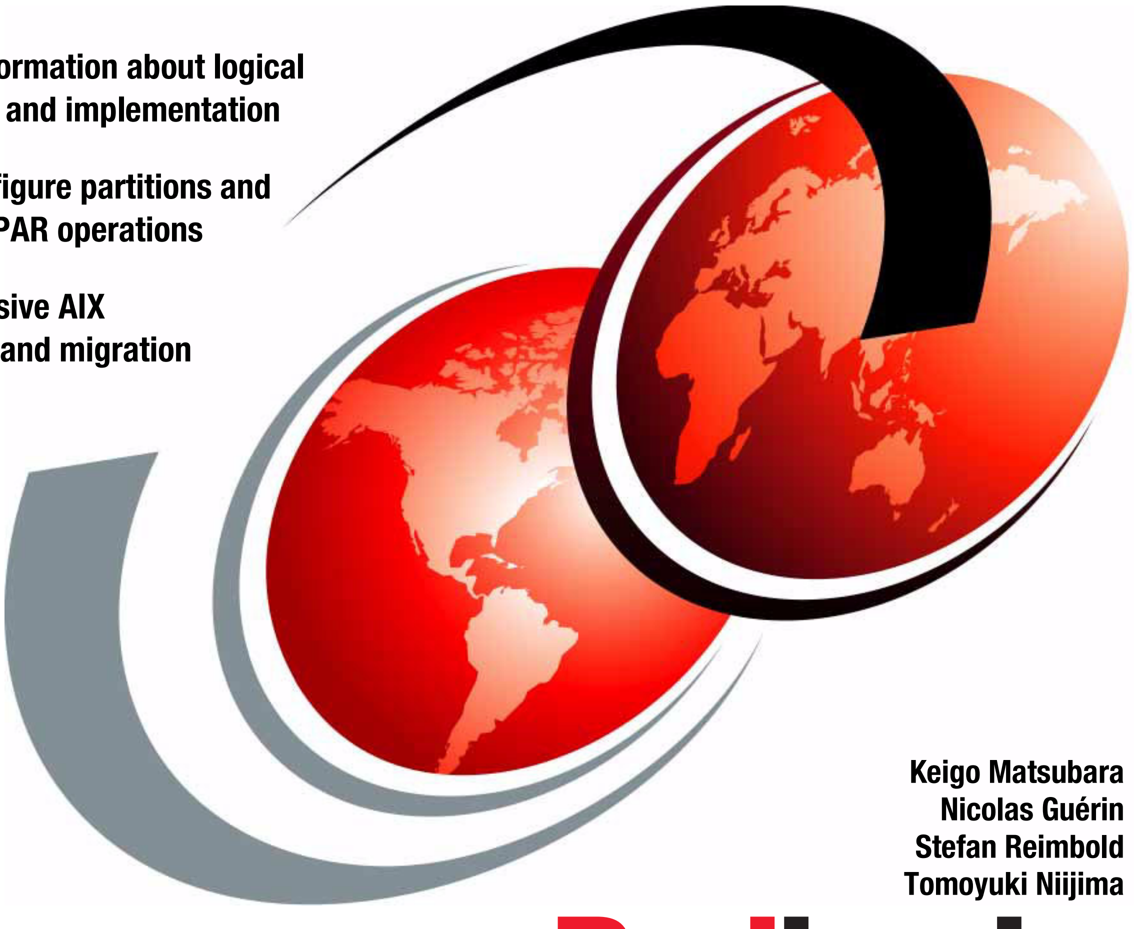
IBM

The Complete Partitioning Guide for IBM *e*server pSeries Servers

Detailed information about logical partitioning and implementation

How to configure partitions and manage DLPAR operations

Comprehensive AIX installation and migration tasks



Keigo Matsubara
Nicolas Guérin
Stefan Reimbold
Tomoyuki Niijima

Redbooks

ibm.com/redbooks

International Technical Support Organization

The Complete Partitioning Guide for IBM eserver pSeries Servers

October 2003

SG24-7039-01

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

Second Edition (October 2003)

This edition applies to currently available partitioning-capable IBM @server pSeries servers for use with AIX 5L Version 5.1 (product number 5765-E61) and AIX 5L Version 5.2 (product number 5765-E62).

This document created or updated on December 18, 2003.

Contents

Figures

Tables

Notices

Trademarks

Preface

This IBM Redbook provides a broad understanding of the logical partitioning on the IBM @server partitioning-capable pSeries® servers. This is the most outstanding feature of these servers, because it enables the servers to run multiple operating system instances concurrently on a single system. We focus on the following topics:

- Logical partitioning overview
- Partitioning implementation on pSeries servers
- Dynamic logical partitioning
- Creating and managing partitions
- Installing and migrating AIX in a partitioned environment

This redbook is a single-source handbook for IBM and IBM Business Partner technical specialists who support the partitioning-capable pSeries servers and for application developers who need to develop or modify DLPAR-aware applications.

The team that wrote this redbook

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

Keigo Matsubara is an advisory IT Specialist at the International Technical Support Organization (ITSO), Austin Center. Before joining the ITSO, he worked in the System and Web Solution Center in Japan as a Field Technical Support Specialist (FTSS) for pSeries. He has been working for IBM for 11 years.

Nicolas Guérin is an IT Specialist in IBM La Gaude, France. He has seven years of experience in the Information Technology field. His areas of expertise include AIX®, system performance tuning, HACMP, pSeries, SP, ESS, and SAN. He has been working for IBM for nine years.

Stefan Reimbold is a staff engineer at the IBM development lab in Boeblingen, Germany. He has seven years of experience in the UNIX® field. He has been working for IBM for two years. He holds a Ph.D. in Physics from the University of Heidelberg. His areas of expertise include IT-Security, RS/6000®, AIX, and AFS®.

Tomoyuki Niijima is an advisory IT architect in ERM service/BIS/IGS in IBM Japan. He has more than 10 years experience in AIX. This is his third redbook.

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

International Technical Support Organization, Austin Center
Scott Vetter, Vasfi Gucer

International Technical Support Organization, Poughkeepsie Center
Alfred Schwab, editor

IBM Austin
Andy McLaughlin, Bob Foster, Carolyn Scherrer, Dave Willoughby, John Bissell, Luke Browning, Mark D. Rogers, Randy Swanberg, Richard Cutler, Stephanie Jensen

Become a published author

Comments welcome

- Mail your comments to:

  IBM® Corporation, International Technical Support Organization
  Dept. JN9B Building 003 Internal Zip 2834
  11400 Burnet Road
  Austin, Texas 78758-3493

Summary of changes

This section describes the technical changes made in this edition of the book and in previous editions. This edition may also include minor corrections and editorial changes that are not identified.

Summary of Changes
for SG24-7039-01
for The Complete Partitioning Guide for IBM @server pSeries Servers
as created or updated on December 19, 2003.

October 2003, Second Edition

This revision reflects the addition, deletion, or modification of new and changed information described below.

New information

The following chapters are new:

- Chapter 10, “Dynamic reconfiguration using DLPAR scripts” on page 329
- Chapter 11, “Resource sets” on page 341
- Chapter 12, “Autonomic applications” on page 375
- Appendix D, “Using the Job Scheduling Console” on page 471
- Appendix E, “Advanced DLPAR script examples” on page 483
- Appendix F, “Autonomic application example” on page 489

Changed information

The following chapters were rewritten in order to cover new features, enhancements, and usage examples provided by the latest products:

- Chapter 4, “HMC graphical user interface” on page 129
- Chapter 9, “DLPAR operation using a command line interface” on page 277

Unchanged information

The following chapters and appendixes are unchanged but reviewed again:

- Chapter 1, “Logical partitioning overview” on page 3
- Chapter 2, “Partitioning implementation on pSeries servers” on page 17
- Chapter 3, “Dynamic logical partitioning” on page 53
- Chapter 5, “Basic managed system operation tasks” on page 151
- Chapter 6, “Creating and managing partitions” on page 165

- Chapter 7, “Installing and migrating AIX in a partitioned environment” on page 201
- Chapter 8, “DLPAR operation using graphical user interface” on page 255
- Appendix A, “Test environment” on page 389
- Appendix B, “Dynamic logical partitioning program templates” on page 407
- Appendix C, “Dynamic logical partitioning output samples” on page 449

January 2003, First Edition

The first version of this book was written by the following authors:

Keigo Matsubara, Akichika Ozeki, Erlander Lo, Deniz S. Erguvan, Jennifer Davis, Theeraphong Thitayanun, Viraf Patel

The following list shows contributors for the first version of this book:

IBM Austin

Andy McLaughlin, Ann Wigginton, Bob Foster, Bob Minns, Carolyn Scherrer, Christopher Chan, David Sheffield, Dave Willoughby, Duke Paulsen, Edward Shvartsman, Jane Chilton, John O'Quin, John Purcell, Julie Craft, Kanisha Patel, Larry Amy, Luke Browning, Mark Rogers, Michael Mall, Michael S. Williams, Minh Nguyen, Paul B. Finley, Randy Swanberg, Richard Cutler, Steven Molis, Susan Caunt, Trish Pierce, Truc Nguyen, and Walter Lipp

IBM Philadelphia

Rob Jackard

IBM Poughkeepsie

Michael Schmidt and Ron Goering

Part 1

Implementations

1

Logical partitioning overview

In this chapter, we first introduce necessary concepts and terminology to understand the logical partitioning implementation on the IBM @server partioning-capable pSeries servers. We discuss the following topics:

- Several partitioning implementations
- Partitioning support on pSeries servers
- Terminology used in partitioning
- Four terms regarding memory

As a system administrator who has responsibility for managing the partioning-capable pSeries servers, it is imperative that you become familiar with the aspects described in this chapter before you run the system in a logical partitioned environment.

1.1 Several partitioning implementations

There is a strong demand for high-end systems to provide greater flexibility, in particular the ability to subdivide them into smaller partitions that are capable of running a version of an operating system or a specific set of application workloads.

IBM initially started work on partitioning S/370™ mainframe systems in the 1970s. Since then, logical partitioning (LPAR) on IBM mainframes (now called IBM @server zSeries®) has evolved from a predominantly physical partitioning scheme, based on hardware boundaries, to one that allows for virtual and shared resources with dynamic load balancing. In 1999, IBM implemented LPAR support on the AS/400® (now called IBM @server iSeries™) platform. In 2000, IBM announced the ability to run the Linux operating system in an LPAR on a zSeries server.

Throughout this publication, we refer to the various partitioning mechanisms available on the market. Therefore, it is appropriate to clarify the following terms and definitions by which we classify these mechanisms:

Building block	A collection of system resources, such as CPUs, memory, and I/O connections. These may be physically packaged as a self-contained symmetric multiprocessing (SMP) system (rack-mounted or stand-alone) or as boards within a larger multiprocessor system. There is no requirement for the CPUs, memory, and I/O slots to occupy the same physical board within the system, although they often do. Other vendors use the terms system board, cell, and Quad Building Block (QBB) to refer to their building blocks.
Physical partition	One or more building blocks linked together by a high-speed interconnect. Generally, the interconnect is used to form a single, coherent memory address space. In a system that is only capable of physical partitioning, a partition is a group of one or more building blocks configured to support an operating system image. Other vendors may refer to physical partitions as *domains* or *nPartitions*.
Logical partition	A subset of logical resources that are capable of supporting an operating system. A logical partition consists of CPUs, memory, and I/O slots that are a subset of the pool of available resources within a system.

Note: The major difference between logical partitioning and physical partitioning is the granularity and flexibility available for allocating resources to an operating system image. Logical partitions have finer granularities than physical partitions.

It should be noted that the zSeries LPAR implementation is unique in comparison to the other partitioning implementations available from IBM and other hardware vendors. It is a mature and dynamic partitioning technology. The experience of IBM with physical and logical partitioning over the last 25 years has greatly influenced the design and implementation of logical partitioning on pSeries.

The pSeries 690 server is the first pSeries server to incorporate the ability of being partitioned. Its architectural design brings logical partitioning to the UNIX world, capable of multiple partitions inside a single server, with great flexibility in resource selection. The partitioning implementation on pSeries 690 differs from those of other UNIX system vendors in that the physical resources that can be assigned to a partition are not limited by internal physical system board boundaries. Now, IBM is expanding the partioning-capable pSeries server lineup, as explained in 1.2, “Partitioning support on pSeries servers” on page 5.

Processors, memory, and I/O slots can be allocated to any partition, regardless of their locality. For example, two processors on the same POWER4™ silicon chip can be in different partitions. Peripheral component interconnect (PCI) slots are assigned individually to partitions, and memory can be allocated in fixed-size increments. The fine granularity of the resources that can be assigned to partitions provides flexibility to create systems with the desired resources.

The partioning-capable pSeries servers are also capable of running both AIX 5L Version 5.1 or later and Linux inside a partition on the single system simultaneously.

Many of the features described in this document are operating system-dependant and may not be available on Linux. For more information, see:

`http://www.ibm.com/servers/eserver/pseries/linux/whitepapers/linux_pseries.html`

Note: Hereafter, we refer to logical partitions as partitions.

1.2 Partitioning support on pSeries servers

In addition to the first partioning-capable pSeries server, pSeries 690, IBM now is expanding the partioning-capable pSeries server lineup, as explained in this section.

1.2.1 Supported models

At the time of writing, the pSeries servers shown in Table 1-1 support partitioning.

Table 1-1 Supported partioning-capable pSeries servers

Official product model name	Short product name	M/T-MDL
IBM @server pSeries 690 Model 681	pSeries 690	7040-681
IBM @server pSeries 670 Model 671	pSeries 670	7040-671
IBM @server pSeries 655	pSeries 655	7039-651
IBM @server pSeries 650 Model 6M2	pSeries 650 Model 6M2	7038-6M2
IBM @server pSeries 630 Model 6C4	pSeries 630 Model 6C4	7028-6C4
IBM @server pSeries 630 Model 6E4	pSeries 630 Model 6E4	7028-6E4

Depending on the supported number of processors, the maximum number of partitions are shown in Table 1-2.

Table 1-2 Maximum number of processors, memory size, and partitions

Short product name	Maximum number of processors	Maximum memory size in GB	Maximum number of I/O drawers	Maximum number of partitions
pSeries 690	32[a]	512	8	16
pSeries 670	16	128	3	16
pSeries 655	8	32	1	2
pSeries 650 Model 6M2	8	64	8	8[b]
pSeries 630 Model 6C4	4	32	2	4[c]
pSeries 630 Model 6E4	4	32	0	2

a. The High Performance Computing (HPC) feature of pSeries 690 is equipped with up to 16 processors.
b. Needs external disk subsystems for the boot disk.
c. When equipped with I/O drawers.

Note: Hereafter, we use the short product names throughout this redbook.

The logical partitioning concept and required tasks are basically similar on these partioning-capable pSeries server models. However, assigning I/O resources to partitions depends on the models, and there are substantial differences. For the hardware model-specific information about the I/O resource assignments, see 2.3, “I/O device assignment considerations” on page 43.

Logical partitioning support on pSeries 630 and pSeries 650

Due to the I/O structure on these pSeries models, some resources may have to be assigned (and moved) as a group.

On the pSeries 650 systems, the internal disks and the CD/DVD devices that are on the same SCSI controller must be assigned together and moved together with dynamic logical partitioning (DLPAR) operations.

Note: Throughout this redbook, we use the term *CD/DVD devices* to refer to CD-ROM, DVD-RAM, and DVD-ROM devices.

On the pSeries 630 systems, PCI slot 1, PCI slot 2, internal Ethernet 2 (physical location code: U0.1-P1/E2), the internal SCSI disk subsystem (U0.1-P2/Z1), and ISA-based I/O (serial, keyboard, mouse) must be assigned to a single partition as a group and cannot be assigned separately to different partitions. These resources cannot be moved with DLPAR operations because they share a PCI host bridge with the Industry Standard Architecture (ISA) subsystem.

For further information, see 2.3, “I/O device assignment considerations” on page 43.

Note: The parallel ports on pSeries 630 are not supported in a partitioned environment.

1.2.2 IBM Hardware Management Console for pSeries (HMC)

In order to configure and administer a partioning-capable pSeries server, you must attach at least one IBM Hardware Management Console for pSeries (HMC) to the system. Depending on the partioning-capable pSeries server models, the HMC can be ordered as a feature code or a separate orderable product, as shown in Table 1-3.

Table 1-3 Required Hardware Management Console feature code or M/T-MDL

Short product name	**HMC FC or M/T-MDL**	**Note**
pSeries 690	FC 7316	1
pSeries 670	FC 7316	1
pSeries 655	M/T-MDL 7315-C01	1
pSeries 650 Model 6M2	M/T-MDL 7315-C01	2
pSeries 630 Model 6C4	M/T-MDL 7315-C01	2
pSeries 630 Model 6E4	M/T-MDL 7315-C01	2

1. The HMC is required regardless of whether the system is partitioned or running in the Full System Partition.
2. The HMC is required if the system is partitioned. If the system is running in the Full System Partition, the HMC is not required.

The HMC is a dedicated desktop workstation that provides a graphical user interface for configuring and operating pSeries servers functioning either non-partitioned or in the Full System Partition. It is configured with a set of hardware management applications for configuring and partitioning the server. One HMC is capable of controlling multiple pSeries servers. At the time of writing, a maximum of 16 non-clustered pSeries servers and a maximum of 64 partitions are supported by one HMC.

The HMC provides two native serial ports. One serial port should be used to attach a modem for the Service Agent. The second port can be used to attach a server. If multiple servers are attached to the HMC, additional serial ports are necessary. The ports can be provided by adding a maximum of two of the following features to the HMC:

- 8-Port Async Adapter (FC 2943)
- 128-Port Async Controller (FC 2944)

Note: To ensure that the Async adapter is installed in the HMC and not in the server, make sure that the adapter is configured as a feature of the HMC at the time of order.

The HMC is connected with special attachment cables to the HMC ports of the partioning-capable pSeries server models. Only one serial connection to a server is necessary despite the number of partitions.

The following cables are available:

- Attachment Cable, HMC to host, 15 meters (FC 8121)
- Attachment Cable, HMC to host, 6 meters (FC 8120)

With these cables, the maximum length from any server to the HMC is 15 meters.

To extend this distance, a number of possibilities are available:

- Another HMC can be used for remote access. This remote HMC must have a network connection to the HMC, which is connected to the servers.
- AIX 5L™ Web-based System Manager client can be used to connect to the HMC over the network, or the Web-based System Manager PC client can be used, which runs on a Windows® or Linux operating system-based system.

- When a 128-Port Async Controller is used, the RS-422 cables connect to a remote asynchronous node (RAN) breakout box, which can be up to 330 meters. The breakout box is connected to the HMC port on the server using the attachment cable. When the 15-meter cable is used, the maximum distance the HMC can be is 345 meters, providing the entire cable length can be used.

The HMC provides a set of functions that are necessary to manage partition configurations by communicating with the service processor, as shown in Figure 1-1 on page 11. These functions include:

- Creating and storing partition profiles that define the processor, memory, and I/O resources allocated to an individual partition
- Starting, stopping, and resetting a system partition
- Booting a partition or system by selecting a profile
- Displaying system and partition status

  In a non-partitionable system, the LED codes are displayed in the operator panel. In a partitioned system, the operator panel shows the word `LPAR` instead of any partition LED codes. Therefore, all LED codes for system partitions are displayed over the HMC.
- Using a virtual console for each partition or controlled system

  With this feature, every partition can be accessed over the serial HMC connection to the server. This is a convenient feature when the partition is not reachable across the network.

