processor type) |
| LogicalProcPerCore | hard-coded (2 for POWER5, POWER5+, 1 otherwise) |
| table count | NodeCapacity / CorePerPackageCount |
| ID | 1..table count |
| Manufacturer | IBM |
| Family | PAL_libcfg() or hardcoded POWERN |
| CpuFreq | PAL_libcfg() |
| L2CacheSize | _system_configuration.L2_cache_size() or hardcoded for POWERN |
| L3CacheSize | hardcoded per proc type |
| BrandName | PAL_libcfg() or "unknown" |

In WPARs, you can obtain the full componentID information and lpar_get_info() information but this does not fully and accurately report the system identity and operating capacity for processes running within a WPAR. CIT on a WPAR cannot report storage characteristics because it is not possible at this time to manage storage devices in a WPAR. All of that management has to occur in the Global environment.

There are two identifiers associated with a given WPAR. When a WPAR is created, it is given a static identifier called a *ckey*, which can be used to identify that WPAR in persistent storage on a given system. The ckey is stored in the /etc/wpars/index file.

When that WPAR is loaded, it is given a dynamic identifier called a *cid* to differentiate it from other WPARs loaded at that time in the kernel. For the purposes of identifiers that must be persistent across WPAR reboots, the ckey must be used. Software licensing systems require a persistent identifier to identify the operating system. The persistent identifier for identifying the operating system is the WPAR key.

To achieve software licensing support for WPAR, a flag was introduced for the existing commands **uname** and **lparstart** which gets the unique identifier to distinguish the WPARs, as explained here.

### *uname -W*

The **uname -W** command displays zero (0) for the global environment, and displays the persistent identifier (the ckey) otherwise. When the **-W** flag is processed, **uname** will invoke a new system call *corral_getckey()* to query the kernel to get the current WPAR key (the persistent ID). Example 10-11 shows the global environment key and WPAR key.

*Example 10-11 The uname -W command*

```
On Global system:
# uname -W
0

On WPAR:
# uname -W
56
```

### *lparstat -W*

The **lparstat-W** command displays the information about the WPAR CPU capacity. Specifically, **-W** provides the following information for a WPAR:

| | |
|---|---|
| **WPAR Key** | WPAR static identifier |
| **WPAR Configured ID** | WPAR dynamic identifier |
| **WPAR Maximum CPUs** | Number of CPUs in resource set or 0 if unrestricted |
| **WPAR CPU Percentage** | WPAR CPU Limit (%) |

Example 10-12 displays WPAR CPU capacity information about the global and WPAR environments.

*Example 10-12 The lparstat -W command*

```
Global environment:
# lparstat -W
WPAR Key                                   : 0
WPAR Configured ID                         : 0
WPAR Maximum Logical CPUs                  : 0
WPAR Maximum Virtual CPUs                  : 0
WPAR Percentage CPU Limit                  : 100.00%
WPAR environment:

# lparstat -W
WPAR Key                                  : 56
WPAR Configured ID                        : 1
WPAR Maximum Logical CPUs                 : 0
WPAR Maximum Virtual CPUs                 : 0
WPAR Percentage CPU Limit                 : 100.00%
```

## Inventory scan requirements on WPAR

ITLM performs an inventory scan of the installed products on the system (WPAR, in this case). This inventory scan works by matching data from a knowledge base to data available on the system.

There are currently two main types of resources that are matched:

- Files on filesystem

  The match is based on a comparison of filename and size. It is key for ITLM to be able to detect files that are related to the applications that are installed in the WPAR where the scan is run, and to skip applications that are not actually installed in the WPAR.

- Installation registry keys (for example, ISMP keys, VPD keys, and so on)

  Keys need to be available from within the WPAR where the scan is run, even if the application has been installed from outside the WPAR.

## Software application license usage monitoring

ITLM monitors software application usage in three ways, as explained here.

### *Native processes*

The agent performs a periodic scan of the /proc filesystem and gathers information about running executables (filename, file size, path/inode of the executable, user and group of the process). This information is matched against the knowledge base to associate a product to the executable. The ITLM Agent needs to be able to detect all (and only) the processes that are running within the WPAR where the agent runs.

The processes that are running in WPARS are visible to the global system.The agent runs in the global environment needs to skip those processes to avoid data duplication.

In order for the process scan to skip the processes that are not running in the zone where the agent is running, it will compare the WPAR ID (the cid) for the processes that it scans against the WPAR ID where the scan is running and also against any application WPARs that are running in that environment, because application WPARs run within the operating system context of the global.

The additional comparison for application WPARs is only relevant if the agent is running in the global environment (where cid = 0), because application WPARs cannot be created within system WPARs at this time. Even if they were, any processes that would be visible would rightfully belong to the WPAR in which the agent was running. The changes made to make the dynamic WPAR identifier(CID) for a process visible via getprocs64(). The agent will get the CID using corral_getcid() call.

If corral_getcid() returns 0 in the agent, meaning that it is running in the context of the global environment, then for every process for which information is returned by getprocs64(&pinfo,...) which has a pi_cid field which is non-zero, it must discard those processes unless corral_getinfo(CAPPWPAR, pinfo.pi_cid, &isappwpar, sizeof(boolean_t)) returns a value of 1 for is application WPARWPAR.

It is suggested that corral_getinfo() should only get called once for a given WPAR, because a system WPAR will never change into an application WPAR and vice versa. So saving the value will reduce the number of required calls.

#### *Instrumented Java applications*

These applications explicitly notify the ITLM Agent start and stop events. The notification mechanism is based on a TCP/IP connection on a local (127.0.0.1) port. The only thing that must be guaranteed by WPAR is that the loopback device in each WPAR is available and different from the other WPARs, to make applications connect to the correct ITLM Agent.

##### *Web applications running in WebSphere Application Server containers*

These applications are monitored by means of a Java subagent that discovers WebSphere Application Server instances and monitors the associated servers. This should not be impacted by WPARs. However, if the WebSphere Application Server installation is modified to support WPARs, code could probably be implemented to make the subagent work with the modified installation.

## 10.4 WPAR messaging and logging

WPAR messaging and logging is a feature that logs the messages based on WPAR command execution. It is helpful for debugging problems that are generated by WPAR command execution. It is also helpful to monitor the events that occur during the execution of WPAR commands.

For this feature, a set of functions are implemented that log events and output messages to the screen. The functions will determine which language to use, and then open the appropriate message catalog for that language. These messages will be routed to stdout, stderr, or a log file, based on the message classification and user options.

The message classifications are described in 10.4.1, “Message classifications” on page 307.

## 10.4.1 Message classifications

Messaging output is controlled by setting the level of verbosity, as configured by the WPAR administrator. The messages are classified into seven categories. The default level for a standard production system will be INFO. The least verbose logging will be ERROR, and will increase through the levels WARNING, INFO, VERBOSE and finally DEBUG. Messaging can also be disabled entirely with a level of OFF.

When a user setting is not configured, it will be assumed that the desired message level is UNDEFINED. The UNDEFINED user settings will later be configured to their default settings.

A more detailed explanation of each category is provided here.

1. UNDEFINED

   This value is reserved for use by the messaging functions. Configuration parameters (such as command-specific message levels) and the environment variables are classified as UNDEFINED until they are specified or defaulted. Messages themselves will never be classified as UNDEFINED.

2. OFF

   No messages are output at this level. This classification can be specified by a user to disable messaging or logging. Messages will never be classified as OFF.

3. ERROR

   This class will be output to stderr. An ERROR message is logged when either:

   – A run-time error occurs and requires user interaction to continue

   – A run-time error occurs that immediately or eventually results in program termination (possibly after some cleanup routine)

   Messages at this level are assigned a 7-digit error identification number, for example 0960-001. The 0960 is the prefix belonging to WPAR messages.

The last three digits are from the ordering when the message catalog is indexed. Both sets of numbers are hardcoded into the message.

4. WARNING

   A WARNING message is logged when there has been, is, or will be:

   – An action performed that might not lead to the user's intended results
   – An action performed that might later result in an error

   This class is output to stderr. Like ERROR messages, WARNING messages are also assigned a 7-digit error identification number. This level is the default for the **wparexec** command.

5. INFO

   An INFO message logs events that:

   – Show the basic, expected status messages pertaining to the progression of the program
   – Explain if the program is waiting for a system or network resource (for example, waiting on a remote NFS file system to respond, so the command does not appear to be "hung")
   – Are understandable without advanced knowledge of WPAR internals.

   This class is output to stdout. Most of the WPAR commands will default to this level of output. The **wparexe**c command will default to WARNING.

6. VERBOSE

   A VERBOSE message is logged for actions performed that:

   – Show meaningful messages pertaining to the progression of the command, but are not normally necessary
   – Have a diagnostic value for an experienced administrator
   – Are understandable without advanced knowledge of WPAR internals

   This class is output to stderr. This messaging level will be displayed when the verbose flag (**-v**) is used, or when WPAR_SCREEN_LVL=VERBOSE.

7. DEBUG

   A DEBUG message is logged for actions performed that:

   – Would more than likely require knowledge of WPAR internals such as the design specifications or viewing source code
   – Would not likely be useful to anyone other than a developer
   – DEBUG messages will not be stored in or received from the message catalog.

This class is output to stderr.This messaging level will be displayed when WPAR_SCREEN_LVL=DEBUG, and will be logged (via a wpar_print_msg() call) if WPAR_LOGFILE_LVL = DEBUG.

Example 10-13 shows the screen output of a **startwpar** command with the DEBUG level. By default, the **startwpar** command logs the DEBUG level of information in the /var/adm/corrals/event.log file if you export WPAR_LOGFILE_LVL=DEBUG.

*Example 10-13 The DEBUG level output of the stopwpar command*

```
# export WPAR_SCREEN_LVL=DEBUG
# export WPAR_LOGFILE_LVL=DEBUG

# startwpar shaper
COMMAND START, ARGS: shaper.
Loading file: '/etc/wpars/shaper.cf'
Syntax checking 'general' stanza
        ...checking auto = no
              :
              :
Syntax checking 'network' stanza
        ...checking broadcast = 9.3.63.255
              :
Syntax checking 'security' stanza
        ...checking privs = PV_AU_,PV_AU_ADD,PV_AU_ADMIN
              :
              :
ckwpar: Passed.
Starting workload partition shaper.
Mounting all workload partition file systems.
Loading workload partition.
$corral_t = {
              'name' => 'shaper',
              'wlm_cpu' => [
                             undef,
                             undef,
                             undef,
                             undef
                           ],
              'path' => '/wpars/shaper',
              'hostname' => 'shaper',
              'vips4' => [
                           [
                             'en0',
                             '9.3.63.30',
                            '255.255.255.0',
                            '9.3.63.255'
                          ]
                        ],
               'wlm_procVirtMem' => [
                                    -1,
                                    undef
                                  ],
               'vips6' => [],
               'wlm_mem' => [
                              undef,
                              undef,
                              undef,
                              undef
                            ],
               'key' => 13,
               'wlm_rset' => undef,
               'opts' => 36,
               'id' => 0
             };
DEBUG: SUCCESS
        $!:
Exporting workload partition devices.
Starting workload partition subsystem cor_shaper.
0513-059 The cor_shaper Subsystem has been started. Subsystem PID is
565284.
Verifying workload partition startup.
Return Status = SUCCESS.
```

## 10.4.2 Environment variables

The user commands will be able to query an environment variable that indicates the desired level of logging or output to screen. The values will be named (presented in ascending order of verbosity):

OFF
ERROR
WARNING
INFO
VERBOSE
DEBUG

The values accepted by the environment variables are *not* case sensitive, and can be truncated to a first-letter abbreviation.