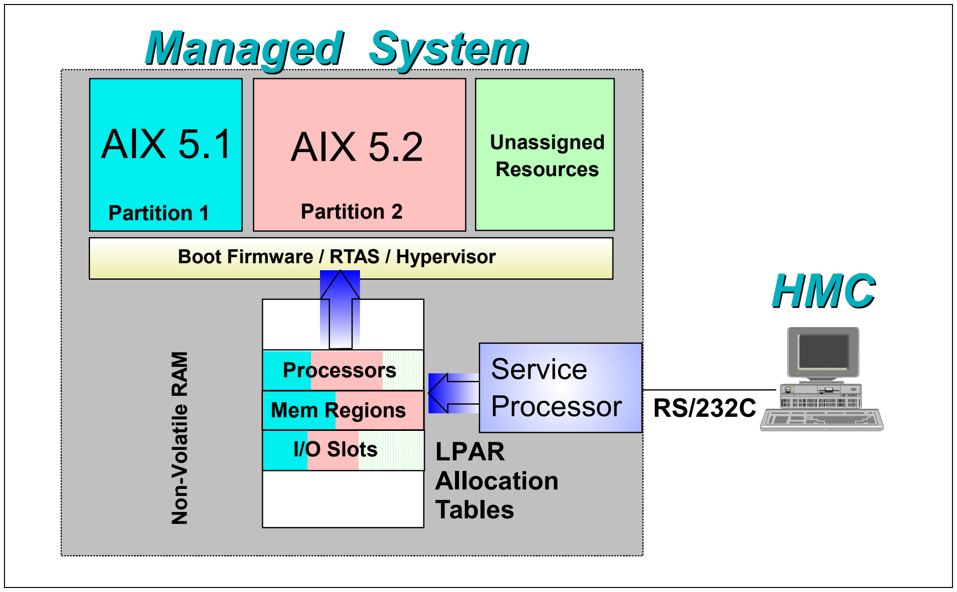


Figure 1-1 Communication between the HMC and the service processor

The HMC also provides a Service Focal Point for the systems it controls. It is connected to the service processor of the system using the dedicated serial link and must be connected to each partition using an Ethernet LAN for Service Focal Point and to coordinate dynamic logical partitioning operations. The HMC provides tools for problem determination and service support, such as call-home and error log notification through an analog phone line.

Note: A partioning-capable pSeries server managed by HMC is also referred to as a managed system.

1.3 Terminology used in partitioning

In this section, we explain the concepts and terminology used in logical partitioning on partioning-capable pSeries servers.

1.3.1 Logical partitioned environment

The partioning-capable pSeries servers support a logical partitioned environment that enables you to run multiple logical partitions concurrently. The maximum number of partitions that can concurrently run depends on the specific partitioning-capable pSeries server model. For example, both the pSeries 670 and pSeries 690 support up to 16 partitions running concurrently.

In a logical partition, an operating system instance runs with dedicated resources: processors, memory, and I/O slots. These resources are statically assigned to the logical partition. The total amount of assignable resources is limited by the physically installed resources in the system.

Because the first implementation of logical partitioning provided by AIX 5L Version 5.1 is static, you have to shut down every operating system instance in the targeted logical partitions to change between these two modes or to change the resource assignment of running logical partitions. Logical partitions that are not changed are unaffected.

With the release of AIX 5L Version 5.2, dynamic logical partitioning (also called DLPAR) extends the capabilities of static logical partitioning by allowing the dynamic reassignment of resources across partitions. Dynamic logical partitioning provides an improved solution by allowing users to dynamically move hardware resources between partitions without requiring a reboot of partitions.

All the partioning-capable pSeries servers also support a special type of logical partition, called the Full System Partition. In a Full System Partition, only one operating system instance runs in the system. This instance has access to all the resources installed in the system. In Full System Partition, a partioning-capable pSeries server acts as a conventional pSeries SMP server, except for the performance and enhanced reliability, availability, and serviceability (RAS) feature.

1.3.2 Partition isolation and security

From a functional point of view, applications are running inside partitions in the same way they run on a stand-alone pSeries machine. There are no issues when moving an application from a stand-alone server to a partition. The design of partioning-capable pSeries servers is such that one partition is isolated from software running in the other partitions, including protection against natural software defects and even deliberate software attempts to break the partition barriers. It has the following security features:

- Protection against inter-partition data access

  The design of partioning-capable pSeries servers prevents any data access between partitions, other than using regular networks. This isolates the partitions against unauthorized access across partition boundaries.

- Unexpected partition crash

  A software crash within a partition should not cause any disruption to the other partitions. Neither an application failure nor an operating system failure inside a partition interferes with the operation of other partitions.

- Denial of service across shared resources

  The design of partioning-capable pSeries servers prevents partitions from making extensive use of a shared resource so that other partitions using that resource become starved. This means that partitions sharing the same PCI bridge chips, for example, cannot occupy the bus indefinitely.

In this way, applications can be safely consolidated in partitions in a partioning-capable pSeries server without compromising overall system security.

1.4 Four terms regarding memory

Because the word memory is overused in various contexts, we have to provide precise definitions of the four terms regarding memory, that is, virtual, physical, real, and logical memory, before explaining the partitioning implementation details.

First, we introduce Figure 1-2 on page 14 to explain the relationship between virtual and physical memory.

The term *virtual memory* is used in many operating system environments to express the function that enables the operating system to act as if it were equipped with a larger memory size than it physically has.

For example, there are two user processes, 1 and 2, in Figure 1-2 on page 14. Because each process should be isolated from the other processes, each has its own virtual memory address range, called the *process address space*. Each process address space is classified into several memory chunks called *segments* (shown as hatched rectangles in the figure). Each segment is again divided into small size memory chunks, called *pages* (not shown in the figure). The page is the minimal allocation unit size of virtual memory. As shown in Figure 1-2 on page 14, the process address space is partially filled, but is mostly vacant space.

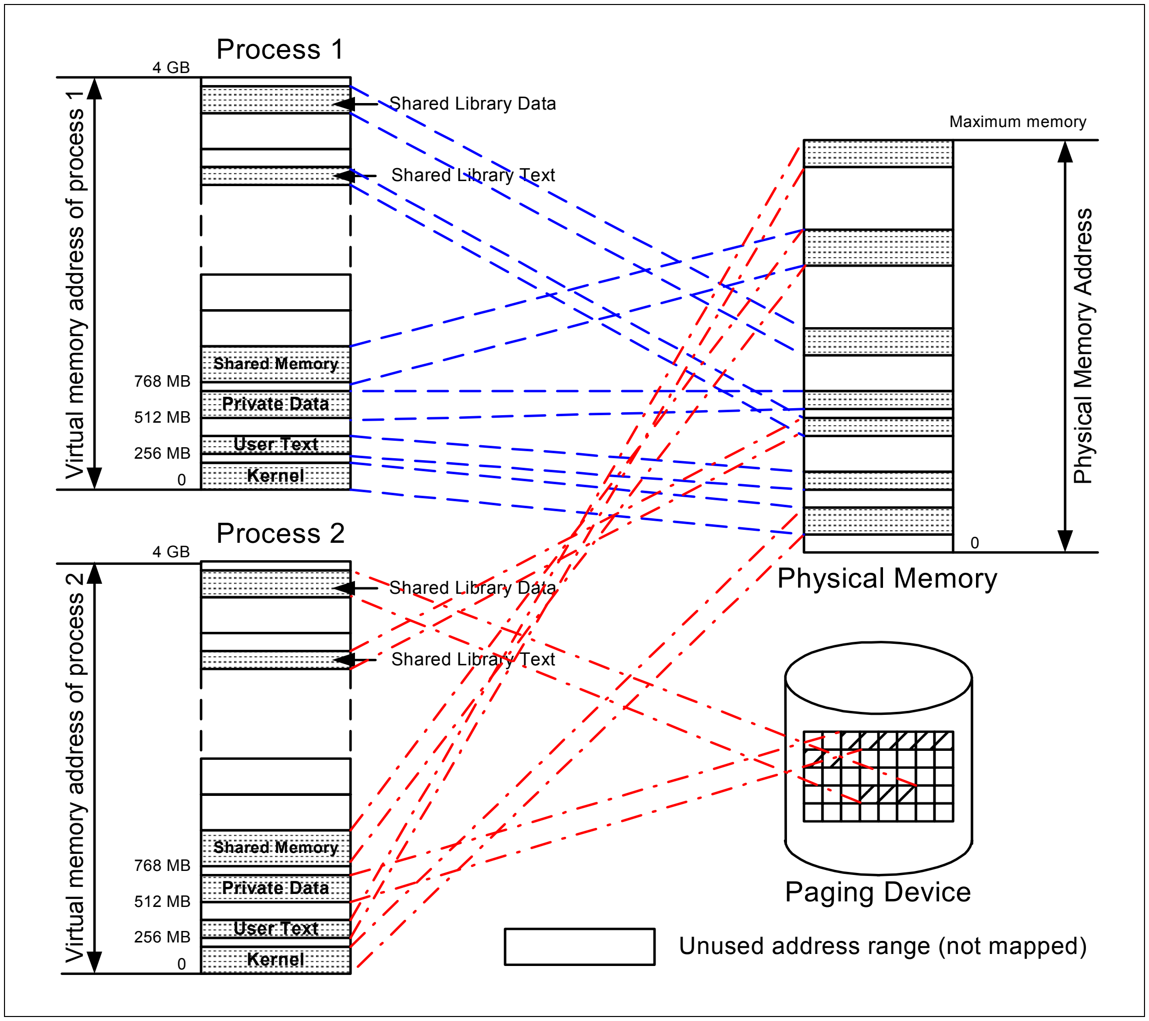


Figure 1-2   Virtual and physical memory relationship

The term *physical memory* is also used in many operating system environments to express the virtual memory function. Because not all of the virtual memory can sit in the physical memory in the system, only some portions of virtual memory are mapped to physical memory. The rest of the virtual memory is divided by page size, and each page can be mapped to a disk block in paging spaces, or still reside in a block of files in the file systems. This address translation is managed by the virtual memory manager (VMM) of the operating system using hardware components, such as the hardware page frame table (PFT) and translation look-aside buffer (TLB).

The term *real memory* is often used to represent the physical memory, especially when discussing the VMM functionality in the kernel. The modifier "real" comes from the real addressing mode defined in some processor architectures (including PowerPC®), where address translation is turned off. In a non-partitioned environment, because there is a one-to-one relationship between the real and physical memory, we can ignore the difference between these two terms in most cases.

The physical address space must encompass all addressable hardware components, such as memory cards, I/O ports, bus memory, and so on. Depending on the hardware implementation and restrictions, address ranges might need to be dispersed throughout the physical address space, which could result in a discontinuous physical memory address space. For example, if a PCI adapter device requires direct memory access (DMA), the device's DMA address is mapped on the specific physical memory address range by a PCI host bridge (PHB). Most VMMs of modern operating systems are designed to deal with non-contiguous physical memory addresses.

However, operating systems require a certain amount of contiguous physical memory that can be addressed translate-off, typically for bootstrapping, in a non-partitioned environment.

In a partitioned environment, real and physical memories have to be distinguished, because the physical memory address, which previously meant the real memory address, is no longer being used in that way because there is an extra level of addressing in a partitioned environment.

To support any operating system, including AIX and Linux, which require real mode code execution and the ability to present a real address space starting at 0 to each partition in the system, the *logical memory* concept is adopted. Logical memory is an abstract representation that provides a contiguous memory address to a partition (see Figure 1-3 on page 16). Multiple non-contiguous physical memory blocks are mapped to provide a contiguous logical memory address space. The logical address space provides the isolation and security of the partition operating system from direct access to physical memory, allowing the hypervisor (see "Hypervisor" on page 21) to police valid logical address ranges assigned to the partition. The contiguous nature of the logical address space is geared more to simplifying the hypervisor's per-partition policing than it is due to an operating system requirement. The operating system's VMM handles the logical memory as if it were physical memory in a non-partitioned environment.

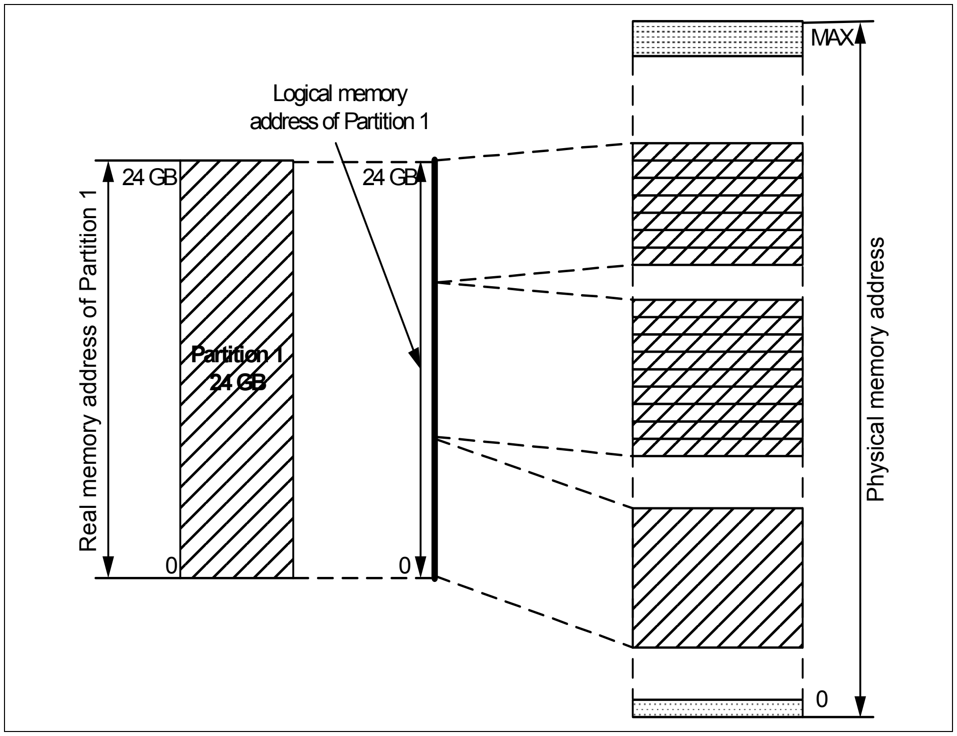

Logical memory address of Partition 1
24 GB
0
24 GB
0
Real memory address of Partition 1
Partition 1 24 GB
MAX
0
Physical memory address

Figure 1-3 Non-contiguous mapping

We also use the following terms throughout this redbook:

Physical memory block (PMB)	The physically contiguous memory block unit size of 256 MB manipulated inside the global firmware. A PMB ID is unique throughout the system.
Logical memory block (LMB)	The memory block unit size of 256 MB seen from the partitions. An LMB ID is unique inside a partition. An LMB is associated to a PMB. Therefore, a partition can have the same LMB IDs as other partitions. However, the same LMB IDs for different partitions are associated to different PMBs.

2

Partitioning implementation on pSeries servers

This chapter explains the partitioning implementation on pSeries servers in the following sections:

- Partitioning implementation
- Partition resources
- I/O device assignment considerations
- Service authority

2.1 Partitioning implementation

In a partitioned environment, each partition must have exclusive access to its assigned resources, but these resources also have to be strictly isolated from the other partitions. To implement these two demands, several system components have to work together to support the partitioning on partioning-capable pSeries servers. We categorize these components into the four groups explained in the following sections:

- 2.1.1, “Hardware” on page 18
- 2.1.2, “Firmware” on page 20
- 2.1.3, “Operating system: AIX 5L Version 5.1” on page 24
- 2.1.4, “Operating system: AIX 5L Version 5.2” on page 29

2.1.1 Hardware

To support partitioning on partioning-capable pSeries servers, in addition to the HMC explained in 1.2.2, “IBM Hardware Management Console for pSeries (HMC)” on page 8, the following hardware components have been newly developed or enhanced.

POWER4 processor

The POWER4 processor is a highly reliable and high-performance processor. Besides these characteristics, it also provides the following new functions to support a partitioned environment:

- Hypervisor call

  The POWER4 processor supports a special form of instructions. These instructions are exclusively used by a new controlling firmware named *hypervisor*. If an operating system instance in a partition requires access to hardware, it first invokes the hypervisor using *hypervisor calls*. Hypervisor allows privileged access to an operating system instance for dedicated hardware facilities and includes protection for those facilities in the processor. We discuss the hypervisor functions in 2.1.2, “Firmware” on page 20.

- Real mode offset (RMO) register

  The POWER4 processor supports a real mode offset register (RMO). By utilizing the RMO, the processor enables a mapping between the logical memory of a partition and the physical memory. A logical memory address of zero in a partition is registered at a fixed offset in the physical memory address space. The RMO is the physical address that corresponds to the beginning of a partition's memory. This is the partition's logical address 0. This address offset is set in the RMO register when this partition is activated. The processor adds the value stored in the RMO to each logical address fetch and store made by code running in a partition.

  The RMO address space only applies to Supervisor Real Mode Execution (MSR.DR/MSR.IR = 0). The RMO defines the low logical memory (starting at 0) that the operating system (Supervisor) can directly address in translation-off mode. Other logical address ranges (beyond the RMO) have no interaction or dependency on the real mode offset register, because they cannot be addressed in the translate-off mode.

- Real mode limit register (RML)

  The POWER4 processor supports a real mode limit register (RML). By utilizing the RML, the processor enables you to limit the range of real mode addressing. The RML is the size of a partition's memory region that is accessed in real mode.

For further information about the POWER4 processor, refer to the following publications:

- *IBM @server pSeries 670 and pSeries 690 System Handbook*, SG24-7040
- *POWER4 Processor Introduction and Tuning Guide*, SG24-7041

Interrupt controller

In a Full System Partition, the interrupt controller that manages the peripheral interrupt requests to the processors works in a fashion similar to other pSeries SMP servers. In a partitioned environment, the interrupt controller supports multiple global interrupt queues, which can be individually programmed to send external interrupts only to the set of processors allocated to a specific partition. Therefore, the processors in a partition can only receive interrupt requests from devices inside their partition.

PCI host bridges

In a Full System Partition, the PCI host bridges (PHBs) control the PCI slots in the I/O drawers, as in conventional pSeries servers. The PHBs use translation control entry (TCE) tables for the I/O address to memory address translation in order to perform direct memory access (DMA) transfers between memory and PCI adapters. The TCE tables are allocated in the physical memory.

In a partitioned environment, the hypervisor controls the DMA addressing to the partition memory for all I/O devices in all partitions. The hypervisor uses central TCE tables for all I/O devices, which are located outside of the memory of the partitions (see Table 2-2 on page 38). The hypervisor can manage as many TCE tables as it needs to. For example, each PCI host bridge could have its own TCE table. The number of TCEs needed, and thus the number of TCE tables, is a function of the number of PCI host bridges and slots. The address mapping is protected on a per-adapter basis. The PCI host bridges used in partioning-capable pSeries servers support the control of the PCI adapter DMA by the hypervisor.

The key point is that a logical partition is only given a *window* of TCEs per I/O slot necessary to establish DMA mappings for the device in that slot. The hypervisor polices TCE accesses by the operating system to ensure they are to a window owned by the partition.

Error handling

Various system components have the ability to limit the impact of hardware errors on a single partition. Generally, this is achieved by turning most hardware error reporting into bad data packets that flow back to the requesting processor. In many cases, this causes a machine check interrupt, which may or may not be recoverable within the partition. However, no other partitions are affected.

The enhancement includes the enhanced error handling (EEH) on the PCI bus. For further explanation about EEH, refer to *IBM @server pSeries 670 and pSeries 690 System Handbook*, SG24-7040.

Service processor

All the partioning-capable pSeries server models have an enhanced service processor (compared to existing pSeries models). The major enhancement of the service processor is a communication function with the IBM Hardware Management Console for pSeries (see Figure 1-1 on page 11).

2.1.2 Firmware

Support of partitioning on partioning-capable pSeries servers requires a new firmware named hypervisor, partition Open Firmware, and Run-Time Abstraction Service (RTAS).

Hypervisor

The hypervisor firmware provides major additions to firmware functionality. It implements the following three major categories of service calls:

- Virtual memory management

  Hypervisor becomes the only function that can update the address translation page tables in memory or the TCEs of the PHBs. In this way, hypervisor controls the physical memory locations that can be accessed from within a partition.

- Debug register and memory access

  For the debug and dump environments, hypervisor provides controlled access to protected facilities and memory locations.