## WPAR_SCREEN_LVL

Default value: INFO

This environment variable indicates the maximum level at which messages are sent to the screen. ERROR, WARNING, VERBOSE and DEBUG messages will always be sent to stderr. INFO will always be sent to stdout.

The messages are only output to these file handles when the user has set the screen messaging level high enough to permit the message to be printed. For example, setting WPAR_SCREEN_LVL=ERROR outputs only ERROR messages, which will be sent to stderr.

Setting WPAR_SCREEN_LVL=VERBOSE outputs ERROR, WARNING, INFO and VERBOSE messages, where ERROR and WARNING messages are sent to stderr, and INFO and VERBOSE messages are sent to stdout.

Allowable values are OFF, ERROR, WARNING, INFO, VERBOSE, and DEBUG.

The Workload Partition commands will accept a **-v** flag for verbosity. This will set the level of output to the screen to VERBOSE. It is equivalent to setting the environment variable WPAR_SCREEN_LVL=VERBOSE.

Example 10-14 shows the screen output of creating an application WPAR by exporting the WPAR_SCREEN_LVL=VERBOSE.

*Example 10-14 Screen level output of application WPAR in VERBOSE mode*

```
# export WPAR_SCREEN_LVL=VERBOSE

# wparexec /usr/bin/sleep 500 &
[2]     356550
# wparexec: Verifying file systems...
Workload partition sleep2 created successfully.
COMMAND START, ARGS: sleep2.
ckwpar: Passed.
Starting workload partition sleep2.
Mounting all workload partition file systems.
Loading workload partition.

# lswpar sleep2
Name     State  Type  Hostname  Directory
-----------------------------------------------
sleep2 A      A     sleep2 /
```

If the WPAR_SCREEN_LVL environment variable happens to be already set to DEBUG, then the **-v** flag will not decrease the SCREEN_LVL setting to VERBOSE. The command line verbose flag has no effect on the output to the log file.

The verbose flag overrides the command-specific messaging level. In other words, passing the **-v** flag to a command such as **wparexec**, which has a default output level of WARNING, will still set the command's output level to VERBOSE, not to INFO.

Commands such as **syncwpar**, **startwpar**, **stopwpar**, **rmwpar**, **rebootwpar**, **mkwpar**, **chwpar** and **lswpar** will support verbose flag.

Example 10-15 shows the screen output of the **stopwpar** command with the verbose mode flag (**-v**).

*Example 10-15 Screen level output of stopwpar command in verbose mode*

```
# stopwpar -F -v shaper
Stopping workload partition shaper.
Stopping workload partition subsystem cor_shaper.
0513-044 The cor_shaper Subsystem was requested to stop.
Shutting down all workload partition processes.
WPAR='shaper' CID=3
ID=262144 KEY=0x4107001c UID=0 GID=9 RT=-1
ID=524289 KEY=0x620002db UID=0 GID=0 RT=-1
Unmounting all workload partition file systems.
Umounting /wpars/shaper/var.
Umounting /wpars/shaper/usr.
Umounting /wpars/shaper/tmp.
Umounting /wpars/shaper/shared3.
Umounting /wpars/shaper/proc.
Umounting /wpars/shaper/opt.
Umounting /wpars/shaper/home.
Umounting /wpars/shaper.
Return Status = SUCCESS.
```

## WPAR_LOGFILE_LVL

Default value: VERBOSE

This environment variable indicates the level of messaging output to the log file. The messages output will be limited to the level specified by the environment variable, and all other levels below.

Allowable values are OFF, ERROR, WARNING, INFO, VERBOSE, DEBUG.

This setting has no relation to the WPAR_SCREEN_LVL environment variables.

Note that even when WPAR_LOGFILE_LVL is set to OFF, some messages can still be logged because corlogevent() can be called directly. The log file will only display messages in English.

Example 10-16 gives the logging WARNING messages by setting WPAR_LOGFILE_LVL=WARNING.

*Example 10-16 Log WARNING messages to event log file*

```
# mkwpar -n test -c -F
**********************************************************************
ERROR
mkwpar: 0960-100 All writable file systems must be remote for
checkpointable workload partitions.
Found local file system /var.

ERROR
mkwpar: 0960-100 All writable file systems must be remote for
checkpointable workload partitions.
Found local file system /.

ERROR
mkwpar: 0960-100 All writable file systems must be remote for
checkpointable workload partitions.
Found local file system /tmp.

ERROR
mkwpar: 0960-100 All writable file systems must be remote for
checkpointable workload partitions.
Found local file system /home.
**********************************************************************
mkwpar: Creating file systems...
 /
   :
   :

The /var/adm/corrals/event.log contains the warning messages only.The
following lines logged in that file.

#cat /var/adm/corrals/event.log
E 2008-06-30 22:32:48 606324 mkwpar test
**********************************************************************
```

```
E 2008-06-30 22:32:48 606324 mkwpar test ERROR
E 2008-06-30 22:32:48 606324 mkwpar test
E 2008-06-30 22:32:48 606324 mkwpar test mkwpar: 0960-100 All writable
file systems must be remote for checkpointable workload
 partitions.
        Found local file system /var.
E 2008-06-30 22:32:48 606324 mkwpar test
E 2008-06-30 22:32:48 606324 mkwpar test ERROR
E 2008-06-30 22:32:48 606324 mkwpar test
E 2008-06-30 22:32:48 606324 mkwpar test mkwpar: 0960-100 All writable
file systems must be remote for checkpointable workload
 partitions.
        Found local file system /.
              :
Found local file system /tmp.
              :
Found local file system /home.
E 2008-06-30 22:32:48 606324 mkwpar test
W 2008-06-30 22:32:48 606324 mkwpar test
*********************************************************************
I 2008-06-30 22:37:16 643304 populate test Removing work directory
/tmp/.workdir.356462.643304_1
~
```

## 10.4.3 Application programming interfaces

The functions implemented for translation and output to the screen are listed as follows:

**wpar_msg_init()** This should be called shortly after a command begins execution. Parameters include the command-specific messaging level and a Boolean value that indicates whether the verbose flag was passed. The function internally queries the environment and then makes a determination for the appropriate messaging output.

For C and Perl, wpar_msg_init() opens the appropriate message catalog, and keeps it open for the duration of the programs execution.

In shell, an additional parameter specifies a string representing the command line flags. It is inserted as the first parameter. This is to maintain backward compatibility with the init_logevent() call in libcor_sh.

| | |
|---|---|
| **wpar_print_msg()** | This outputs a message to stderr, stdout, or a log file, based on the message classification and the user configuration. Messages output to the screen are retrieved from a message catalog if they are not DEBUG messages. Messages to the log file are output in English. |
| **wpar_print_dbg()** | This is a wrapper to wpar_print_msg(), which does not require the message level, a message set or message number. Because it calls wpar_print_msg() with a message level of DEBUG_MSG_LVL, it outputs the messages to stderr. Messages are not looked up. Corlogevent will be called if logging is enabled. Messages to the log file are output in English. |
| **wpar_lookup_msg()** | This retrieves a message from the message catalog. This is written for C and Perl only. The shell commands did not require a standalone lookup function. |
| **wpar_msg_term()** | This is called when a command exits. It closes the catalog. It is written for C and Perl only. The shell code does not maintain an opened catalog, so this has not been implemented for shell. |
| **corlogevent()** | The function corlogevent() will use for logging WPAR events. The corlogevent() function will receive some minor updates so that the log file has consistent information for each message output. The ability to call corlogevent() directly to be retained. This allows messages to be logged regardless of the user-specified settings. After the logging is enabled, wpar_print_msg() also calls this function to log the English version of the message to a logfile. |

## 10.5 XTISO mobility

The X/OPEN Transport Interface (XIT) over Sockets (xtiso) driver is a STREAM-based pseudo driver that provides a Transport Interface (TLI) to the socket-based protocols. The only supported use of xtiso is by the TLI and XTI libraries.

The original implementation for the checkpoint and restart requires that the applications using XTI/TLI are not running inside the WPAR. If an application using the XTI/TLI library were running inside the WPAR, then the checkpoint would fail, because the XTISO stream returns a negative ACK for the checkpoint query. NFS applications that use the XTI and TLI interface are not checkpointed and restarted. Instead, they are started again on the target system.

The new version of AIX has solved this issue by providing enhancements to the XTI/TLI library and to the XTISO and TIMOD module to be able to support checkpoint and restart functionality. When a checkpoint message is sent to the XTISO stream, it will save the control block information and messages in the queue.

When the restart message is sent to the XTISO stream, it will restore the messages and the control block properties as they existed during the checkpoint. Similarly, the XTI/TLI library will restart the system call that was interrupted during the checkpoint.

## 10.5.1 XTISO checkpoint and restart flow

The checkpoint and restart of an application that uses the XTI /TLI library will require following actions:

- Save all connection-related information such as socket and mbuf (Transport layer).
- Control information of the XTISO and TIMOD module.
- Restart the interrupted system call, for example getmsg() and putmsg().

A typical XTI/LTI application stream is shown in Figure 10-2 on page 317.