- Virtual TTY support

  Hypervisor provides input/output streams for a virtual TTY device that can be used on the HMC.

Hypervisor is a passive object loaded into the first PMB in a partitioned environment. It is loaded only when the system is running in a partitioned environment and does not reserve a processor resource for itself. Hypervisor only runs when a partition needs a service executed on its behalf, such as creating a page table entry.

Hypervisor can be thought of as a call-back library used as any partition requires. Care has been taken to minimize the number of instructions required to implement the call-backs, so in most cases, AIX performance is identical for AIX in a non-partitioned environment where call-backs are not made, versus AIX in a partitioned environment where call-backs are required.

Hypervisor resides outside of the partition system memory in the first PMB at the physical address zero. This first PMB is not usable by any of the partition operating systems in a partitioned environment.

Open Firmware

A partioning-capable pSeries server has one instance of Open Firmware both in the partitioned environment and the Full System Partition. Open Firmware has access to all devices and data in the system. Open Firmware is started when the system goes through a power-on reset. Open Firmware, which runs in addition to the hypervisor in a partitioned environment, runs in two modes: global and partition. Global and partition Open Firmware share the same firmware binary stored in the flash memory.

In a partitioned environment, the partition Open Firmware runs on top of the global Open Firmware instance. The partition Open Firmware is started when a partition is activated. Each partition has its own instance of the partition Open Firmware, and it has access to all the devices assigned to that partition, but has no access to devices outside the partition in which it runs. Partition firmware resides within the partition memory, but is replaced when AIX takes control; it is just needed for the time necessary to load AIX into the partition system memory. The global firmware resides with the hypervisor firmware in the first 256 MB of the physical memory.

The global Open Firmware includes the partition manager component, which is an application in the global Open Firmware that establishes partitions and their corresponding resources (such as CPU, memory, and I/O slots), which are defined in partition profiles. The partition manager manages the operational partitioning transactions. It responds to commands from the service processor external command interface that originate in the application running on the HMC.

The partition profiles are stored in and retrieved from nonvolatile random access memory (NVRAM) by system firmware upon requests sent from the HMC. After the profiles are set up, the system automatically returns to this configured state on a power-on, even if the HMC is unavailable. The NVRAM also provides separate address spaces, called *slots*, to store up to several partitions. These slots, or partition IDs, are numbered from 1 to the supported maximum partition number.

One non-virtual, hardware password (PAP) controls access to systems management services (SMS), per system, which is generally presented on the virtual terminal window. This is consistent with the system administrator having access to the HMC.

To confirm the current firmware level, you can use the **lscfg** command as follows:

```
# lscfg -vp | grep -p 'Platform Firmware:'
      Platform Firmware:
        ROM Level.(alterable).......RH020930
        Version.....................RS6K
        System Info Specific.(YL)...U1.18-P1-H2/Y1
      Physical Location: U1.18-P1-H2/Y1
```

This example shows firmware level RH020930.

Note: The firmware level shown in this example is taken from our system, which is using the internal firmware release.

Run-Time Abstraction Services (RTAS)

RTAS presents the same platform service calls (with a few exceptions) that are presented in a non-partitioned environment, but have some underlying implementation changes to properly handle multiple AIX images, including:

- RTAS calls are only serialized within a partition.

  In general, RTAS operations are restricted to only those resources dedicated to that partition, with an error code return for invalid requests.

- Multiple virtual operator panels for all partitions.

  The information provided by operator panels in a traditional pSeries server are represented on the HMC on a per-partition basis.

- Per-partition time-of-day clock values.

  Time-of-day (TOD) is virtualized for each partition (including Full System Partition). Each partition can set its own time and date using of the AIX **`date mmddhhMMYYyy`** command.

- Restricted access to the per-partition NVRAM areas.

  Each partition has its own segment of NVRAM for the storage of its configuration variables, including a unique boot list for every partition. There is also a unique segment of NVRAM for when the system is in Full System Partition, with its own boot list. For example, there are up to 16 partition boot lists and a 17th Full System Partition boot list on the pSeries 670 equipped with 7040-61D I/O drawers.

- Partition reset capabilities.

  Previous pSeries servers had a service processor-based serial port snoop function that allowed remote reset of an unresponsive AIX image. The service processor would snoop the serial port data stream (which is the AIX console), but when AIX is not longer reachable through the keystrokes, and when a certain special command sequence was seen, the service processor would reset the system.

  On partioning-capable pSeries servers, with the HMC, each partition has its own, very powerful, reset capabilities: A soft reset that causes the partition operating system to get a PowerPC reset interrupt, and a hard reset that is the equivalent of a virtual power off of a partition (see 6.5, “Reset the operating system in a partition” on page 198). For the hard reset, no matter how disabled the partition operating system is, the hard reset will bring all the processors out of the partition and back to the global firmware partition manager so that the partition is ready to be reactivated.

To confirm the current Open Firmware RTAS version, you can use the **lsattr** command as follows:

```
# lsattr -El sys0 | grep rtasversion
rtasversion  1             Open  Firmware  RTAS version             False
```

This example shows RTAS Version 1.

2.1.3 Operating system: AIX 5L Version 5.1

The operating system in a partition must use the hypervisor calls in place of direct access to the hardware and address mapping facilities in conventional pSeries machines. An operating system without partition-enabled functions does not work in a partition.

If you are going to use AIX as a partition-enabled operating system, you will need to use AIX 5L Version 5.1 with the 5100-01 Recommended Maintenance Level plus APAR IY39794. From the operational point of view, there are a few noticeable differences in AIX when running inside a partition, as shown in Figure 2-1 on page 25:

- There is no physical console on the partition, unless you assign it explicitly. The built-in native serial ports[1] on the partioning-capable pSeries server can be assigned together only to one partition at the same time. To provide an output for console messages, and also for diagnostic purposes, the firmware implements a virtual TTY, called virtual terminal (see “Virtual terminal device support” on page 26), that is seen by AIX as a standard TTY device. Its output is streamed to the HMC. The AIX diagnostics subsystem uses this virtual TTY as a system console.
- Certain platform operations are constrained in partitions. For example, in a non-partitioned environment, platform firmware updates can be performed from AIX by the root user. Because firmware updates may affect all partitions in a partitioned environment, the administrator has the ability to specify that a particular partition has this authority. Within that partition, firmware updates will work the same way they do for a non-partitioned environment. See 2.4, “Service authority” on page 51 for more information.

Besides these considerations, AIX runs inside a partition the same way it runs on a stand-alone pSeries server. No difference is observed either from the application or the administrator’s point of view.

Note: The 32-bit AIX kernel supports up to 96 GB of physical memory size and 32-way processors, regardless of whether it is in a partition or in a Full System Partition.

[1] The pSeries 655 has no built-in native serial port.

Partitions are very transparent to AIX applications. In fact, third-party applications only need to be certified for a level of AIX that runs in a partition, and not for the partitioned environment itself. In this way, a partition on a partioning-capable pSeries server can be viewed as just another pSeries hardware platform environment.

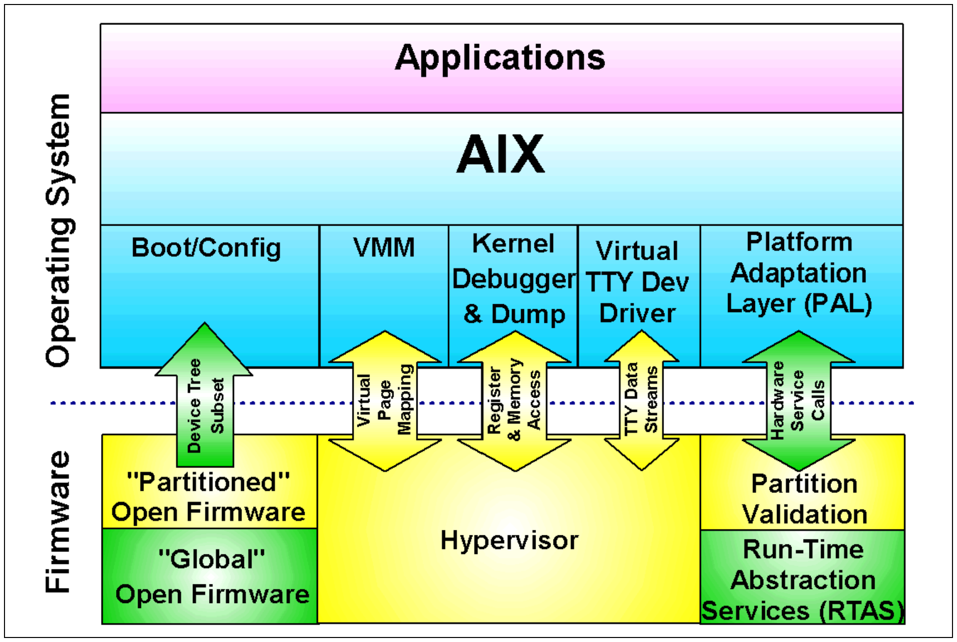


Figure 2-1 Interaction of AIX and firmware in a partition

Beginning with AIX 5L Version 5.1 with the 5100-01 Recommended Maintenance Level plus APAR IY39794, which is required for the support of partitioning on the pSeries 690, the following two filesets are provided:

```
# lslpp -qcl | grep LPAR | awk -F: '{print $2, "\t", $3, "\t", $7}' | uniq
devices.chrp_lpar.base.ras       5.1.0.15        CHRP LPAR RAS Support
devices.chrp_lpar.base.rte       5.1.0.15        Base CHRP LPAR Devices
```

These two filesets support the following functions, which are also shown in Figure 2-1:

- devices.chrp_lpar.base.ras

  Kernel debugger and dump function.

- devices.chrp_lpar.base.rte

  Virtual TTY device driver. We explain this fileset in the following section.

Please note that the Virtual Memory Manager (VMM) enhancement to support the partitioning is provided as an update to the AIX kernel fileset (bos.mp for the 32-bit SMP kernel and bos.mp64 for the 64-bit kernel); therefore, no separate fileset is provided for this function.

Virtual terminal device support

Example 2-1 shows the virtual terminal and its parent device. When configured in a partition, the virtual terminal is configured with a normal asynchronous device port tty0. The parent device sa0 is shown as `LPAR Virtual Serial Adapter`.

Example 2-1 Virtual terminal and its parent device

```
# lsdev -Cc tty
tty0 Available -00 Asynchronous Terminal
# lsdev -Cl tty0 -F parent
sa0
# lsdev -Cl sa0
sa0 Available  LPAR Virtual Serial Adapter
```

The virtual serial adapter sa0 is defined in the CuDv ODM class, as shown in Example 2-2.

Example 2-2 Customized device definition for the virtual serial adapter

```
# odmget -q name=sa0 CuDv

CuDv:
        name = "sa0"
        status = 1
        chgstatus = 1
        ddins = "vconsdd"
        location = ""
        parent = ""
        connwhere = ""
        PdDvLn = "adapter/lpar/vcon"
```

This definition is taken from the PdDv ODM class and then customized. The predefined device definitions for this device are shown in Example 2-3.

Example 2-3 Predefined device definition for the virtual serial adapter

```
# odmget PdDv | grep -i lpar
        subclass = "lpar"
        uniquetype = "adapter/lpar/vcon"
# odmget -q subclass=lpar PdDv

PdDv:
        type = "vcon"
        class = "adapter"
        subclass = "lpar"
        prefix = "sa"
        devid = ""
        base = 0
        has_vpd = 0
        detectable = 0
        chgstatus = 1
        bus_ext = 0
        fru = 0
        led = 1262
        setno = 158
        msgno = 1
        catalog = "devices.cat"
        DvDr = "vconsdd"
        Define = "/etc/methods/define -n"
        Configure = "/etc/methods/cfgvcon"
        Change = "/etc/methods/chgvcon"
        Unconfigure = "/etc/methods/ucfgvcon"
        Undefine = "/etc/methods/undefine"
        Start = ""
        Stop = ""
        inventory_only = 0
        uniquetype = "adapter/lpar/vcon"
```

The configuration method of this device is /etc/methods/cfgvcon, which is a symbolic link to /usr/lib/methods/cfgvcon. This file is included in the devices.chrp_lpar.base.rte fileset, as shown in Example 2-4. The fileset level has to be 5.1.0.15 or later.

Example 2-4 Device support fileset for virtual serial adapter

```
# ls -l /etc/methods/cfgvcon
-r-x------   1 root     system        9860 Aug 05 2001  /etc/methods/cfgvcon
# ls -ld /etc/methods
lrwxrwxrwx   1 root     system          16 Mar 26 10:08 /etc/methods ->
/usr/lib/methods
# ls -l /usr/lib/methods/cfgvcon
-r-x------   1 root     system        9860 Aug 05 2001
/usr/lib/methods/cfgvcon
# lslpp -w /usr/lib/methods/cfgvcon
  File                                      Fileset              Type
  ----------------------------------------------------------------------------
  /usr/lib/methods/cfgvcon
                                  devices.chrp_lpar.base.rte       File
# lslpp -L devices.chrp_lpar.base.rte
  Fileset                      Level  State  Type  Description (Uninstaller)
  ----------------------------------------------------------------------------
  devices.chrp_lpar.base.rte
                            5.1.0.15    C     F    Base CHRP LPAR Devices
```

For usage and functionality of the virtual terminal, see 4.5, “Virtual terminal window” on page 146.

Paging performance in a partitioned environment

An operating system in a partition has slightly degraded page table management performance, because it must use hypervisor services for page table management. An operating system in a Full System Partition has full use of all the system memory and *native* virtual memory management performance. In high volume paging environments, system performance is slightly less than native performance. In normal paging environments, there is no observable difference in performance.

Fast reboot in a partitioned environment

Rebooting an operating system instance in a partition is much faster than a full system reboot of a comparable conventional pSeries system because less hardware initialization is required.

Partition reboots are merely a re-establishment of the pSeries Open Firmware OS boot loader environment and, by nature, are very quick. A Full System Partition reboot repeats all the hardware initialization phases of the processors, caches, and memory. These phases are done by the service processor, and the I/O drawers and I/O adapters are done by the system firmware. When configuring all system resources in a single partition, hypervisor remains resident in memory. This enables the extremely rapid re-establishment of the boot environment, but requires the reservation of the first PMB by the hypervisor (see 2.2.4, “Reserved memory regions in a partitioned environment” on page 35).

Full System Partition reboots still have full system initialization phases and are almost comparable to traditional pSeries reboot times.

Affinity logical partitions

On the pSeries 670 and pSeries 690, another type of partition, called affinity logical partition, has been supported with the following configuration:

- AIX 5L Version 5.1 with 5100-02 Recommended Maintenance Level or later
- HMC software Release 2, Version 1 or later
- April 2002 system microcode update or later

Affinity logical partitions are useful for running CPU and memory-intensive applications, typically found in the High Performance Computing (HPC) area, on these server models. For further information about affinity logical partitions, see 6.2, “Affinity logical partitions” on page 183.

Large page support

Although AIX uses a fixed-size 4 KB page regardless of the release, an enhancement, called *large page*, was introduced in AIX 5L POWER for Version 5.1 with 5100-02 Recommended Maintenance Level. Exploiting the large page function, memory intensive applications, such as HPC and relational database management system (RDBMS) applications, can realize performance benefits, because less overhead is required by the hardware and VMM. For further information about large pages, refer to the *AIX Support for Large Pages* white paper found at:

http://www.ibm.com/servers/aix/whitepapers/large_page.html

2.1.4 Operating system: AIX 5L Version 5.2

In addition to all the enhancements and components provided in the former releases, AIX 5L Version 5.2 provides many enhancements, which are explained in *AIX 5L Differences Guide*, SG24-5765.

In this section, we briefly introduce some of enhancements that are covered in this redbook.

Dynamic logical partitioning

Starting with Version 5.2, AIX supports dynamic logical partitioning, which is a function to allow a partition whose resources are dynamically added and removed without requiring a partition reboot. Both 32- and 64-bit kernels running in a partition (except for the Full System Partition) support the dynamic logical partitioning function.

The support of dynamic logical partitioning also provides the following features:

- Dynamic processor deallocation with a minimum of two processors

  AIX supports the dynamic processor deallocation[2] function starting with Version 4.3.3, which can dynamically take a processor off-line when an internal threshold of recoverable errors on the processor is exceeded. Before AIX 5L Version 5.2, an SMP system or a partition had to have at least three processors, while an SMP system or a partition installed with AIX 5L Version 5.2 requires only two processors, in order to support dynamic processor deallocation.

- Capacity Upgrade on Demand (CUoD)

  CUoD is a feature that allows customers to dynamically activate pre-installed but inactive processors by purchasing the license key without requiring a system reboot.

  For further information about CUoD, refer to *IBM @server pSeries 670 and pSeries 690 System Handbook*, SG24-7040.

We provide a detailed explanation of dynamic logical partitioning in Chapter 3, “Dynamic logical partitioning” on page 53.

Partition on Demand

The system microcode Version 3.0 provides the ability to plug adapters into empty slots, define a new partition at the HMC, and activate and boot a previously nonexistent partition, while other partitions are running, without a power recycle of the system. This functionality, called *Partition on Demand*, obviously needs empty PCI slots.

Previously, you could not do this because the partition boot did not change the power state of PCI slots. However, you could power on PCI slots on existing partitions with AIX PCI Hot Plug, but not *spontaneously* create a new partition, including a new boot device, and activate that new partition with all the other partitions running. One limitation is that when you define your partition, you will not see the type of PCI cards (SCSI, network, and so on) present on the HMC, because the PCI slots are still powered off. You can also do a DLPAR I/O slot remove and then build a new partition.

[2] The dynamic processor deallocation function is also known as CPU Guard.

You can always power off the central electronics complex (CEC) or halt partitions, change I/O assignments, and restart, but this is quite severe on the running systems and not as common. With DLPAR in the system microcode Version 3.0 and AIX 5L Version 5.2, you can drop processors, memory, and I/O from an existing partition or set of partitions using DLPAR remove, and then spawn one or more new partitions that did not exist before. Spontaneous partitions up to a memory size of 16 GB are supported. With a memory size greater than 16 GB, two contiguous LMBs are required for the spontaneous partition's page table, and all the LMBs shed by the partition or partitions could all be discontiguous. Although it is possible to create a spontaneous partition with a memory size greater than 16 GB, Partition on Demand does not guarantee that it is always possible.

2.2 Partition resources

The logical partitioning function on partioning-capable pSeries servers allows you to assign processors, physical memory, and I/O devices to partitions. In the following section, we explain the rules of resource assignment.

2.2.1 Partition and system profiles

The information about resources assigned to a partition is stored in a partition *profile*. Each partition can have multiple partition profiles. By switching from one partition profile to another, you can change how resources are assigned. For example, you can assign relatively small resources to small online transactions on weekdays, and large resources to high-volume batch transactions on weekends.

In a static environment, to switch partition profiles, you have to shut down the operating system instance that is running in the partition and stop the partition (deactivate). You can also define a system profile (for administrative purposes) as an optional task. By using a system profile, you can power on multiple partitions in a specific order in one operation.

A description of the two profiles follows.

Partition profile

A partition profile stores the information about the assigned resources for a specific partition, such as processor, memory, and I/O devices. Each partition must have a unique name and at least one partition profile. A partition can have several partition profiles, but it reads only one partition profile when it is started (*activated*). You select a partition profile when you activate the partition;

otherwise, the default partition profile[3] is used. You can designate any partition profile as the default partition profile.

System profile

A system profile provides a collection of partition profiles that should be started at the same time. The partition profiles are activated in the order of the list defined in the system profile. You can also use system profiles to start the Full System Partition.

Both types of profiles are stored in the NVRAM of the partioning-capable pSeries server. Although you can create many partition profiles and system partition profiles, the actual number you can create depends on your profile configuration because both types of profiles share the same memory area in the NVRAM.

We summarize the relationship among partitions, partition profiles, and system profiles in Figure 2-2 on page 33. In this figure, partitions A, B, and C have three, one, and two partition profiles, respectively. Each partition has the default partition profile represented with a check mark. The system profile X is associated with partition profiles A1, B1, and C2; also, the system profile Y is associated with partition profiles A1 and C1. Keep in mind the following points:

- You do not have to associate all the partition profiles with system profiles. In this example, the partition profiles A2 and A3 are not associated with any system profile.
- It is possible to associate a partition profile to multiple system profiles. In this example, the partition profile A1 is associated with system profile X and Y.

[3] If you have only one partition profile for a partition, it is always the default partition profile.

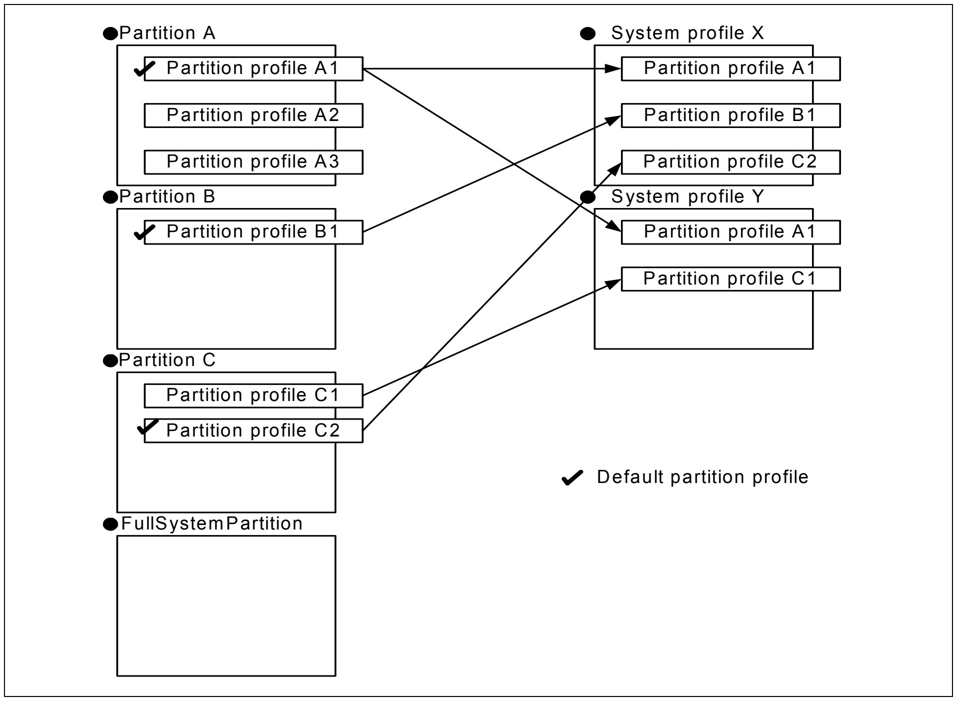


Figure 2-2 Partitions, partition profiles, and system profiles

To create partition profiles and system profiles, use the IBM Hardware Management Console (HMC) for pSeries, explained in 6.1, “Partition and system profile tasks” on page 166.

2.2.2 Three assignable resource types

A partition profile stores the information of the three types of resources: CPU, memory, and I/O slots. Partition profiles store the information of the specific PCI slots assigned in the I/O drawers where the I/O devices are possibly plugged in.

Note: Partition profiles do not store the information about specific I/O devices and PCI adapters.

Processors

Each installed and configured processor in the partioning-capable pSeries server can be assigned to a partition. You do not have to specify the precise location of the assigned processors in the partition profile, because the system selects the resources automatically. At least one processor must be assigned to each partition. Sharing processors between multiple active partitions is not possible.

Memory

In a partitioned environment, some of the physical memory areas are reserved by several system functions to enable partitioning in the partioning-capable pSeries server (see 2.2.4, “Reserved memory regions in a partitioned environment” on page 35). You can assign unused physical memory to a partition. You do not have to specify the precise address of the assigned physical memory in the partition profile, because the system selects the resources automatically.

The minimum amount of physical memory for each partition is 256 MB[4]. You can assign further physical memory to partitions in increments of 256 MB.

The AIX Virtual Memory Manager (VMM) manages the logical memory within a partition as it does the real memory in a stand-alone pSeries server. The hypervisor and the POWER4 processor manage access to the physical memory.

The physical memory allocation is not a simple concept. For further information about it, see 2.2.5, “Physical memory allocation to partitions” on page 38.

I/O slots

I/O devices are assignable to partitions on a PCI slot (physical PCI connector) basis. This means that it is not the PCI adapters in the PCI slots that are assigned as partition resources, but the PCI slots into which the PCI adapters are plugged.

To install an operating system, you have to assign at least one device adapter, typically an SCSI adapter, that is able to boot the operating system, and an adapter to access the install media (see “Boot devices” on page 45).

Once installed, you need at least one device adapter[5] connected to the boot disk or disks. For application use and system management purposes, you also have to assign at least one network adapter.

You can allocate slots in any I/O drawer on the system. We recommend that you assign more PCI slots than required for the number of adapters in the partition, even if these PCI slots are not populated with PCI adapters. This provides you with the flexibility to add PCI adapters into the empty slots of an active partition, using the PCI Hot Plug insertion/removal capability.

For detailed information about the I/O slots assignment, see 2.3, “I/O device assignment considerations” on page 43.

[4] On pSeries 670 and pSeries 690 with a firmware level earlier than system microcode Version 3.0, the minimum amount of physical memory for each partition is 1 GB.
[5] AIX installed in a partition can boot from SCSI, SSA, and Fibre Channel attached disks.

2.2.3 Three kinds of values for resource assignment

In a partition profile, you need to specify three kinds of values for each resource. For CPU and memory, specify *minimum*, *desired*, and *maximum* values. For I/O slots, specify the *required* and *desired* values.

If any of the three types of resources cannot satisfy the specified minimum and required values, the activation of a partition will fail. If the available resources satisfy all the minimum and required values, but do not satisfy desired values, the activated partition will get as many of the resources as are available.

The maximum value[6] is used to limit the maximum CPU and memory resources when dynamic logical partitioning operations are performed on the partition.

Note: If you are going to install operating systems that do not support dynamic logical partitioning, you should specify the same values for both the desired and maximum values.

For a detailed operation example for the resource assignment, see 6.1, “Partition and system profile tasks” on page 166.

2.2.4 Reserved memory regions in a partitioned environment

In a partitioned environment, some of the physical memory regions are reserved by several system functions to enable partitioning on partioning-capable pSeries servers. Before understanding the mapping between the logical memory address of a partition and the physical memory address, you have to consider the following memory regions:

- Hypervisor
- Partition page tables
- Translation control entry (TCE) tables

These three memory regions are not usable for the physical memory allocation of the partition.

Hypervisor

In a partitioned environment, hypervisor is loaded into the first PMB at the physical address zero and reserves the PMB. Although hypervisor does not occupy the whole memory block, this first PMB is reserved and cannot be used for any other purpose.

[6] The maximum value is not shown and is unavailable if the HMC software level is earlier than Release 3, Version 1.

Note: This memory region is not allocatable to any partitions.

Partition page tables

The AIX Virtual Memory Manager (VMM) uses hypervisor services to manage the partition page table. The AIX VMM communicates the desired virtual-to-logical mapping, and hypervisor translates that into the virtual-to-physical mapping within the page table.

The partition table resides outside of the physical address range mapped to the logical address of the partition. Partition page tables are additional memory that is required for a partition to operate, in addition to the total logical memory size of a partition. The partition page table size is determined to be four page table entries per 4096 bytes real page. Each page table entry has a size of 16 bytes. Therefore, the partition page table is an amount of contiguous physical memory blocks equal to 1/64 of the logical memory address range of the partition, rounded up to the nearest power of two, and it must be on an address alignment equal to its size. This relationship is shown in Table 2-1.

Table 2-1 Partition page table size

Partition memory size in GB	Partition page table size	Alignment	Number of PMBs
0.25	4 MB	4 MB	1
0.5	8 MB	8 MB	1
0.75 - 1	16 MB	16 MB	1
1.25 - 2	32 MB	32 MB	1
2.25 - 4	64 MB	64 MB	1
4.25 - 8	128 MB	128 MB	1
8.25 - 16	256 MB	256 MB	1
16.25 - 32	512 MB	512 MB	2
32.25 - 64	1 GB	1 GB	4
64.25 - 128	2 GB	2 GB	8
128.25 - 256	4 GB	4 GB	16
Note: Partition memory size less than 1 GB requires system microcode Version 3.0 or later on pSeries 670 and pSeries 690.			

For example, a 2.5 GB partition would have a page table space of 39 MB, but when rounded up, it would consume 64 MB of space, even if only 39 MB were used. The partition page table has to be allocated in contiguous physical memory, and the address alignment would have to be equal to its size. For example, the 64 MB page table needs to be aligned on an address that is an integer multiple of 64 MB.

Partition page tables are placed in the first available PMB that can house the table. If there is enough space to hold the partition page table in that PMB, it will be placed there. Otherwise, the next available PMB is used.

Note: This memory region is allocated to a partition, but is not usable by an operating system.

Translation control entry (TCE) tables

In a Full System Partition, TCE tables are controlled by the operating system, as in conventional pSeries systems. In a partitioned environment, the operating system uses hypervisor services to manage the TCE tables. The operating system communicates the desired I/O bus address to logical mapping, and the hypervisor translates that into the I/O bus address to physical mapping within the specific TCE table. The hypervisor needs a dedicated memory region for the TCE tables in order for the I/O address to partition memory address translation to perform direct memory access (DMA) transfers to PCI adapters. Each PCI slot that can be assigned to a partition is isolated underneath a PCI-to-PCI bridge. This PCI-to-PCI bridge is programmed with the *window* of allowable DMA addresses from this slot. This window corresponds to a window of TCEs allocated from the parent PCI host bridge (PHB) TCE table. Therefore, TCE tables can be shared across partitions when slots under the same PHB are assigned to different partitions.

An individual TCE table cannot be larger than 8 MB, which contains 2^{20} 8-byte entries capable of mapping 4 K entries each, which results in a 4 GB physical address range that can be mapped by a single TCE table. An individual PHB is programmed to index a single TCE table.

In a partitioned environment, TCE tables are allocated at the top of the physical memory and extend downward. The total size of TCE tables is based on the number of PHBs; therefore, it depends on the configured I/O on the server (see Table 2-2 on page 38).

Note: On the other partioning-capable pSeries server models, TCE tables always occupy one PMB located at the top of the physical memory.

Table 2-2 TCE table sizes

Server type	Number of SCM or MCMs	Total size of TCE tables in MB	Number of I/O features
p630	All configs	256	All configs
p650	1-4	256	1 primary with or without daughter card
p650	3-4	512	2 primary with 1 or 2 daughter cards
p690	1-4	256	1 I/O book or up to 2 in 2 MCM configuration
p690	3-4	512	2 I/O books
p690	3-4	768	3 I/O books
p690	4	1024	4 I/O books

Note: This memory region is not allocatable to any partitions.

The followin points summarize this information:

- The first PMB located at physical address 0 is always occupied by hypervisor.
- Because of the dynamic nature of partition tables, the actual allocation address for the page tables may vary.
- The number of configured I/O drawer enablement cards and the number of processors affects the total memory requirement.

2.2.5 Physical memory allocation to partitions

As explained in 1.4, “Four terms regarding memory” on page 13, a partition references contiguous logical memory, and the logical memory is mapped by multiple non-contiguous physical memory blocks. These physical memory blocks are composed of a *real mode offset* (RMO) region and possibly multiple *logical mode* regions, as shown in Figure 1-3 on page 16.

Besides the reserved memory regions described in 2.2.4, “Reserved memory regions in a partitioned environment” on page 35, the rest of the physical memory is available to be allocated to your partitions, as shown in Figure 2-3.

The RMO region is actually required by any operating system in the partition that has any real mode execution dependency on low memory, such as the PowerPC interrupt vector real memory addresses. The required RMO region size is where things become more operating system-specific. The VMM in AIX 5L Version 5.1 requires an RMO region large enough to hold all of its real mode data structures whose size scaled with the amount of total real memory.

In AIX 5L Version 5.2, this dependency is removed. AIX 5L Version 5.2 (as with other operating systems that implement virtual memory) only requires an RMO region large enough to satisfy the remaining real mode execution dependencies. The largest of these for AIX is now the boot strapping of the partition, because the AIX boot loader runs in real mode and manipulates the boot image contents (kernel, RAM file system, and ODM) in real mode, which must then all fit within the RMO region.

Mapping information between a logical and a physical memory address in a logical memory region is stored in a partition page table. The rest of the logical memory is mapped by possible multiple logical regions. Multiple logical mode regions do not have to be contiguous.

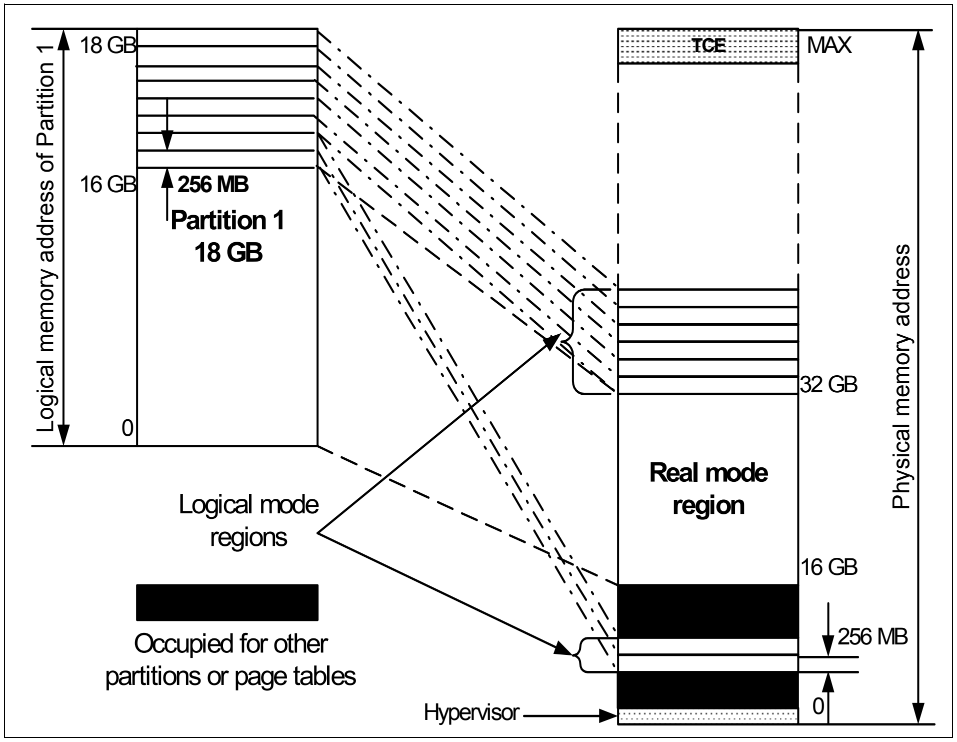


Figure 2-3 Real and logical mode regions

The following two algorithms are for the RMO region size allocation and alignment:

- Scaled RMO region size allocation
- Fixed RMO region size allocation

Note: The fixed RMO region size allocation algorithm is used only when:

- A partition is installed with AIX 5L Version 5.2 or later, or Linux.
- The partition is activated with the partition profile that is selected with the Small Real Mode Address Region check box (see Figure 6-5 on page 170).

If the partition does not satisfy this requirement, it always uses the scaled RMO region size allocation algorithm. The two algorithms can coexist on a partioning-capable pSeries server.

Scaled RMO region size allocation

The scaled RMO region size allocation algorithm uses different sizes depending on the maximum logical memory size of the partition, as shown in Table 2-3.

Table 2-3 Logical and associated real memory sizes in scaled RMO

Maximum logical memory size	**RMO region size**	**Assigned PMBs**
256 MB - 4 GB[a]	256 MB	1
4.25 GB - 16 GB	1 GB	4
16 GB - 256 GB	16 GB	64
More than 256 GB	256 GB	1024

a. The maximum partition size of less than 1 GB is only supported on the pSeries 670 and pSeries 690 with the post system microcode Version 3.0.

The following points summarize scaled RMO region size allocation:

- There is only one RMO region mapped to logical memory.
- An RMO region has to be contiguous.
- An RMO region is always mapped to logical memory, starting from address 0 of the logical memory address space.

- The RMO region size is 1 GB or 16 GB, as follows:
  - If the partition logical memory size is smaller than or equal to 16 GB, a 1 GB size RMO region is used.
  - If the partition logical memory size is greater than 16 GB, a 16 GB size RMO region is used.
- The RMO region must be aligned in the physical memory address space using the following rules:
  - If a 1 GB size RMO region is used, it has to be aligned at the address in multiplies of 1 GB (1 GB, 2 GB, 3 GB, and so on).
  - If a 16 GB size RMO region is used, it has to be aligned at the address in multiples of 16 GB (16 GB, 32 GB, 48 GB, and so on).

The RMO region is required by AIX VMM. AIX 5L Version 5.1 requires processor real mode memory relocation to be at least 1/16 of the total partition logical memory size as an RMO region. Upon activation of a partition, the offset starting address is set in the RMO register. The RMO region range is set in the RML register (see “POWER4 processor” on page 18). The RMO region must be one contiguous memory space, aligned on a 1 GB or 16 GB boundary.

Note: For partitions running AIX 5L Version 5.1, do not select the Small Real Mode Memory Region check box (see Figure 6-5 on page 170).

Fixed RMO region size allocation

The fixed RMO region size allocation algorithm uses a fixed size of 256 MB (a PMB) regardless of the maximum logical memory size. It can be aligned at any physical memory address in multiplies of 256 MB.

Summary

In Figure 2-3 on page 39, partition 1 installed with AIX 5L Version 5.1 is assigned 18 GB memory; therefore, the logical address ranges from 0 to 18 GB. A 16 GB RMO region is mapped at the logical address 0 from the physical address of 16 GB. The rest of logical memory (18 GB - 16 GB = 2 GB) is mapped by eight logical memory regions. These eight logical memory regions are scattered in several physical memory address ranges. Note that we assume that the physical memory areas shown as black rectangles are already occupied by other partitions or page tables. If these memory areas are not used before partition activation, multiple logical memory regions can be allocated contiguously.

We summarize the maximum possible number of partitions less or greater than 16 GB with the total memory size in Table 2-4 on page 42. It also provides the reserved and allocatable memory to partitions.

Table 2-4 Physical memory size and number of allocatable partitions

Total memory	**Approximate memory overhead**	**Approximate usable partition memory**	**Maximum number of partitions** ▶ **AIX 5L Versions 5.1, 5.2** ▶ **Pre-10/2002 firmware** **(Notes® 1, 2, and 3)**	**Maximum number of partitions** ▶ **AIX 5L Version 5.1** ▶ **Post-10/2002 firmware** **(Notes 1, 2, and 4)**	**Maximum number of partitions** ▶ **AIX5L Version 5.2** ▶ **Post-10/2002 firmware** **(Notes 2 and 5)**
4 GB	.75 - 1 GB	3 - 3.25 GB	2 and 0	13 and 0	13
8 GB	.75 - 1 GB	7 - 7.25 GB	6 and 0	16 and 0	16
16 GB	.75 - 1 GB	15 - 15.25 GB	14 and 0	16 and 0	16
24 GB	1 - 1.25 GB	22.75 - 23 GB	16 and 0	16 and 0	16
32 GB	1 - 1.25 GB	30.75 - 31 GB	16 and 0	16 and 0	16
40 GB	1.25 - 1.75	46.25 - 46.75 GB	16 and 1	16 and 1	16
48 GB	1.25 - 1.75 GB	46.25 - 46.75 GB	16 and 1	16 and 1	16
64 GB	1.5 - 2 GB	62 - 62.5 GB	16 and 2	16 and 2	16
80 GB	2G - 2.5 GB	93.5 - 94 GB	16 and 3	16 and 3	16
96 GB	2 - 2.5 GB	93.5 - 94 GB	16 and 4	16 and 4	16
128 GB	2.5 - 3.5 GB	124.5 - 125.5 GB	16 and 6	16 and 6	16
160 GB	3.5 - 4.5 GB	187.5 to 188.5 GB	16 and 8	16 and 8	16
192 GB	3.5 - 4.5 GB	187.5 to 188.5 GB	16 and 10	16 and 10	16
224 GB	5 - 6 GB	250 to 251 GB	16 and 12	16 and 12	16
256 GB	5 - 6 GB	250 to 251 GB	16 and 14	16 and 14	16

The following notes apply to Table 2-4:

1. In column 4 and 5, a difference is made between partitions with memory less or equal to 16 GB and greater than 16 GB, respectively.
2. All partition maximums are subject to availability of sufficient processor, memory, and I/O resources to support that number of partitions. For example, a system with only eight processors can only support a maximum of eight partitions.