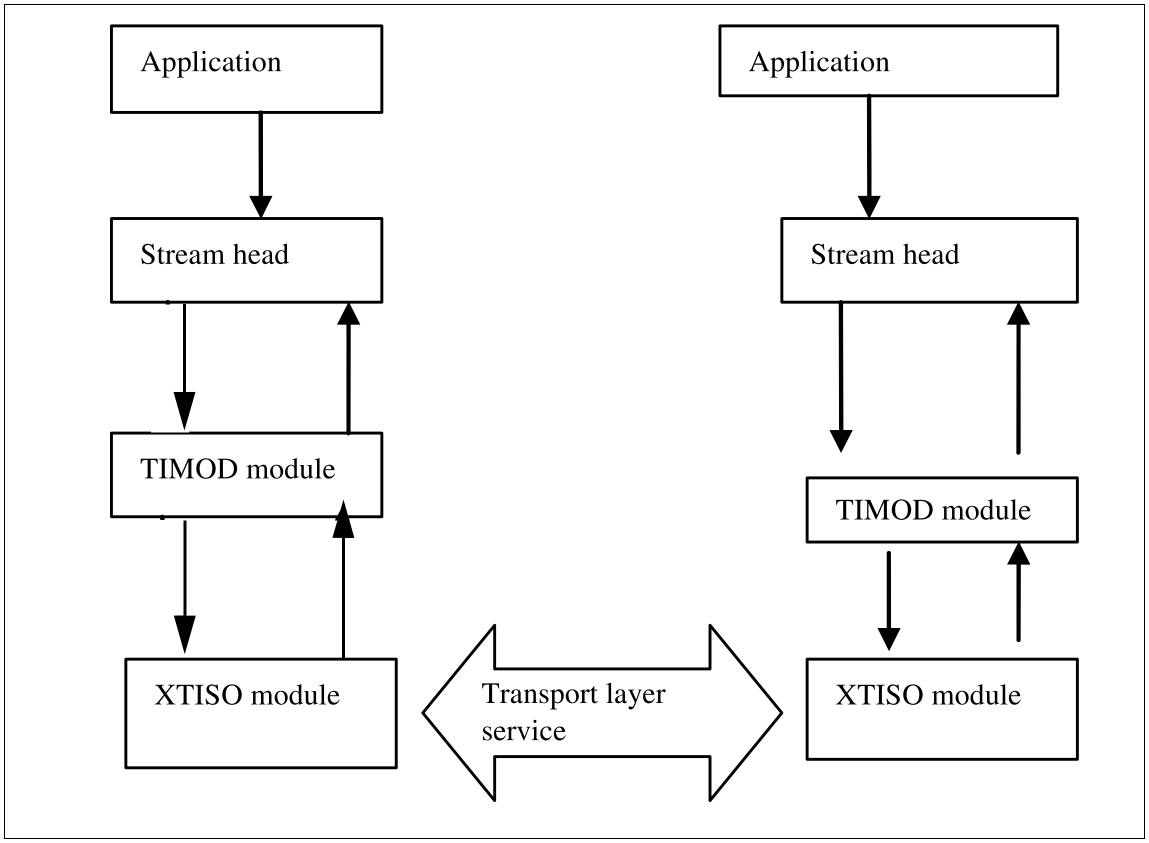


*Figure 10-2 A typical XTI/LTI application stream*

This typical XTI/TLI application stream contains a TIMOD module that is pushed when the stream is opened. The t_open() call pushes the TIMOD module.

When checkpoint occurs, the M_CKPT message is sent down the stream and each module puts its control block and the messages present on its queue in a buffer. Then it sends the message down. The last module puts in its data and sends the ACK back to the stream head.

During restart, the same data (checkpointed data) is passed down with the M_RESTART priority message. Each module finds its checkpointed data and sets up its control block and messages on the queue. The last module sends the ACK back to the stream head. Figure 10-2 illustrates the normal flow of events for checkpoint and restart.

The command flow for checkpoint and restart is shown in Figure 10-3 on page 318.