3. These rules apply to systems running partitions with any version of AIX or Linux, if the firmware and HMC release levels are earlier than the 10/2002 release level.
4. These rules apply to systems running partitions with AIX 5L Version 5.1, if the firmware and HMC release levels are at the 10/2002 release level or later. The HMC partition profile option for Small Real Mode Address Region should not be selected for AIX 5L Version 5.1 partitions. These numbers reflect the maximum when running only AIX 5L Version 5.1 partitions, but AIX 5L Version 5.1 and AIX 5L Version 5.2 partitions can be mixed, and may allow for additional partitions to be run (up to the maximum of 16).
5. These rules apply to systems running partitions with AIX 5L Version 5.2 (or later) or Linux, if the firmware and HMC release levels are at the 10/2002 release level or later. The HMC partition profile option for the Small Real Mode Address Region check box should be selected for these partitions.

The following two points summarize the physical memory allocation:

- For systems with 16 GB or less of physical memory installed, this rule is valid:

  The maximum number of partitions = Total memory (in GB) - 2.

- You have to install more than 32 GB of physical memory on the partioning-capable pSeries server to activate AIX 5L Version 5.1 partitions greater than 16 GB. Because the 16 GB real mode region should be aligned on physical address 16 GB, 32 GB, 48 GB, and so on, in 32 GB memory configuration, the physical memory addresses 16 GB to 32 GB is partially used by TCE and cannot be used to allocate the 16 GB real mode region.

2.3 I/O device assignment considerations

Assignment of I/O slots to partitions is a relatively easy task once you understand the following considerations. The hardware architecture of each partioning-capable pSeries server model also influences some aspects of it.

For detailed information about each of the partioning-capable pSeries server models, refer to the following publications:

- pSeries 630 Model 6C4 and pSeries 630 Model 6E4

  pSeries630 Models 6C4 and 6E4 Technical Overview and Introduction

  Available at:

  `http://techsupport.services.ibm.com/server/library`

- pSeries 650 Model 6M2

  pSeries650 Model 6M2 Technical Overview and Introduction

Available at:

http://techsupport.services.ibm.com/server/library

- pSeries 670 and pSeries 690

  IBM @server pSeries 670 and pSeries 690 System Handbook, SG24-7040

2.3.1 Media devices

If your installation media is removable media (CD-ROM, DVD-RAM, 4 mm tape, and so on), the corresponding devices should be configured. However, the way of configuring removable media devices depends on the hardware architecture of partioning-capable pSeries servers as described here:

- pSeries 630 Model 6C4 and pSeries 630 Model 6E4

  An internal IDE CD-ROM (FC 2633) or DVD-ROM (FC 6634) device can be configured on these models. However, there are several device-reassignment operations required on the HMC in order to use these devices as the installation media device on these models if you run multiple partitions. You can also configure the following SCSI-attached removable media devices on these models[7]:

  – FC 2623: DVD-RAM drive (4.7 GB per surface)
  – FC 6120: 8 mm 80/160 GB tape drive
  – FC 6134: 8 mm 60/150 GB tape drive
  – FC 6158: 4 mm 20/40 GB tape drive

- pSeries 650 Model 6M2

  The pSeries 650 Model 6M2 supports the following SCSI-attached removable media devices:

  – FC 2635: 16/48X DVD-ROM auto-docking module
  – FC 2629: 4.7 GB R/W DVD-RAM auto-docking module
  – FC 2628: 40X CD-ROM auto-docking module
  – FC 6169: 8 mm 80/160 GB auto-docking module
  – FC 6131: 8 mm 60/150 GB auto-docking module
  – FC 6185: 4 mm 20/40 GB auto-docking module

  However, there are several device-reassignment operations required on the HMC in order to use these devices as the installation media device on the pSeries 650 Model 6M2. Because the SCSI controller connected to devices in the auto-docking media bays is shared with disk drives in the internal four hot-swappable disk drive bays, if a partition is using these disk drives as its boot device, you must shut down and power off that partition before reassigning the SCSI controller to another partition.

[7] These features require a single SCSI adapter, FC 6203 with 4260 (2-drop connector cable) inserted in a PCI slot within the machine.

- pSeries 655

  The pSeries 655 does not support any removable media devices. You can install AIX 5L Version 5.1 using PSSP[8] (requires control workstation) on this model.

- pSeries 670 and pSeries 690

  Configure devices in the media drawer and assign the SCSI adapter connected to the media drawer to the partition.

  In 7.2, “Installing AIX using removable media devices” on page 207, we provide detailed information about the installation from CD-ROM on the pSeries 670 and pSeries 690.

If your installation media is on the network, one of the following network adapters must be assigned to the partition:

- Ethernet
- Token ring

For detailed information about the installation from the network, see 7.3, “Installing AIX using Network Installation Manager (NIM)” on page 218.

2.3.2 Boot devices

Each partition requires its own separate boot device. Therefore, you must assign at least one boot device and a corresponding adapter per partition. The partioning-capable pSeries server models support boot devices connected with SCSI, SSA, and Fibre Channel adapters.

The following describes boot device considerations:

- pSeries 630 Model 6C4

  The pSeries 630 Model 6C4 can have up to four internal SCSI disk drives in the 4-pack disk bay. However, these disks are connected to one internal SCSI controller, so they only can be assigned to a partition. The other partitions must be assigned the boot adapter and disk drive from the following options:

  – A boot adapter inserted in one of four PCI-X slots in the system. A bootable external disk subsystem is connected to this adapter.

  – A bootable SCSI adapter (which can have various features) is inserted in the PCI-X slot 7 in a 7311-D20 I/O drawer connected to the system. The adapter is connected to one of 6-pack disk bays of drawer that houses the boot disk drive.

[8] Parallel System Support Program

  - A boot adapter inserted in one of seven PCI-X slots in a 7311-D20 I/O drawer connected to the system. A bootable external disk subsystem is connected to this adapter.

> **Note:** The pSeries 630 Model 6C4 supports up to two 7311-D20 I/O drawers.

- pSeries 630 Model 6E4

  The pSeries 630 Model 6C4 can have up to four internal SCSI disk drives in the 4-pack disk bay. However, these disks are connected to one internal SCSI controller, so they can only be assigned to a partition. The other partitions must be assigned the boot adapter and disk drive by the following:

  - A boot adapter inserted in one of four PCI-X slots in the system. A bootable external disk subsystem is connected to this adapter.

> **Note:** The pSeries 630 Model 6E4 does *not* support any I/O drawers.

- pSeries 650 Model 6M2

  The pSeries 650 Model 6M2 can have up to four internal SCSI disk drives in the 4-pack disk bay. However, these disks are connected to one internal SCSI controller, so they can only be assigned to a partition. The other partitions must be assigned the boot adapter and disk drive from the following options:

  - A boot adapter inserted in one of seven PCI-X slots in the system. A bootable external disk subsystem is connected to this adapter.
  - A boot adapter inserted in one of six PCI-X slots in a 7311-D10 I/O drawer connected to the system. A bootable external disk subsystem is connected to this adapter.

> **Note:** The pSeries 650 Model 6M2 supports up to eight 7311-D10 I/O drawers. The 7311-D10 I/O drawer does *not* have any internal disk drives. Therefore, bootable external disk subsystems are mandatory in order to configure and run multiple partitions on the pSeries 650 Model 6M2.

- pSeries 655

  The pSeries 655 can have up to two hot-swappable internal SCSI disk drives. However, these disks are connected to one internal SCSI controller, so they can only be assigned to a partition. Another partition must be assigned the boot adapter and disk drive from the following options:

  - A boot adapter inserted in one of three PCI slots in the system. A bootable external disk subsystem is connected to this adapter.

- An internal disk drive inserted in one of the 4-pack disk bays on a 7040-61D I/O drawer and the SCSI controller on the drawer.

  The 7040-61D I/O drawer can have up to 16 internal SCSI disk drives in the four 4-pack disk bays. Each of disk bays is connected to a separate internal SCSI controller on the drawer.
- A boot adapter inserted in one of 20 PCI slots in a 7040-D61 I/O drawer connected to the system. A bootable external disk subsystem is connected to this adapter.

Note: The pSeries 655 supports one 7040-61D I/O drawer.

- pSeries 670 and pSeries 690

  Partitions must be assigned the boot adapter and disk drive from the following options:

  - An internal disk drive inserted in one of the 4-pack disk bays on I/O drawer and the SCSI controller on the drawer.

    The 7040-61D I/O drawer can have up to 16 internal SCSI disk drives in the four 4-pack disk bays. Each of the disk bays is connected to a separate internal SCSI controller on the drawer.
  - A boot adapter inserted in one of 20 PCI slots in a 7040-61D I/O drawer connected to the system. A bootable external disk subsystem is connected to this adapter.

Note: The pSeries 670 supports up to three 7040-61D I/O drawers, and the pSeries 690 supports up to eight. The minimum hardware configurations of these models require at least one I/O drawer.

You should select the adapter of the boot device from the PCI slot of the system or the first I/O drawer (on pSeries 670 and pSeries 690) if the system is running in a Full System Partition, because the system can quickly find the boot device. In a partitioned environment, the placement of the boot adapter does not affect the speed of partitions' boot.

2.3.3 Network devices

It is mandatory to assign a network adapter to each partition. In addition to providing network access to client systems of a partition, the connection is also needed to provide the capability to manage the operating system and the applications in the partition remotely, either with a Telnet session or a graphical user interface, such as the Web-based System Manager. An Ethernet network connection between partitions and the HMC must be available if you want to use one of the following services:

- Service Agent
- Service Focal Point
- Inventory Scout
- Dynamic logical partitioning

These services communicate over the TCP/IP network between the partitions and the HMC.

2.3.4 Native Industry Standard Architecture (ISA) devices

All pSeries server models are equipped with several natively equipped ISA devices, such as native serial ports, a diskette drive, and keyboard and mouse ports, in order to support minimum hardware startup and diagnostics requirement. These ISA devices have the following characteristics when assigning resources to a partition:

- ISA devices are typically equipped on the system planner of the system (except for the pSeries 670 and pSeries 690) and they are connected by a single PCI bus through the PCI-ISA bridge chip. Therefore, these ISA devices can be assigned to only one partition as a group.
- ISA devices are not supported by dynamic logical partitioning.

The following list shows ISA devices on the partioning-capable pSeries servers:

- pSeries 630 Model 6C4 and pSeries 630 Model 6C4

  These models have the following ISA devices:

  – A diskette drive (if configured)
  – An IDE CD-ROM or DVD-ROM drive (if configured)
  – Three native serial ports (S1R, S2, and S3)
  – Keyboard and mouse ports

  Note: The parallel ports on these models are not supported in a partitioned environment.

- pSeries 650 Model 6M2

  The pSeries 650 Model 6M2 has the following ISA devices:

  – A diskette drive
  – Four native serial ports (S1, S2, S3, and S4)
  – Keyboard and mouse ports

- pSeries 655

  The pSeries 655 has no assignable ISA devices.

- pSeries 670 and pSeries 690

  These models have the following ISA devices:

  – Two native serial ports (S1 and S2) in the primary I/O book
  – A diskette drive in the media drawer

  These devices are shown as Group_XXX in the I/O selection when creating a partition profile (see Figure 5-4 on page 156 and the corresponding explanation).

2.3.5 Console devices

The HMC provides one virtual TTY console, called virtual terminal, for each partition, which removes most of the need for partition access to native serial ports. However, this virtual terminal was designed for limited purposes, such as installation of AIX and running diagnostics (see 4.5, “Virtual terminal window” on page 146 for detailed information about virtual terminal). If you need direct console access without using the network, the partition must be assgined a graphics console.

A graphics console is available on a partition by configuring the following features on the partition:

- A graphics adapter (FC 2848) with a graphics display
- A USB keyboard and mouse adapter (FC 2737) with a USB keyboard and a USB mouse attached.

Only one graphics console is supported per partition[9]. The graphics console is functional only when AIX is running. For any installation or service processor support functions, you have to use the virtual terminal function on the HMC.

In case of connecting serial devices (modems, serial printers, or terminal servers) to a partition as a serial console, you have to assign an 8-port (FC 2943) or 128-port (FC 2944) serial adapter to the partition.

2.3.6 High availability

You should place redundant devices of a partition in separate I/O drawers for highest availability. For example, if two Fibre Channel adapters support multipath I/O to one logical unit number (LUN), and if one path fails, another path using another adapter in another I/O drawer is automatically chosen by the device driver.

Some PCI adapters do not have enhanced error handling (EEH) capabilities built in to their device drivers. If these devices fail, the PCI host bridge (PHB) they are placed in and the other adapters in this PHB will be affected. Therefore, it is strongly recommended that you place all adapters without EEH capabilities on their own PHB, and not assign non-EEH adapters on the same PHB to different partitions.

For detailed information about EEH and supported PCI adapters on the pSeries servers, please refer to *PCI Adapter Placement References*, SA38-0538.

[9] Up to eight partitions can have graphics consoles on the pSeries 670 and pSeries 690.

2.4 Service authority

You can give one of the partitions in a partioning-capable pSeries server the *service authority* attribute. Service authority enables this partition, if running the AIX operating system, to perform system firmware updates or to set system policy parameters. Firmware updates also can be done from the service processor menus. Firmware updates are done at the system level, not on a per-partition basis.

A partition with service authority can perform firmware updates without having to power off the managed system. All other partitions must be shut down before the firmware update is initiated. The partition that has service authority must also have access to the firmware update image. If the firmware update image is provided on diskette, the diskette drive must belong to the partition that has service authority. If you are downloading the firmware update from the network, download it to the partition with service authority.

In the Full System Partition, you do not have to take additional steps to prepare for firmware updates.

See Figure 5-3 on page 155 and the related information for an explanation about how to set service authority on a partition.

3

Dynamic logical partitioning

Starting from AIX 5L Version 5.2, AIX supports dynamic logical partitioning (DLPAR). DLPAR is a function to allow a partition whose resources are dynamically added and removed without requiring a partition reboot.

The function enables the partioning-capable pSeries server models to be used in the strategic autonomic computing infrastructure by dynamically shifting resources among partitions on a single system.

This chapter contains the following sections:

- Dynamic logical partitioning overview
- The process flow of a DLPAR operation
- Internal activity in a DLPAR event
- DLPAR-safe and DLPAR-aware applications
- Integrating a DLPAR operation into the application
- Script-based DLPAR event handling
- DLPAR script subcommands
- How to manage DLPAR scripts
- API-based DLPAR event handling
- Error handling of DLPAR operations

3.1 Dynamic logical partitioning overview

DLPAR supports the following dynamic resource change in a partition without requiring a partition reboot:

- Resource addition
- Resource removal

By achieving the resource changes sequentially in the following order on two partitions in a system, the specified resource can be moved from a partition to another partition:

1. Resource removal from a partition
2. Resource addition to another partition

This resource movement is implemented as single task on the HMC although it is actually composed of two separate tasks on two partitions internally.

A resource is either of the following types:

- CPU

  The granularity of a CPU resource of a DLPAR operation is one CPU. More than one CPU can be specified as a resource of a DLPAR operation.

  A partition must be assigned at least the minimum number of processors specified in the partition profile, and it can be assigned up to the maximum number of processors specified in the partition profile (see 2.2.3, “Three kinds of values for resource assignment” on page 35).

  Therefore, you can dynamically add or remove processors for that partition within the range of the minimum and maximum values.

- Memory

  The granularity of a memory resource of a DLPAR operation is 256 MB[1]. Multiplies of 256 MB memory can be specified as a resource of a DLPAR operation.

  A partition must be assigned at least the minimum size of memory specified in the partition profile, and it can be assigned up to the maximum size of memory specified in the partition profile (see 2.2.3, “Three kinds of values for resource assignment” on page 35).

  Therefore, you can dynamically add or remove memory for that partition within the range of the minimum and maximum values.

[1] This memory chunk is referred to as a logical memory block (LMB).

- I/O resource

  The granularity of an I/O resource of a DLPAR operation is a PCI slot with a PCI adapter. Multiple I/O slots can be specified as a resource of a DLPAR operation. If a PCI adapter has multiple ports, all the ports and devices configured beneath the ports are treated as a resource.

  For example, if a 10/100 4-Port Ethernet adapter (FC 4961) is selected, all Ethernet devices (entX0) and interfaces (enX) configured on this adapter are treated as a single resource.

  A partition must be assigned all the adapters specified as *required* in the partition profile, and it can be assigned adapters specified as *desired* in the partition profile (see 2.2.3, "Three kinds of values for resource assignment" on page 35).

  You cannot remove I/O slots listed as required; however, you can remove I/O slots listed as desired, or those that were added as a result of a DLPAR operation. In other words, a partition can currently contain an I/O slot that is *not* listed as either desired or required in the active partition profile.

Note: A DLPAR operation can perform only one type of resource change. You cannot add and remove memory to and from the same partition in a single DLPAR operation. Also, you cannot move CPU and memory from a partition to another partition in a single DLPAR operation.

Resources removed from a partition are marked free (free resources) and owned by the global firmware of system; you can consider these resources as kept in the "free resource pool." Free resources can be added to any partition in a system as long as the system has enough free resources.

It is imperative to understand that the DLPAR function is not solely provided by AIX 5L Version 5.2, but it is supported by the integration of following components:

- Hardware

  A partioning-capable pSeries server model is required (see 1.2.1, "Supported models" on page 6).

- Firmware

  Depending on the models you have selected, a firmware update might be required (see 1.2.1, "Supported models" on page 6).

- HMC

  HMC software Release 3, Version 1 or later is required (see 1.2.2, "IBM Hardware Management Console for pSeries (HMC)" on page 8).

- Operating system

  AIX 5L Version 5.2 or later is required.

If one of these components does not satisfy the requirement to support DLPAR, the function is not available. For example, if a partition is installed with AIX 5L Version 5.1, that partition does not support DLPAR, although the other partitions installed with AIX 5L Version 5.2 support DLPAR.

3.2 The process flow of a DLPAR operation

A DLPAR operation initiated on the HMC is transferred to the target partition through Resource Monitoring and Control (RMC). The request produces a DLPAR event on the partition. After the event has completed, regardless of the result from the event, a notification will be returned to the HMC in order to mark the completion of the DLPAR operation. This means that a DLPAR operation is considered as a single transactional unit, thus only one DLPAR operation is performed at a time.

A DLPAR operation is executed in the process flow is illustrated in Figure 3-1 on page 57.