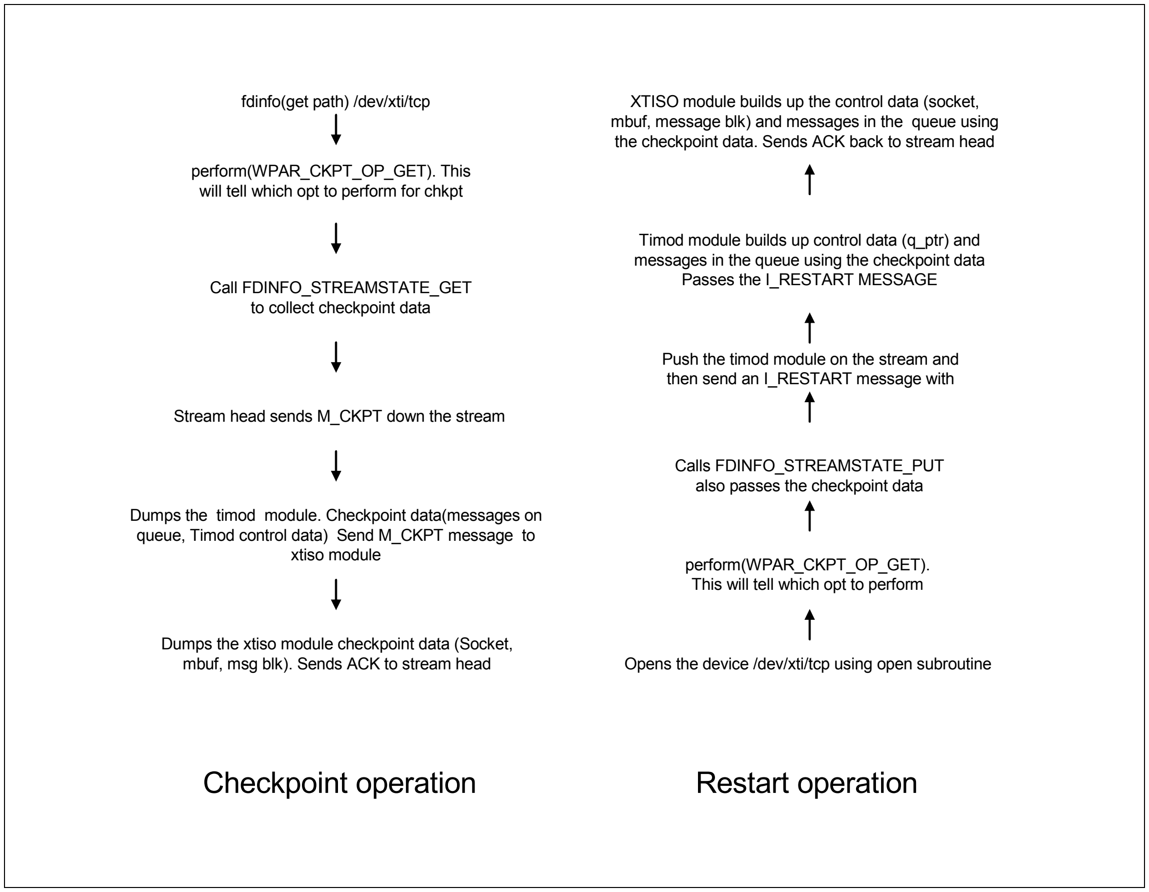

fdinfo(get path) /dev/xti/tcp
perform(WPAR_CKPT_OP_GET). This will tell which opt to perform for chkpt
Call FDINFO_STREAMSTATE_GET to collect checkpoint data
Stream head sends M_CKPT down the stream
Dumps the timod module. Checkpoint data(messages on queue, Timod control data) Send M_CKPT message to xtiso module
Dumps the xtiso module checkpoint data (Socket, mbuf, msg blk). Sends ACK to stream head
Checkpoint operation
XTISO module builds up the control data (socket, mbuf, message blk) and messages in the queue using the checkpoint data. Sends ACK back to stream head
Timod module builds up control data (q_ptr) and messages in the queue using the checkpoint data Passes the I_RESTART MESSAGE
Push the timod module on the stream and then send an I_RESTART message with
Calls FDINFO_STREAMSTATE_PUT also passes the checkpoint data
perform(WPAR_CKPT_OP_GET). This will tell which opt to perform
Opens the device /dev/xti/tcp using open subroutine
Restart operation

*Figure 10-3 Checkpoint and restart command flow*

### 10.5.2 XTISO library changes

Changes have been made to:

- t_bind()
- t_chk_ack()
- iostate_lookup()
- t_get_primitive()
- t_rcv()
- t_rcvv()
- t_rcvudata()

t_rcvvudata()

t_snd()

t_connect()

t_snddis ()

t_sndrel()

t_sndreldata()

t_sndv()

t_sndudata()

t_sndvudata()

### *Kernel space changes*

The kernel space will receive a checkpoint or restart message to initiate the checkpoint and restart. Therefore, support for new priority message types (M_CKPT, M_RESTART, M_CR_ACK) is provided.

Consult the technical reference documentation for more detailed information:

`http://publib.boulder.ibm.com/infocenter/systems/index.jsp?topic=/com.ibm.aix.basetechref/doc/basetrf1/About.htm`

## 10.6 Network File System client mobility support

Network File System (NFS) is the preferred file system for mobile WPARS. All applications within a mobile WPAR are supposed to access network files.

When a WPAR is checkpointed, the NFSv2/v3/v4 client subsystem is also checkpointed for migration. Later, when the WPAR is restarted on the local system or on a remote system, all NFS infrastructures have to be restored prior to the restart of the processes so that the applications can continue working seamlessly.

### *NFS checkpoint command line interface*

The NFS checkpoint command is called **`fschkpnt`**. It has the following interface:

`fschkpnt` <WPAR name> <checkpoint directory name>

### *NFS restart command line interface*

The NFS restart command is called **`fsrestrt`**. It has the following interface:

`fsrestrt` <WPAR name> <checkpoint directory name>

### *System call to sync an NFS based on its VFS number*

The system call has the following prototype:

int syncvfsnum (int vfs_number);

This call will return zero (0) on success and a non-zero value on failure (usually ETIMEDOUT or EINPROGRESS).

**Important:** Both application WPARs and system WPARs should be able to relocate when holding locks on NFS version 2, 3, or 4 files. After the WPARs are migrated the locks should be preserved, along with the internal mounts. The multiple mounts that may be present inside the WPARs should also be preserved.

Both application WPARs and application WPARs should be able to preserve the kerberoes-authenticated mounts after the WPARs are migrated.

Figure 10-4 on page 321 displays a mobile WPAR with NFS and kerberoes-authenticated mounted file systems. It also shows the locks owned by the process.