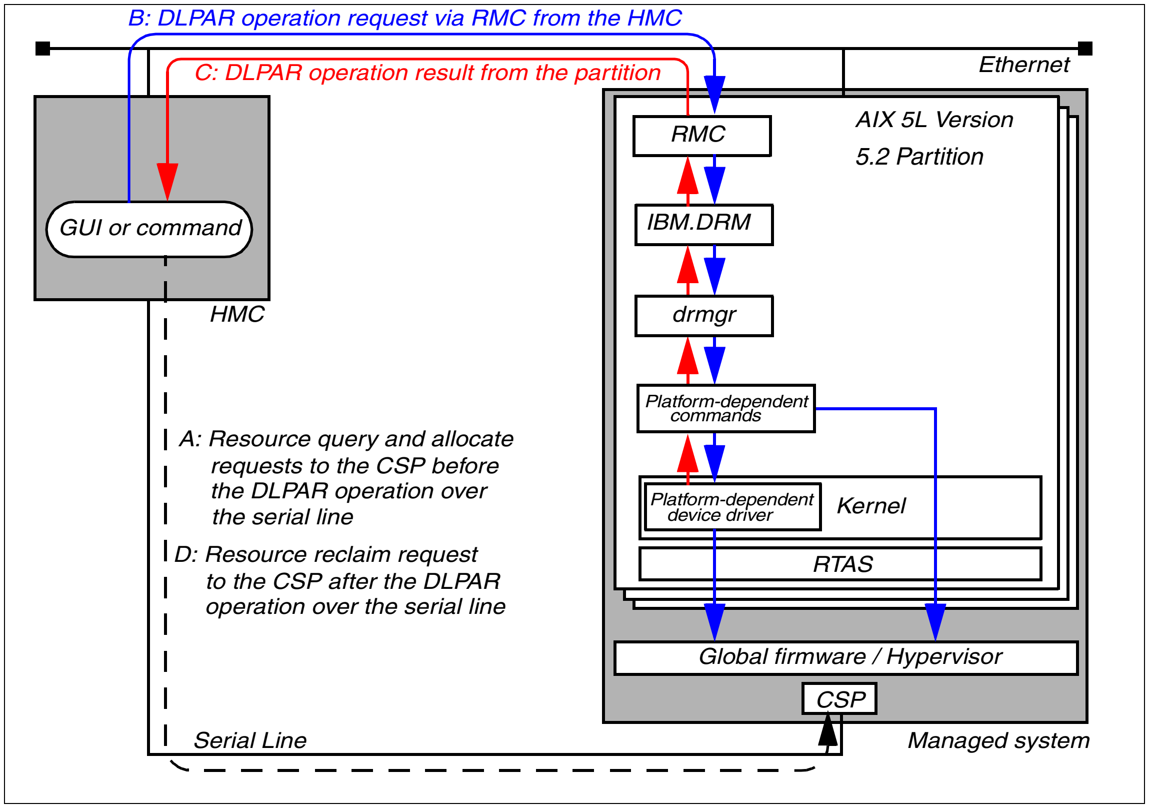

B: DLPAR operation request via RMC from the HMC
C: DLPAR operation result from the partition
Ethernet
AIX 5L Version 5.2 Partition
RMC
IBM.DRM
drmgr
Platform-dependent commands
Platform-dependent device driver
Kernel
RTAS
Global firmware / Hypervisor
CSP
Managed system
GUI or command
HMC
A: Resource query and allocate requests to the CSP before the DLPAR operation over the serial line
D: Resource reclaim request to the CSP after the DLPAR operation over the serial line
Serial Line

Figure 3-1 Process flow of a DLPAR operation

The following steps explain the process flow of a DPLAR operation:

1. The system administrator initiates a DLPAR operation request on the HMC using either the graphical user interface or command line interface. For information about the use of these interfaces, see 8.1, “Dynamic logical partitioning” on page 256 or 9.3, “Dynamic logical partitioning operations using chhwres” on page 285.
2. The requested DLPAR operation is verified on the HMC with the current resource assignment to the partition and free resources on the managed system before being transferred to the target partition. In other words, the HMC provides the policy that determines whether or not a DLPAR operation request is actually performed on the managed system. The policy is determined by the *partition profile*, explained in 2.2.3, “Three kinds of values for resource assignment” on page 35.

3. If the request is a resource addition, the HMC communicates with the global firmware in order to allocate free resources to the target partition through the service processor indicated as arrow A in Figure 3-1 on page 57.

   If enough free resources exist on the system, the HMC assigns the requested resource to the specified partition and updates the partition's object to reflect this addition, and then creates associations between the partition and the resource to be added.

4. After the requested DLPAR operation has been verified on the HMC, it will be transferred to the target partition using Resource Monitoring and Controlling (RMC), which is an infrastructure implemented on both the HMC and AIX partitions, as indicated as arrow B in Figure 3-1 on page 57. The RMC is used to provide a secure and reliable connection channel between the HMC and the partitions.

   Note: The connection channel established by RMC only exists between the HMC and the partition where the DLPAR operation is targeted to. There are no connection paths required between partitions for DLPAR operation purposes.

5. The request is delivered to the IBM.DRM resource manager running on the partition, which is in charge of the dynamic logical partitioning function in the RMC infrastructure in AIX. As shown in the following example, it is running as the IBM.DRMd daemon process and included in the devices.chrp.base.rte fileset on AIX 5L Version 5.2 or later:

```
# lssrc -ls IBM.DRM
Subsystem         : IBM.DRM
PID               : 18758
Cluster Name      : IW
Node Number       : 1
Daemon start time : Wed Aug 21 16:44:12 CDT 2002

Information from malloc about memory use:
   Total Space    : 0x003502c0 (3474112)
   Allocated Space: 0x0030b168 (3191144)
   Unused Space   : 0x00043e40 (278080)
   Freeable Space : 0x00000000 (0)

Class Name(Id)    : IBM.DRM(0x2b) Bound
# ps -ef | head -1 ; ps -ef | grep DRMd | grep -v grep
     UID   PID  PPID   C    STIME    TTY  TIME CMD
    root 18758 10444   0   Aug 21      -  0:22 /usr/sbin/rsct/bin/IBM.DRMd
# lslpp -w /usr/sbin/rsct/bin/IBM.DRMd
  File                                        Fileset              Type
  ----------------------------------------------------------------------------
  /usr/sbin/rsct/bin/IBM.DRMd
                                       devices.chrp.base.rte       File
```

Note: The absence of the IBM.DRM resource manager in the **lssrc -a** output does not always mean that the partition has not been configured appropriately for the dynamic logical partitioning. The resource manager will be automatically configured and started by RMC after the first partition reboot if the network configuration is correctly set up on the partition and the HMC.

Resource managers are sub-systems used in the RMC infrastructure. For further information about RMC and its sub-components, please refer to the following publications:

– *A Practical Guide for Resource Monitoring and Control*, SG24-6615

– *IBM Reliable Scalable Cluster Technology for AIX 5L: RSCT Guide and Reference*, SA22-7889

6. The IBM.DRM resource manager invokes the **drmgr** command, which is an platform-independent command designed as the focal point of the dynamic logical partitioning support on AIX.

   As shown in the following example, the **drmgr** command is installed in the /usr/sbin directory provided by the bos.rte.methods fileset:

```
# whence drmgr
/usr/sbin/drmgr
# lslpp -w /usr/sbin/drmgr
  File                                        Fileset               Type

----------------------------------------------------------------------------
  /usr/sbin/drmgr                             bos.rte.methods       File
```

   Note: The **drmgr** command should not be invoked by the system administrator in order to directly perform resource changes in a partition. It must be invoked in the context explained here to do so. In 3.8, “How to manage DLPAR scripts” on page 93, we provide another usage of the **drmgr** command.

7. The **drmgr** command invokes several platform-dependent commands[2] depending on the resource type (CPU, memory, or I/O resource) and request (resource addition or removal) in order to instruct the kernel to process the actual resource change with necessary information.
8. The kernel does many tasks, as described in 3.3, “Internal activity in a DLPAR event” on page 60.

[2] On the current partitioning-capable pSeries server models, the platform-dependent commands are included in the devices.chrp.base.rte fileset and installed in the /usr/lib/boot/bin directory.

9. After the DLPAR event has completed, regardless of the result, a notification will be returned to the HMC in order to mark the completion of the DLPAR operation, indicated as arrow C in Figure 3-1 on page 57. The notification also includes the exit code, standard out, and standard error from the **drmgr** command. The system administrator who has initiated the DLPAR operation will see the exit code and outputs on the HMC.
10. If the request is a resource removal, the HMC communicates with the global firmware in order to reclaim resources to the free resource pool from the source partition through the service processor indicated as arrow D in Figure 3-1 on page 57.

    The HMC unassigns the resource from the partition and updates the partition's object to reflect this removal, and then removes associations between the partition and the resource that was just removed.

A DLPAR operation can take noticeable time depending on the availability and the capability to configure or deconfigure a specific resource.

3.3 Internal activity in a DLPAR event

The AIX kernel communicates with the partition firmware through Run-Time Abstraction Services (RTAS). For more information about RTAS, see "Run-Time Abstraction Services (RTAS)" on page 23. The partition firmware manages resources in the partition (see "Open Firmware" on page 21). The resources are represented in the Open Firmware device tree that serves as a common reference point for the operating system and firmware. The RTAS operate on objects represented in this database.

Each AIX partition has a private copy of the Open Firmware device tree that reflects the resources that are actually assigned to the partition and those that might be in the future. Structurally, it is organized like a file system with directories and files, where the files represent configured instances of resources, and the directories provide the list of potential assignments. Each installed resource is represented in this list and are individually called dynamic reconfiguration connectors.

3.3.1 Internal activity for CPUs and memory in a DLPAR event

As described previously, the **drmgr** command handles all DLPAR operations by calling the appropriate commands and controls the process of the reconfiguration of resources. We provide detailed information about the three phases, check, pre, and post, in 3.5.1, “Three phases in a DLPAR event” on page 67.

The following briefly describes the kernel internal activity for CPUs and memory in a DLPAR event.

1. The Object Data Manager (ODM) lock is taken to guarantee that the ODM, Open Firmware device tree, and the kernel are automatically updated. This step can fail if the ODM lock is held for a long time and the user indicates that the DLPAR operation should have a time limit.
2. The platform-dependent command reads the Open Firmware device tree.
3. The platform-dependent command invokes the kernel to start the DLPAR event. The following steps are taken:
   a. Requesting validation.
   b. Locking DLPAR event. Only one event can proceed at a time.
   c. Saving request in global kernel DR structure that is used to pass information to signal handlers, which runs asynchronously to the platform-dependent command.
   d. Starting check phase.
4. The check phase scripts are invoked.
5. The check phase signals are sent, conditional wait if signals were posted.
6. The check phase kernel extension callout. Callback routines of registered kernel extensions are called.

   The event may fail in steps 4, 5, or 6 if any check phase handler signals an error. After the check phase has passed without an error, and the DLPAR event is in the pre phase, all pre phase application handlers will be called, even if they fail, and the actual resource change is attempted.
7. The kernel marks the start of the pre phase.
8. Pre phase scripts are invoked.
9. Pre phase signals are sent–conditional wait, if signals were posted.
10. The kernel marks the doit phase start[3]. This is an internal phase where the resource is either added to or removed from the kernel.

[3] The doit phase is shown as Resource change in Figure 3-2 on page 67.

Steps 11-13 may be repeated depending on the request. Processor-based requests never loop; only one processor can be added or removed at a time in one DLPAR operation. If more than one processor needs to be added or removed, the HMC invokes AIX once for each processor.

Memory-based requests loop at the LMB level, which represent contiguous 256 MB segments of logical memory, until the entire user request has been satisfied. The HMC remotely invokes AIX once for the complete memory request.

11. This step is only taken if adding a resource. The Open Firmware device tree is updated. The resource allocated, un-isolated, and the connector configured. When un-isolating the resource, it is assigned to the partition, and ownership is transferred from Open Firmware to AIX:
    - For processors, the identity of the global and local interrupt service is discovered.
    - For memory, the logical address and size is discovered.
12. Invoke kernel to add or remove resource:
    a. The callback functions of registered kernel extensions are called. Kernel extensions are told the specific resource that is being removed or added.
    b. The resources in the kernel are removed or added.
    c. The kernel extension in post or posterror phase are invoked.

    If steps a or b fail, the operation fails.
13. This step is only taken if removing a resource.

    The Open Firmware device tree is updated. Resources are isolated and unallocated for removal. The Open Firmware device tree must be kept updated so that the configuration methods can determine the set of resources that are actually configured and owned by the operating system.
14. Kernel marks post (or posterror) phase start depending on the success of the previous steps.
15. Invoke configuration methods so that DLPAR-aware applications and registered DLPAR scripts will see state change in the ODM.
16. The post scripts are invoked.
17. The post signals are sent to registered processes, conditional wait if signals were posted.
18. The kernel clears the DLPAR event.
19. ODM locks are released.

3.3.2 Internal activity for I/O slots in a DLPAR event

Dynamic removal and addition of I/O adapters has been provided by AIX prior to DLPAR support, utilizing the PCI adapter Hot Plug capability on the IBM RS/6000 and IBM @server pSeries models. To allow for the dynamic addition and removal of PCI I/O slots, enhancements to the **lsslot** command have been made.

PCI slots and integrated I/O devices can be listed using the new connector type slot in the **lsslot** command, as shown in the following example:

```
# lsslot -c slot
```

The output of this command looks similar to the following:

```
#Slot Description Device(s)
U1.5-P1-I1 DLPAR slot pci13 ent0
U1.5-P1-I2 DLPAR slot pci14 ent1
U1.5-P1-I3 DLPAR slot pci15
U1.5-P1-I4 DLPAR slot pci16
U1.5-P1-I5 DLPAR slot pci17 ent2
U1.5-P1/Z1 DLPAR slot pci18 scsi0
```

Before the I/O slot removal, you must delete the PCI adapter device and all its children devices from AIX. Given that ent2 in the slot U1.5-P1-I5 in the previous example is not used, the devices could be removed using the following command as the root user on the partition[4]:

```
# rmdev -l pci17 -d -R
```

After the devices have been removed from AIX, the I/O slot can be removed from the partition using the graphical user interface or command line interface on the HMC (see 8.1, “Dynamic logical partitioning” on page 256 and 9.3, “Dynamic logical partitioning operations using chhwres” on page 285, respectively).

Note: Any PCI slots defined as *required* are not eligible for the DLPAR operation (see 2.2.3, “Three kinds of values for resource assignment” on page 35).

In order to let AIX recognize the dynamically added I/O slot and its children devices to a partition, you must invoke the **cfgmgr** command as the root user on the partition. To add the previously removed I/O slot from a partition, it first needs to be reassigned to the partition using the HMC.

[4] The **-R** option instructs the **rmdev** command to delete all children devices recursively.

3.4 DLPAR-safe and DLPAR-aware applications

The dynamic logical partitioning function on AIX 5L Version 5.2 is designed and implemented to not impact the existing applications. In fact, most applications are not affected by any DLPAR operations results. Therefore, those applications are called *DLPAR-safe* applications.

There are two types of application classifications regarding DLPAR operations:

DLPAR-safe	Applications that do not fail as a result of DLPAR operations. The application's performance may suffer when resources are removed, or it may not scale as resources are added.
DLPAR-aware	Applications that incorporate DLPAR operations that allow the application to adjust its use of the system resources equal to the actual capacity of the system. DLPAR-aware applications are always DLPAR-safe.

3.4.1 DLPAR-safe

Although, most applications are DLPAR-safe without requiring any modification, there are certain instances where programs may not be inherently DLPAR-safe.

There are two cases where DLPAR operations may introduce undesirable effects in the application:

- Programs that are optimized for uni-processor may have problems when a processor is added to the system resources.
- On programs that are indexed by CPU numbers, the increased processor number may cause the code to go down an unexpected code path during its run-time checks.

In addition, applications that use uni-processor serialization techniques may experience unexpected problems. In order to resolve these concerns, system administrators and application developers need to be aware of how their applications get the number of processors.

3.4.2 DLPAR-aware

DLPAR-aware applications adapt to system resource changes caused by DLPAR operations. When these operations occur, the application will recognize the resource change and accommodate accordingly.

Two techniques can be used to make applications DLPAR-aware:

- The first method is to consistently poll for system resource changes. Polling is not the recommended way to accommodate for DLPAR operations, but it is valid for systems that do not need to be tightly integrated with DLPAR. Because the resource changes may not be immediately discovered, an application that uses polling may have limited performance. Polling is not suitable for applications that deploy processor bindings, because they represent hard dependencies.
- Applications have other methods to react to the resource change caused by DLPAR operations. See 3.5, "Integrating a DLPAR operation into the application" on page 66.

Several applications should be made DLPAR-aware, because, they need to scale with the system resources. These types of applications can increase their performance by becoming DLPAR-aware. Table 3-1 lists some examples of applications that should be made DLPAR-aware.

Note: These are only a few types of common applications affected by DLPAR operations. The system administrator and application developer should be sensitive to other types of programs that may need to scale with resource changes.

Table 3-1 Applications that should be DLPAR-aware

Application type	**Reason**
Database applications	The application needs to scale with the system. For example, the number of threads may need to scale with the number of available processors, or the number of large pinned buffers may need to scale with the available system memory.
Licence Managers	Licenses are distributed based on the number of available processors or the memory capacity.
Workload Managers	Jobs are scheduled based on system resources, such as available processors and memory.
Tools	Certain tools may report processor and memory statistics or rely on available resources.

3.5 Integrating a DLPAR operation into the application

The DLPAR operation can be integrated into the application using the following two methods:

- Script-based DLPAR event handling

  If the application is externally controlled to use a specific number of threads or to size its buffers, use this method. In order to facilitate this method, a new command, **`drmgr`**, is provided. The **`drmgr`** command is the central focal point of the dynamic logical partitioning function of AIX. The following several sections discuss the **`drmgr`** command, and typical usage examples are provided in 3.8, “How to manage DLPAR scripts” on page 93.

  We explain this method in 3.6, “Script-based DLPAR event handling” on page 70.

- API-based DLPAR event handling

  If the application is directly aware of the system configuration, and the application source code is available, use this method.

  We explain this method in 3.9, “API-based DLPAR event handling” on page 103.

Applications can monitor and respond to various DLPAR events, such as a memory addition or processor removal, by utilizing these two methods. Although, at the high-level, both methods share the same DLPAR events flow explained in the following section, several key differences exist between these two methods.

One difference is that the script-based method externally reconfigures the application once a DLPAR event takes place, while the API-based method can be directly integrated into the application by registering a signal handler so that the process can be notified with the SIGRECONFIG signal when the DLPAR event occurs.

Note: The DLPAR events of I/O resources do not notify applications in AIX 5L Version 5.2.

3.5.1 Three phases in a DLPAR event

A DLPAR event executes in three phases: *check*, *pre*, and *post*. Each phase is an automatic execution unit and will be executed in its entirety before the next phase is started. This prevents partial updates to the system. In the pre and post phases, the state of the application is permitted to change. The operating system will only act upon DLPAR requests between the pre and post phases to perform the actual resource change.

Note: If a dynamic processor deallocation occurs in a partition running AIX 5L Version 5.2 or later, it is also treated as a CPU removal DLPAR event, and thus invokes these three phases.

Figure 3-2 illustrates the three phases and the order in which they occur for a DLPAR event.