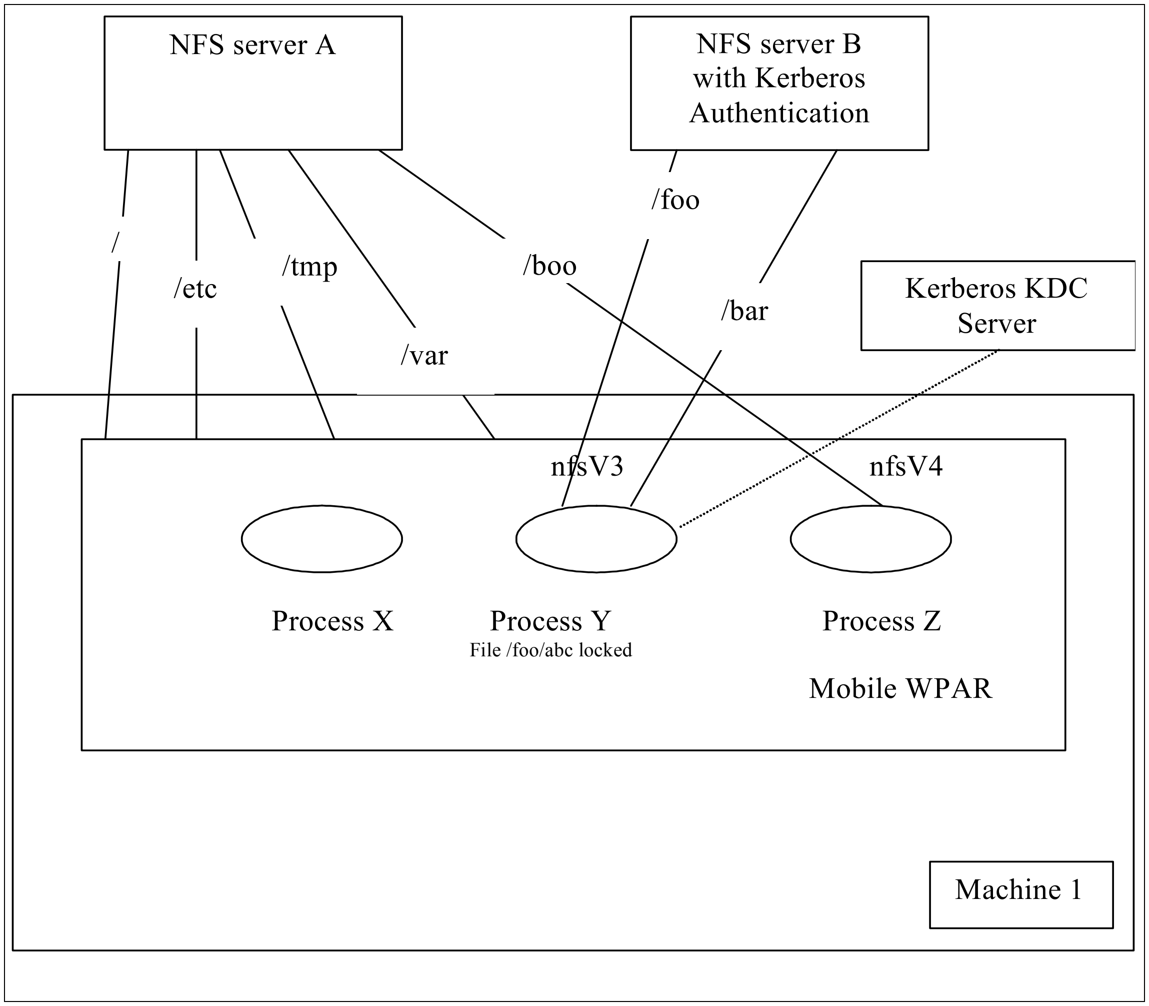


*Figure 10-4 WPAR before the checkpoint operation*

Figure 10-5 on page 322 displays the migrated WPAR with retaining locks by a process and with the preserved NFS and Kerberoes mounts.

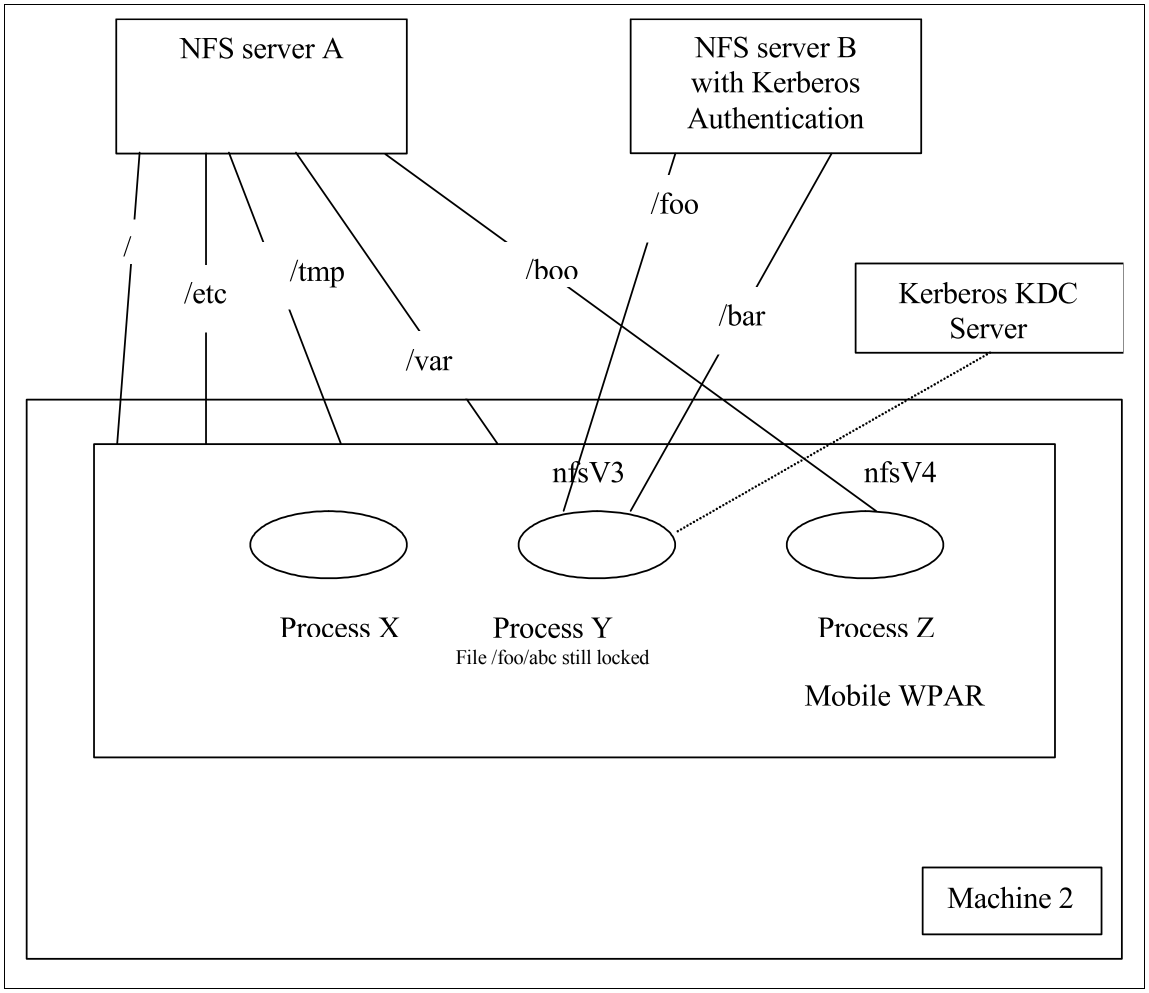


*Figure 10-5 WPAR after the restart operation*

In these figures, a WPAR is created with the file systems /, /tmp, /var, and /boo (NFS version 4) from NFS server A and the file systems /foo (NFS version 3) and /bar from the kerberoes server B. The process Y holds the lock on the nfsv3 mounted file system. After the WPAR is migrated, the lock should be retained and all internal mounts should be preserved.

## 10.7 System V and POSIX IPC in WPAR

System V and POSIX IPC objects are compartmentalized for each WPAR such that each WPAR might instantiate the same key and the same POSIX names, but have objects that are different for each WPAR. This is achieved by virtualizing the identifiers associated with the names and keys for each WPAR. Application programs are ensured, when they create or use System V or POSIX objects, that those objects pertain to the WPAR and are not visible or accessible from other WPARS.

When a WPAR undergoes a check and restart on the same or on a different WPAR, it is guaranteed that all the System V and POSIX IPC objects that the WPAR was using at the time of the checkpoint remain valid after the restart. All IPC objects of the WPAR will be in the same state when restarted as they were when the checkpoint was started.

The checkpoint and restart process of System V and POSIX IPC objects is transparent to the processes running the WPAR. No special programming is needed by applications running in the WPAR to ensure a correct checkpoint and restart of the System V and POSIX IPC. In fact, existing applications should be able to execute perfectly after undergoing a checkpoint and restart.

There is a very small performance penalty due to virtualization of System V and POSIX IPC identifiers. However, because System V and POSIX IPC are virtualized only when a checkpointable WPAR is created (using the **-c** option to the **mkwpar** command), this overhead is only associated with checkpointable WPARs.

# 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 326. Note that some of the documents referenced here may be available in softcopy only.

- *AIX 5L Workload Manager (WLM)*, SG24-5977
- *AIX V6 Advanced Security Features Introduction and Configuration*, SG24-7430
- *Introduction to Workload Partition Management in IBM AIX Version 6.1*, SG24-7431
- *PowerVM Live Partition Mobility on IBM System p*, SG24-7460
- *Hardware Management Console V7 Handbook*, SG24-7491
- *Virtualizing an Infrastructure with System p and Linux*, SG24-7499
- *IBM AIX Version 6.1 Differences Guide*, SG24-7559
- *PowerVM Virtualization on IBM System p: Managing and Monitoring*, SG24-7590
- *PowerVM Virtualization on IBM System p: Introduction and Configuration Fourth Edition*, SG24-7940

## Other publications

These publications are also relevant as further information sources:

- Thierry Fauck, *AIX 6 Workload Partitions and Live Application Mobility*, September 2007

  `http://www.ibm.com/developerworks/aix/library/au-wpar/`

- Brad Cobb, Lee Cheng *AIX: Bringing your applications into workload partitions*, November 2007

  http://www-304.ibm.com/jct09002c/partnerworld/wps/servlet/ContentHandler/whitepaper/aix/r6v1_wpar/introduction
- Clay Ryder, Sageza Group Inc. *The Value of PowerVM Workload Partitions: New Virtualization Options in IBM AIX v6.1*, February 2008

  http://www.sageza.com/available/Snapshot/StratSN%202-26-08%20IBM%20AIX%20WPAR.pdf

## Online resources

This Web site is also relevant as a further information source:

- IBM AIX Information Center: IBM Workload Partitions for AIX

  http://publib.boulder.ibm.com/infocenter/systems/scope/aix/index.jsp?topic=/com.ibm.aix.wpar/wpar-kickoff.htm

## 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

IBM

Redbooks

# Workload Partition Management in IBM AIX Version 6.1

(0.5” spine)
0.475”<->0.875”
250 <-> 459 pages

# Workload Partition Management in IBM AIX Version 6.1

**Presents updated technical planning information for AIX V6.1 TL2**

**Covers new partition mobility, isolation, NIM support, and WPAR Manager features**

**Provides walk-through examples for AIX system administrators**

Workload partition functionality, originally introduced in 2007 with the release of IBM AIX Version 6, is a strategic component of the IBM AIX Operating System. With the release of AIX V6.1 TL2 in November 2008, both the core functionality and features related to managing workload partitions have been improved and expanded.

This IBM Redbooks publication provides an updated introduction and “how to” guide for system administrators and architects using workload partitions in AIX V6.1 TL2. It builds on the original concepts and practices described in the first Redbooks publication about this topic, *Introduction to Workload Partition Management in IBM AIX Version 6.1*, SG24-7431, published in 2007.

In AIX 6.2 TL2, significant enhancements to core workload partition functions and new features have been added. Some of the important feature updates provide more flexibility and support for enhanced mobility, improved isolation, and NIM integration. A new and significantly updated version of IBM Workload Partitions Manager for AIX (WPAR Manager), the Web browser-based graphical user interface for managing and monitoring WPARs, is also available. WPAR Manager is a platform management solution that provides a centralized point of control for managing WPARs across a collection of managed systems running AIX.

SG24-7656-00 ISBN 0738432075