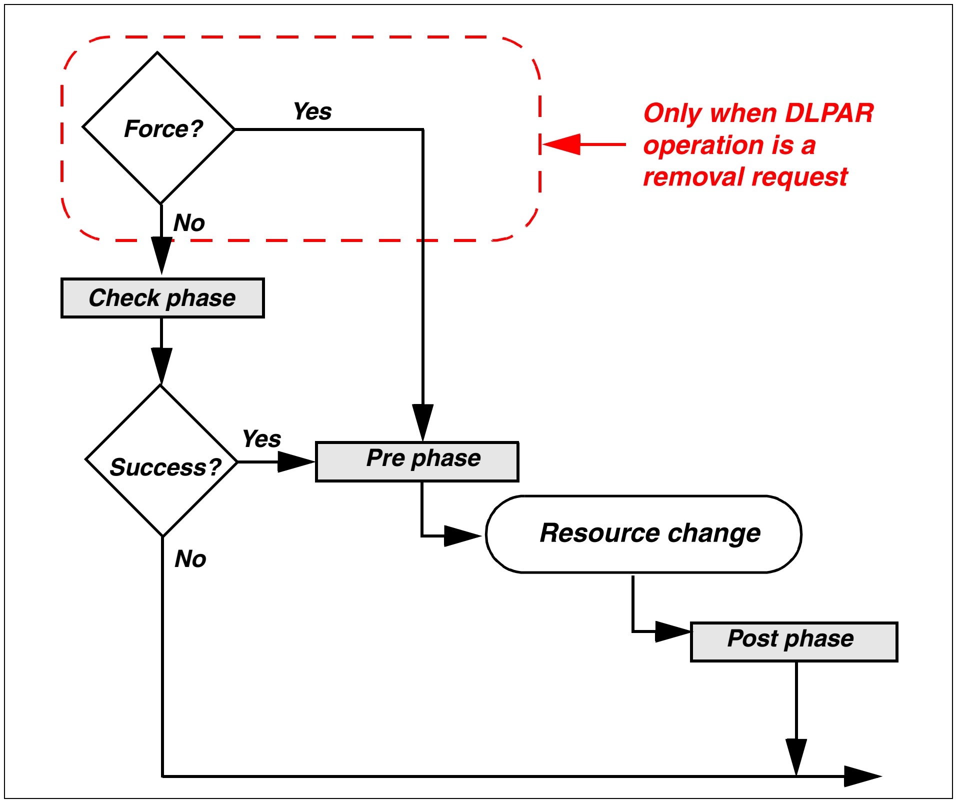


Figure 3-2 Three DLPAR phases of a DLPAR event

Check phase

The *check* phase usually occurs first. It is used to examine the resource's state and to determine if the application can tolerate a DLPAR event. It gives the script or API a chance to fail the current DLPAR operation request without changing any system state.

Note: In a resource removal DLPAR event, the check phase will be skipped if the force option is specified. In a resource addition DLPAR event, the check phase will not be skipped regardless of the force option value.

The check phase can be used in several situations, including the following:

- The check phase can determine if a processor cannot be removed because it still has threads bound to it.
- The check phase can be used by a licence manager to fail the integration of a new processor to the partition because it does not have a valid licence to support the addition of a processor.
- The check phase can even be used to maintain an application's DLPAR safeness by restricting the effects of DLPAR operations. For instance, if the application is optimized for a uniprocessor environment, the check phase could prevent the application from recognizing the addition of a processor, which could prevent the application from executing an unexpected code path with the presence of additional processors.

Note: If a DLPAR script exits with failure from the check phase, the DLPAR event will not continue. Therefore, the resource change will not be performed, and the DLPAR script will not be invoked in the pre and post phases.

Pre phase

Before the actual resource change is made, the application is notified that a resource change (addition or removal) is about to occur. The application is given a chance to prepare for the DLPAR request in the pre phase.

When the application is expecting a resource removal, the DLPAR script or API needs to carefully utilize this phase. This phase handles such things as unbinding processors, detaching pinned shared memory segments, removing plocks, and terminating the application if it does not support DLPAR or will be broken by DLPAR requests.

Note: The actual resource change takes place between the pre and post phases.

Post phase

After a resource change has occurred, the application will have a chance to respond to the DLPAR operation. The application can reconfigure itself in the post phase in order to take advantage of the resource addition or to compensate for the resource removal.

If resources are added, the DLPAR script or API could create new threads or attach to pinned shared memory segments. On the other hand, if resources are removed, the DLPAR scripts or API calls might delete threads for scalability.

3.5.2 Event phase summary

When a DLPAR request is made to change resource configurations in a partition, the **`drmgr`** command will notify applications of the pending resource change. Table 3-2 summarizes the phases of a DLPAR event and some important considerations of what needs to be accomplished in each phase.

Table 3-2 Considerations during each event phase

Phase	Considerations
Check phase	► Can the application support the request? ► Are there licence restrictions? ► Can the system withstand this application failing?
Pre phase	► Is it best to stop the application and then restart it after the DLPAR operation? ► How can the application help facilitate a DLPAR removal or addition? ► What can the application eliminate or reduce when a resource is removed? (that is, kill threads)
Post phase	► Does the application need to be restarted after the DLPAR operation? ► How can the application take advantage of added resource? (that is, start new threads) ► Did the operation complete? Was there a partial success?

3.6 Script-based DLPAR event handling

The script-based DLPAR event handling method is performed by several components, as explained in the following (see Figure 3-3 on page 71):

1. A DLPAR operation request is initiated using either the graphical user interface or command line interface on the HMC.
2. The request is transferred to the target partition through RMC. The IBM.DRM resource manager on the partition receives this request.
3. The IBM.DRM resource manager invokes the **`drmgr`** command with the necessary information that represents a DLPAR event.
4. The **`drmgr`** command invokes registered DLPAR scripts depending on the resource type, CPU, or memory that is specified by the DLPAR event. The information about the registered DLPAR scripts is kept in the DLPAR script database and fetched by the **`drmgr`** command.
5. The invoked DLPAR scripts perform necessary tasks that integrate the DLPAR operation into the application.

The DLPAR scripts should satisfy the application’s demands when a DLPAR event takes place so that the application can take the appropriate actions. Therefore, DLPAR scripts are carefully developed and tested in order for the applications’ DLPAR-awareness.

The DLPAR script can use the following commands in order to resolve the application processes’ resource dependency:

`ps`	To display bindprocessor attachments and plock system call status at the process level.
`bindprocessor`	To display online processors and make new attachments.
`kill`	To send signals to processes.
`ipcs`	To display pinned shared memory segments at the process level.
`lsrset`	To display processor sets.
`lsclass`	To display WLM classes, which might include processor sets.
`chclass`	To change WLM class difinitions.

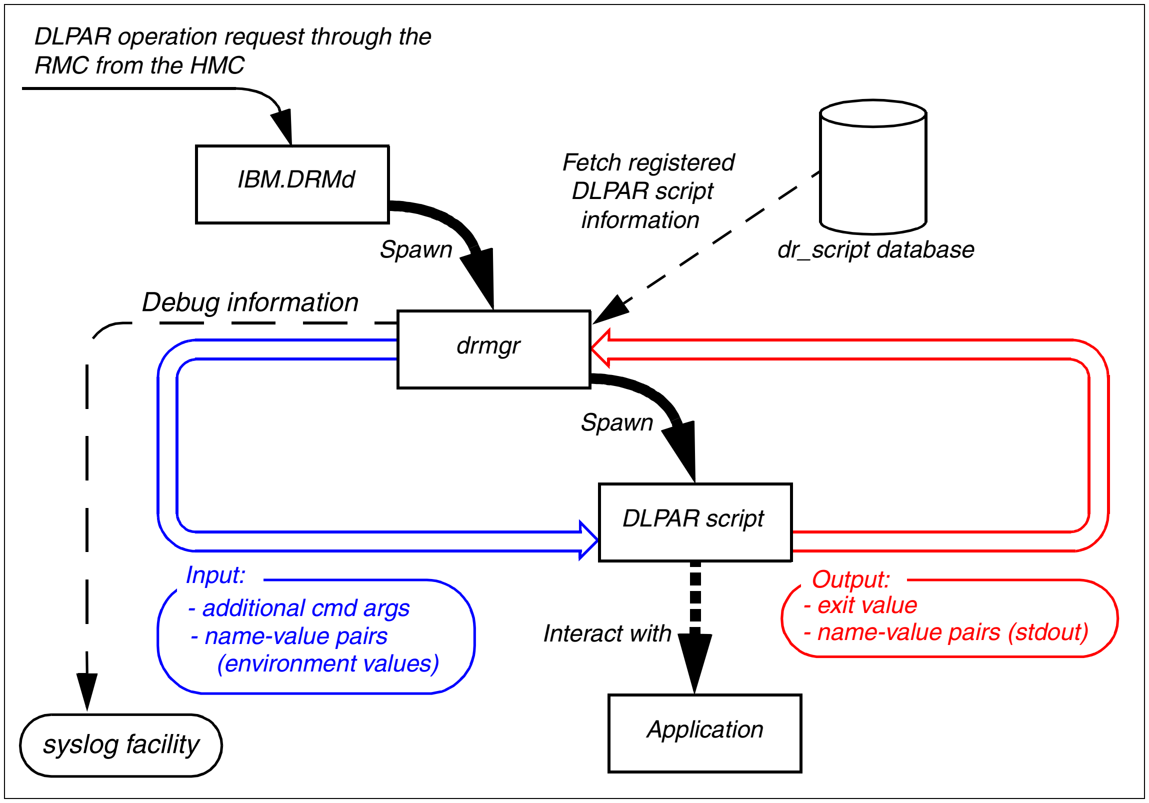

DLPAR operation request through the RMC from the HMC
IBM.DRMd
Spawn
Fetch registered DLPAR script information
dr_script database
Debug information
drmgr
Spawn
DLPAR script
Input:
- additional cmd args
- name-value pairs (environment values)
Interact with
Output:
- exit value
- name-value pairs (stdout)
syslog facility
Application

Figure 3-3 A DLPAR script invoked by the drmgr command

3.6.1 Script execution environment

When DLPAR scripts are invoked by the **drmgr** command, it sets up the following script execution environment. The required information to set up this environment is taken from the DLPAR script database and the DLPAR event.

- The UID and GID of the execution process are set to the ones of the DLPAR script.
- The current working directory is changed to /tmp.
- The PATH environment variable is set to /usr/bin:/etc:/usr/sbin.
- Two pipes are established between **drmgr** and the executing process so that the process reads using the standard in from the **drmgr** command and writes using the standard out to the **drmgr** command.

As illustrated in Figure 3-3, the execution environment defines the input and output for the DLPAR script process.

When the DLPAR script is invoked, the DLPAR script process receives its input using the following two ways:

- Additional command line arguments

  When a DLPAR script is called, the **drmgr** command will invoke it as follows:

  ```
  dr_application_script <sub-command> <additional_cmd_arg>
  ```

  In addition to the subcommands, which are explained in 3.7, "DLPAR script subcommands" on page 76, additional command arguments can be passed to the script.

- Environment variables with specified format

  When a DLPAR script is called, the **drmgr** command passes several environmental variables using a name-value pair format.

  Environmental variables that start with *DR_* are primarily used to send input data to DLPAR scripts; therefore, they should be exclusively set aside for the **drmgr** command.

  There are three types of environment values:

  – General environment values (see Table 3-3 on page 73)

  – CPU-specific environment values (Table 3-4 on page 74)

  – Memory-specific environment values (Table 3-5 on page 74)

  Note: These environment variables only exist during DLPAR events. If you want to view these variable values, the script needs to be coded to write these variables to the standard out using DR_LOG_* variables so that the **drmgr** command can forward these output to the syslog facility (see Table 3-6 on page 75).

The DLPAR script process produces its output using the following two ways:

- Exit values

  The script must return an exit value, either 0 (success) or 1 (failure). If the exit value is 1, then the DR_ERROR name-value pair must be set to describe the reason for failure, and the script must print it to the standard out.

- Standard out with specified format

  The DLPAR scripts can write strings using a name-value pair format to the standard out. The **drmgr** command will read them from the scripts. Strings that start with *DR_* are primarily used to send output data to the **drmgr** command from the DLPAR scripts.

Note: The script should not print the following to the standard out:

- A string whose length is larger than 1024 characters.
- A string that contains new line characters.
- Any strings that are undefined in Table 3-6 on page 75, Table 3-9 on page 79, Table 3-10 on page 81, and Table 3-11 on page 82.

Input (environment variables)

Table 3-3 shows general environment variables.

Table 3-3 General DLPAR environment variables

Environment variable	Description
DR_DETAIL_LEVEL=N	This name-value pair instructs the script to produce the specified level of detailed debug information sent to the standard out. The value of N must be one of the following: ► 0 - None ► 1 - Min ► 2 - Medium/more ► 3 - Max ► 4 - Debug
DR_FORCE=emergency	This name-value pair gives the emergency processing request to the script. The value of emergency must be one of the following: ► FALSE - Emergency processing is required. ► TRUE - Emergency processing is not required (default).

Note: The DR_DETAIL_LEVEL=N environment value will be set on the HMC. If you use the graphical user interface, select the **Detail level** field in the DLPAR operation panel (Figure 8-2 on page 258 can be referenced as an example). If you use the command line interface, use the `-d` option of the `chhwres` command to set the value (9.3, “Dynamic logical partitioning operations using chhwres” on page 285).

Table 3-4 on page 74 shows CPU-specific environment variables.

Table 3-4 CPU-specific DLPAR environment variables

CPU environment variables	Description
DR_LCPUID=N	The logical CPU ID of the processor that is being added or removed. N is a decimal number.
DR_BCPUID=N	The bind CPU ID of the processor that is being added or removed. N is a decimal number.

Table 3-5 shows the memory-specific environment variables.

Table 3-5 Memory-specific DLPAR environment variables

Memory environment variables	Description
DR_MEM_SIZE_REQUEST=N	Size of memory requested in megabytes. N is a decimal value.
DR_MEM_SIZE_COMPLETED=N	Number of megabytes that were successfully added or removed. N is a decimal value.
DR_FREE_FRAMES=N	Number of free frames currently in the system. Each frame is a 4 KB page. N is a 32-bit hexadecimal value.
DR_PINNABLE_FRAMES=N	Total number of pinnable frames currently in the system. Each frame is a 4 KB page. N is a 32-bit hexadecimal value.
DR_TOTAL_FRAMES=N	Total number of frames in the system. Each frame is a 4 KB page. N is a 32-bit hexadecimal value.

Output (standard out)

Table 3-6 shows general output variables. The DR_ERROR=failure_cause name=variable pair is a mandatory output when the script exits with 1 (failure).

Table 3-6 General DLPAR output variables

Variable	Description
DR_ERROR=failure_cause (only if the script exits with 1)	This name-value pair describes the reason for failure.
DR_LOG_ERR=message	This name-value pair describes the information message to be sent to the syslog facility with the err (LOG_ERR) priority.
DR_LOG_WARNING=message	This name-value pair describes the information message to be sent to the syslog facility with the warning (LOG_WARNING) priority.
DR_LOG_INFO=message	This name-value pair describes the information message to be sent to the syslog facility with the info (LOG_INFO) priority.
DR_LOG_EMERG=message	This name-value pair describes the information message to be sent to the syslog facility with the emerg (LOG_EMERG) priority.
DR_LOG_DEBUG=message	This name-value pair describes the information message to be sent to the syslog facility with the debug (LOG_DEBUG) priority.

Note: Except for the DR_ERROR variable, the other variables are used to send messages to the syslog facility.

3.6.2 DLPAR script naming convention

When developing a DLPAR script, you should follow a few simple naming conventions. We suggest naming the script using prefixes that describe the vendor name and the subsystem that it controls.

For example, dr_ibm_wlm.pl would be a good name for a DLPAR Perl script that was written by IBM to control the WLM assignments. WLM stands for Workload Manager, which is a standard function of AIX to prioritize multiple processes depending on the predefined attributes.

Another example is dr_sysadmin_wlm.pl. This could be a DLPAR Perl script provided by system administrator to control the WLM assignments.

3.7 DLPAR script subcommands

Every DLPAR script is required to accept all the subcommands found in Table 3-7 on page 76. This section provides detailed information for each subcommand.

Note: The prefix names for these subcommands (check, pre, and post) coincide with the DLPAR phases explained in 3.6, "Script-based DLPAR event handling" on page 70.

Table 3-7 DLPAR script subcommands

Subcommand name	Description
`scriptinfo`	Identifies script-specific information. It must provide the version, date, and vendor information. This command is called when the script is installed.
`register`	Identifies the resources managed by the script, such as *cpu* or *mem*.
`usage resource_name`	Returns a description of how the script plans to use the named resources. It contains pertinent information so that the user can determine whether or not to install the script. Further, the command describes the software capabilities of the applications that are impacted.
`checkrelease resource_name`	This subcommand is invoked when the **`drmgr`** command initiates the release of the specified resource. The script checks the resource dependencies of the application and evaluate the effects of resource removal on the application the script is monitoring. The script can indicate that the resource should not be removed if the application is not DLPAR-aware or if the resource is critical for the subsystem.
`prerelease resource_name`	Before the removal of the specified resource, this subcommand is invoked. The script uses this time to remove any dependencies the application may have on the resource. This command can reconfigure, suspend, or terminate the application such that the named resource can be released.
`postrelease resource_name`	After the resource is removed successfully, this subcommand is invoked. The script can perform any necessary cleaning up, or it may restart the application if it stopped the application in the prerelease phase.

Subcommand name	Description
`undoprerelease resource_name`	This subcommand is invoked if an error occurs while the resource is being released. The script takes the necessary steps to undo its prerelease operations on the resource and the application. In the case of a partial resource release, this command reads the environment variables to determine the level of success before the fail.
`checkaquire resource_name`	This subcommand is invoked to determine if the **drmgr** command can proceed with a resource addition to the application.
`preacquire resource_name`	This subcommand tells the application that a resource will be available for use.
`postacquire resource_name`	This subcommand informs the **drmgr** command that the resource addition completed, and the script allows the application to use the new resources. If the application was stopped in the preacquire phase, the application is restarted in this command.
`undopreacquire resource_name`	This subcommand notifies the **drmgr** command that the resource addition aborted or partially completed. The script then makes the necessary changes to undo anything it did in the preacquire phase, or the script determines the level of success of the DLPAR addition request by reading the environment variables.

3.7.1 The scriptinfo subcommand

When a script is fist installed, the script is invoked with the `scriptinfo` subcommand by the **drmgr** command. The `scriptinfo` subcommand displays useful information to identify the script, such as the developed date and the vendor name for it, in order to let the **drmgr** command record appropriate information in the DLPAR script database about the script. The `scriptinfo` subcommand is also called by the **drmgr** command in the very early stage of a DLPAR operation request.

When the script is invoked with the `scriptinfo` subcommand, it takes the following syntax:

```
dr_application_script scriptinfo
```

Input to the scriptinfo subcommand

The `scriptinfo` subcommand takes the following input data:

- Additional command line arguments

  None.

- Name-value pairs from environment variables

  See Table 3-3 on page 73.

Output from the scriptinfo subcommand

The `scriptinfo` subcommand produces the following output data.

- Exit value

  The script must return an exit value, either 0 (success) or 1 (failure). If the exit value is 1, the DR_ERROR name-value pair must be set to describe the reason for failure, and the script must print it to the standard out.

- Name-value pairs to the standard output stream

  Table 3-8 lists the name-value pairs that must be returned by the script when it is invoked with the `scriptinfo` subcommand.

Table 3-8 Required output name-value pairs for the scriptinfo subcommand

Required output pair	Description
DR_VERSION=1	This name-value pair indicates the version level of the script that specifies the compatibility level of the DLPAR script with respect to the DLPAR implementation version of AIX. On AIX 5L Version 5.2, the version must be set to 1, which indicates that the script is compatible with DLPAR implementation Version 1.
DR_DATE=DDMMYYYY	This name-value pair is the publication date of the script. The format should be DDMMYYYY, where DD=days, MM=months, and YYYY=year. For example, a valid date would be 08102002, which is October 8, 2002.
DR_SCRIPTINFO=description	This name-value pair contains a description of the script's functionality. This string should be a brief human-readable message.
DR_VENDOR=vendor_information	This name-value pair indicates the vendor name and related information. This string can also be used to highlight the application represented by the script.

In addition to Table 3-6 on page 75, Table 3-9 on page 79 lists the optional name-value pair that can be returned by the script when it is invoked with the `scriptinfo` subcommand. If the script needs to have more processing time for its execution, it prints the timeout value to the standard out explained in Table 3-9 on page 79 so that the **`drmgr`** command can read the appropriate timeout value for this script.

Table 3-9 Optional output name-value pairs for the scriptinfo subcommand

Optional output pair	**Description**
DR_TIMEOUT=timeout_in_seconds	This name-value pair indicates the timeout value in seconds of all DLPAR operations done in this script. The default timeout is 10 seconds. This timeout can be overridden by the **`-w`** flag of the **`drmgr`** command. The **`drmgr`** command waits for the timeout before it sends a SIGABRT to the script. After waiting 1 more second for the script to gracefully end, it will send a SIGKILL. A value of zero (0) disables the timer.

Note: Do not confuse the timeout value specified by DR_TIMEOUT with the timeout value selected in the DLPAR operation dialog panel on the HMC (see the Timeout setting (in mins) value in Figure 8-2 on page 258).

Example

In Example 3-1,two sample DLPAR scripts are registered, dr_test.sh and dr_test.pl. The emphasized lines in this example show the information recorded in the DLPAR script database. The information was derived from the script output with the `scriptinfo` subcommand upon the script registration (see Table 3-8 on page 78).

Also, the two fields, `Script Timeout` and `Admin Override Timeout` correspond to the values specified by the DR_TIMEOUT value and the **`-w`** option, respectively (see Table 3-9).

Example 3-1 Registered sample DLPAR scripts

```
# drmgr -l
DR Install Root Directory: /usr/lib/dr/scripts/all
Syslog ID: DRMGR
------------------------------------------------------------
/usr/lib/dr/scripts/all/dr_test.sh          DLPAR ksh example script
        Vendor:IBM,     Version:1,      Date:10182002
        Script Timeout:10,      Admin Override Timeout:0
        Resources Supported:
                Resource Name: cpu       Resource Usage: cpu binding for performance
                Resource Name: mem       Resource Usage: Shared(Pinned) memory for app XYZ
------------------------------------------------------------
/usr/lib/dr/scripts/all/dr_test.pl          DLPAR Perl example script
        Vendor:IBM Corp.,       Version:1,      Date:04192002
        Script Timeout:5,       Admin Override Timeout:0
        Resources Supported:
                Resource Name: cpu       Resource Usage: Testing DLPAR on CPU removal
                Resource Name: mem       Resource Usage: Testing DLPAR on MEM removal
------------------------------------------------------------
```

3.7.2 The register subcommand

When the script is invoked with the `register` subcommand by the **drmgr** command, the script is registered into the DLPAR script database. The `register` subcommand also informs the **drmgr** command about the resource type (CPU or memory) that the script is designed to handle.

When the script is invoked with the `register` subcommand, it takes the following syntax:

```
dr_application_script register
```

Input to the register subcommand

The `register` subcommand takes the following input data:

- Additional command line arguments

  None.

- Name-value pairs from environment variables

  See Table 3-3 on page 73.

Output from the register subcommand

The `register` subcommand produces the following output data:

- Exit value

  The script must return an exit value, either 0 (success) or 1 (failure). If the exit value is 1, the DR_ERROR name-value pair must be set to describe the reason for failure, and the script must print it to the standard out.

- Name-value pairs to the standard output stream

  Table 3-10 on page 81 lists the name-value pair that must be returned by the script when it is invoked with the `register` subcommand.

Table 3-10 Required output name-value pair for the register subcommand

Required output pair	Description
DR_RESOURCE=resource_name	This string identifies the resource type that the DLPAR script is designed to handle. The valid resource type names are: ► cpu ► mem If a script needs to handle both CPU and memory resource types, the script prints the following two lines: DR_RESOURCE=cpu DR_RESOURCE=mem

Optionally, the script can return the name-value pairs listed in Table 3-6 on page 75.

Example

The emphasized fields in the following example are extracted from Example 3-1 on page 80. The fields show the information recorded in the DLPAR script database. The information was derived from the script output with the `register` subcommand upon the script registration (see Table 3-10).

```
Resources Supported:
                Resource Name: cpu       Resource Usage: Testing DLPAR on CPU
removal
                Resource Name: mem       Resource Usage: Testing DLPAR on MEM
removal
```

3.7.3 The usage subcommand

The main purpose of the usage subcommand is to tell you which resource type (CPU or memory) the script is designed to handle. The usage subcommand is also called by the **drmgr** command in the very early stage of a DLPAR operation request for information purposes only.

When the script is invoked with the usage subcommand, it takes the following syntax:

```
dr_application_script usage <resource_type>
```

Input to the usage subcommand

The usage subcommand takes the following input data:

- Additional command line arguments

  The usage subcommand requires one additional command line argument that tells the **drmgr** command which resource type (CPU or memory) the script is designed to handle. The valid values are cpu or mem.

- Name-value pairs from environment variables

  See Table 3-3 on page 73.

Output from the usage subcommand

The usage subcommand produces the following output data:

- Exit value

  The script must return an exit value, either 0 (success) or 1 (failure). If the exit value is 1, the DR_ERROR name-value pair must be set to describe the reason for failure, and the script must print it to the standard out.

- Name-value pairs to the standard output stream

  Table 3-11 lists the name-value pair that must be returned by the script when it is invoked with the usage subcommand.

Table 3-11 Required output name-value pair for the usage subcommand

Required output pair	**Description**
DR_USAGE=usage_description	This name-value pair contains a human-readable string describing how the resource is used by the associated application. This description should indicate the impact on the application if that resource is removed or added.

Optionally, the script can return the name-value pairs listed in Table 3-6 on page 75.

Example

The emphasized fields in the following example are extracted from Example 3-1 on page 80. The fields show the information recorded in the DLPAR script database. The information was derived from the script output with the `usage` subcommand upon the script registration (see Table 3-10 on page 81).

```
Resources Supported:
                Resource Name: cpu       Resource Usage: Testing DLPAR on CPU
removal
                Resource Name: mem       Resource Usage: Testing DLPAR on MEM
removal
```

3.7.4 The checkrelease subcommand

Before the specified resource type is removed, the script is invoked with the `checkrelease` subcommand by the **drmgr** command. The resource is not actually changed with this subcommand.

When the **drmgr** command invokes the script with the `checkrelease` subcommand, the script determines the resource dependencies of the application, evaluate the effects of resource removal on the application, and indicate whether the resource can be successfully removed. If the resource removal request affects the application, the script returns with an exit status of 1 to let the **drmgr** command know to not release the resource.

When the script is invoked with the `checkrelease` subcommand, it takes the following syntax:

```
dr_application_script checkrelease <resource_type>
```

Input for the checkrelease subcommand

The `checkrelease` subcommand takes the following input data:

- Additional command line arguments

  The `checkrelease` subcommand requires one additional command line argument that tells the **drmgr** command which resource type (CPU or memory) the script is designed to handle. The valid values are `cpu` or `mem`.

- Name-value pairs from environment variables

  The `checkrelease` subcommand takes several required input name-value pairs from environment values depending on the resource type that the script is designed to handle:

  – If the script is registered to handle CPU, see Table 3-4 on page 74.

  – If the script is registered to handle memory, see Table 3-5 on page 74.

The `checkrelease` subcommand can also take an optional input name-value pair from the environment value shown in Table 3-3 on page 73.

> **Note:** If the DR_FORCE=TRUE environment value is passed to a script with prerelease, the script interprets the force option as an order, so it returns as soon as possible.

Output for the checkrelease subcommand

The `checkrelease` subcommand produces the following output data:

- Exit value

  The script must return an exit value, either 0 (success) or 1 (failure). If the exit value is 1, the DR_ERROR name-value pair must be set to describe the reason for failure, and the script must print it to the standard out.

  > **Note:** When invoking scripts in the `prerelease` phase, the failure of a script does not prevent the **`drmgr`** command from attempting to remove the resource. The theory is that resource removal is safe. It may fail, but the kernel is coded to cleanly remove resources, so there is no harm in trying. The return code from each script is stored so that the **`drmgr`** command can determine whether it needs to call it back. If a script fails in the `prerelease` phase, it will not be called in the `postrelease` or `undorelease` phases.

- Name-value pairs to the standard output stream

  Optionally, the script can return the name-value pairs listed in Table 3-6 on page 75.

3.7.5 The prerelease subcommand

Before the specified resource type is actually released, the script is invoked with the `prerelease` subcommand by the **`drmgr`** command. This is called after the `checkrelease` subcommand.

When the **`drmgr`** command invokes the script with the `prerelease` subcommand, the script interacts with the application, as briefly summarized in the following:

1. Informs the application about the resource removal event and lets the application release the specified resource, for example, reconfigure, suspend, or terminate the application process that uses the specified resource.
2. If the application has successfully released the specified resource, the script exits with an exit status of 0 (success).

3. Otherwise, there are two options:
   - The script exits with an exit status of 0 (success) regardless of the response from the application.
   - The script exits with an exit status of 1 (failure).

When the script is invoked with the `prerelease` subcommand, it takes the following syntax:

```
dr_application_script prerelease <resource_type>
```

Input for the prerelease subcommand

The `prerelease` subcommand takes the following input data:

- Additional command line arguments

  The `prerelease` subcommand requires one additional command line argument that tells the **`drmgr`** command which resource type (CPU or memory) the script is designed to handle. The valid values are `cpu` or `mem`.

- Name-value pairs from environment variables

  The `prerelease` subcommand takes several required input name-value pairs from environment values depending on the resource type that the script is designed to handle:

  - If the script is registered to handle CPU, see Table 3-4 on page 74.
  - If the script is registered to handle memory, see Table 3-5 on page 74.

  The `prerelease` subcommand can also take an optional input name-value pair from the environment value shown in Table 3-3 on page 73.

  Note: If the DR_FORCE=TRUE environment value is passed to a script with the `prerelease` subcommand, the script returns as soon as possible.

Output for the prerelease subcommand

The `prerelease` subcommand produces the following output data:

- Exit value

  The script must return an exit value, either 0 (success) or 1 (failure). If the exit value is 1, the DR_ERROR name-value pair must be set to describe the reason for failure, and the script must print it to the standard out.

  Note: If the script exits with 1 (failure), the **`drmgr`** command will not perform actual resource removal; however, it will invoke subsequent events (`postrelease` and `undoprerelease`) against the specified resource.

- Name-value pairs to the standard output stream

  Optionally, the script can return the name-value pairs listed in Table 3-6 on page 75.

3.7.6 The postrelease subcommand

After the specified resource type has been released from the partition, the script is invoked with the `postrelease` subcommand by the **drmgr** command. This is called after the `prerelease` subcommand.

When the **drmgr** command invokes the script with the `postrelease` subcommand, the script interacts with the application, including any necessary cleanup, for example, restarting or resuming the application if it was quiesced in the `prerelease` subcommand.

The script also takes appropriate actions if a partial success occurs. A partial success occurs when a subset of the requested number of resources was successfully removed. For example, the memory-related environment variables are checked to determine if all the requested memory frames were removed.

When the script is invoked with the `postrelease` subcommand, it takes the following syntax:

```
dr_application_script postrelease <resource_type>
```

Input for the postrelease subcommand

The `postrelease` subcommand takes the following input data:

- Additional command line arguments

  The `postrelease` subcommand requires one additional command line argument that tells the **drmgr** command which resource type (CPU or memory) the script is designed to handle. The valid values are `cpu` or `mem`.

- Name-value pairs from environment variables

  The `postrelease` subcommand takes several required input name-value pairs from environment values depending on the resource type that the script is designed to handle:

  – If the script is registered to handle CPU, see Table 3-4 on page 74.

  – If the script is registered to handle memory, see Table 3-5 on page 74.

  The `postrelease` subcommand also can take an optional input name-value pair from the environment value shown in Table 3-3 on page 73.

  Note: The force option should be ignored.

Output for the postrelease subcommand

The `postrelease` subcommand produces the following output data:

- Exit value

  The script must return an exit value, either 0 (success) or 1 (failure). If the exit value is 1, the DR_ERROR name-value pair must be set to describe the reason for failure, and the script must print it to the standard out.

- Name-value pairs to the standard output stream

  Optionally, the script can return the name-value pairs listed in Table 3-6 on page 75.

3.7.7 The undoprerelease subcommand

If the **`drmgr`** command fails to release the specified resource, it invokes the script with the `undoprerelease` subcommand to recover any necessary clean-up tasks that were done by the `prerelease` subcommand. The script undoes any actions that were taken by the script in the `prerelease` subcommand.

Note: If the specified resource has been removed successfully, the **`drmgr`** command will not invoke the script with the `undoprerelease` subcommand.

When the script is invoked with the `undoprerelease` subcommand, it takes the following syntax:

```
dr_application_script undoprerelease <resource_type>
```

Input for the undoprerelease subcommand

The `undoprerelease` subcommand takes the following input data:

- Additional command line arguments

  The `undoprerelease` subcommand requires one additional command line argument that tells the **`drmgr`** command which resource type (CPU or memory) the script is designed to handle. The valid values are `cpu` or `mem`.

- Name-value pairs from environment variables

  The `undoprerelease` subcommand takes several required input name-value pairs from environment values depending on the resource type that the script is designed to handle:

  – If the script is registered to handle CPU, see Table 3-4 on page 74.

  – If the script is registered to handle memory, see Table 3-5 on page 74.

  The `undoprerelease` subcommand also can take an optional input name-value pair from the environment value shown in Table 3-3 on page 73.

Note: The force option should be ignored.

Output for the undoprerelease subcommand

The `undoprerelease` subcommand produces the following output data:

- Exit value

  The script must return an exit value, either 0 (success) or 1 (failure). If the exit value is 1, the DR_ERROR name-value pair must be set to describe the reason for failure, and the script must print it to the standard out.

- Name-value pairs to the standard output stream

  Optionally, the script can return the name-value pairs listed in Table 3-6 on page 75.

3.7.8 The checkacquire subcommand

Before the specified resource type is added, the script is invoked with the `checkacquire` subcommand by the **`drmgr`** command. The resource is not actually changed with this subcommand.

When the **`drmgr`** command invokes the script with the `checkacquire` subcommand, the script determines the resource dependencies of the application, evaluates the effects of resource addition on the application, and indicates whether the resource can be successfully added. For example, there are some MP-unsafe applications. MP-unsafe applications[5] are not tolerative with multiple processors. If the resource addition request affects the application, the script returns with an exit status of 1 to let the **`drmgr`** command know to not add the resource.

When the script is invoked with the `checkrelease` subcommand, it takes the following syntax:

```
dr_application_script checkacquire <resource_type>
```

Input for the checkacquire subcommand

The `checkaquire` subcommand takes the following input data:

- Additional command line arguments

  The `checkacquire` subcommand requires one additional command line argument that tells the **`drmgr`** command which resource type (CPU or memory) the script is designed to handle. The valid values are `cpu` or `mem`.

[5] The MP-unsafeness of application is independent of the DLPAR implementation of AIX. It is an application design issue.

- Name-value pairs from environment variables

  The `checkacquire` subcommand takes several required input name-value pairs from environment values depending on the resource type that the script is designed to handle:

  – If the script is registered to handle CPU, see Table 3-4 on page 74.

  – If the script is registered to handle memory, see Table 3-5 on page 74.

  The `checkacquire` subcommand also can take an optional input name-value pair from the environment value shown in Table 3-3 on page 73.

  Note: The force option should be ignored.

Output for the checkacquire subcommand

The `checkacquire` subcommand produces the following output data:

- Exit value

  The script must return an exit value, either 0 (success) or 1 (failure). If the exit value is 1, the DR_ERROR name-value pair must be set to describe the reason for failure, and the script must print it to the standard out.

  Note: If the script exits with 1 (failure), the **`drmgr`** command will not add the specified resource and will not invoke subsequent events (`preacquire`, `postacquire`, and `undopreacquire`) against the specified resource (see Figure 3-2 on page 67).

- Name-value pairs to the standard output stream

  Optionally, the script can return the name-value pairs listed in Table 3-6 on page 75.

3.7.9 The preacquire subcommand

Before the specified resource type is actually acquired, the script is invoked with the `preacquire` subcommand by the **`drmgr`** command. This is called after the `checkacquire` subcommand.

When the **`drmgr`** command invokes the script with the `preacquire` subcommand, the script interacts with the application, for example, it informs the application about the resource addition and lets the application acquire the specified resource if it is DLPAR-aware.

Note: Most applications are DLPAR-safe. If your application is DLPAR-safe, but not DLPAR-aware, the script with the `preacquire` subcommand does not have to do any processing.

When the script is invoked with the `preacquire` subcommand, it takes the following syntax:

```
dr_application_script preacquire <resource_type>
```

Input for the preacquire subcommand

The `preacquire` subcommand takes the following input data:

- Additional command line arguments

  The `preacquire` subcommand requires one additional command line argument that tells the **`drmgr`** command which resource type (CPU or memory) the script is designed to handle. The valid values are `cpu` or `mem`.

- Name-value pairs from environment variables

  The `preacquire` subcommand takes several required input name-value pairs from environment values depending on the resource type that the script is designed to handle:

  – If the script is registered to handle CPU, see Table 3-4 on page 74.

  – If the script is registered to handle memory, see Table 3-5 on page 74.

  The `preacquire` subcommand also can take an optional input name-value pair from the environment value shown in Table 3-3 on page 73.

Note: When invoking scripts in the `preacquire` phase, the failure of a script does not prevent the **`drmgr`** command from attempting to add the resource. The theory is that resource addition is safe. It may fail, but the kernel is coded to cleanly add resources, so there is no harm in trying. The return code from each script is remembered so that the **`drmgr`** command can determine whether it needs to call it back. If a script fails in the `preacquire` phase, it will not be called in the `postacquire` or `undoacquire` phases.

Output for the preacquire subcommand

The `preacquire` subcommand produces the following output data:

- Exit value

  The script must return an exit value, either 0 (success) or 1 (failure). If the exit value is 1, the DR_ERROR name-value pair must be set to describe the reason for failure, and the script must print it to the standard out.

Note: If the script exits with 1 (failure), the **drmgr** command will not perform actual resource removal; however, it will invoke subsequent events (postacquire and undopreacquire) against the specified resource.

- Name-value pairs to the standard output stream

  Optionally, the script can return the name-value pairs listed in Table 3-6 on page 75.

3.7.10 The postacquire subcommand

After the specified resource type has been added to the partition, the script is invoked with the postacquire subcommand by the **drmgr** command. This is called after the preacquire subcommand.

When the **drmgr** command invokes the script with the postacquire subcommand, the script interacts with the application, including any necessary cleanup, for example, restarting or resuming the application if it was quiesced in the preacquire subcommand.

The script also takes the appropriate actions if a partial success occurs. A partial success occurs when a subset of the requested number of resources was successfully added. For example, the memory-related environment variables should be checked to determine if all of the requested memory frames were added.

When the script is invoked with the postacquire subcommand, it takes the following syntax:

```
dr_application_script postacquire <resource_type>
```

Input for the postacquire subcommand

The postacquire subcommand takes the following input data:

- Additional command line arguments

  The postacquire subcommand requires one additional command line argument that tells the **drmgr** command which resource type (CPU or memory) the script is designed to handle. The valid values are cpu or mem.

- Name-value pairs from environment variables

  The postacquire subcommand takes several required input name-value pairs from environment values depending on the resource type that the script is designed to handle:

  – If the script is registered to handle CPU, see Table 3-4 on page 74.

  – If the script is registered to handle memory, see Table 3-5 on page 74.

  The `postacquire` subcommand also can take an optional input name-value pair from the environment value shown in Table 3-3 on page 73.

  Note: The force option should be ignored.

Output for the postacquire subcommand

The `postacquire` subcommand produces the following output data:

- Exit value

  The script must return an exit value, either 0 (success) or 1 (failure). If the exit value is 1, the DR_ERROR name-value pair must be set to describe the reason for failure, and the script must print it to the standard out.

- Name-value pairs to the standard output stream

  Optionally, the script can return the name-value pairs listed in Table 3-6 on page 75.

3.7.11 The undopreacquire subcommand

If the **drmgr** command fails to add the specified resource to the partition, it invokes the script with the `undopreacquire` subcommand to recover any necessary cleanup tasks that were done by the `preacquire` subcommand. The script undoes any actions that were taken by the script in the `preacquire` subcommand.

Note: If the specified resource has been added successfully, the **drmgr** command will not invoke the script with the `undopreacquire` subcommand.

When the script is invoked with the `undopreacquire` subcommand, it takes the following syntax:

```
dr_application_script undopreacquire <resource_type>
```

Input for the undopreacquire subcommand

The `undopreacquire` subcommand takes the following input data:

- Additional command line arguments

  The `undopreacquire` subcommand requires one additional command line argument that tells the **drmgr** command which resource type (CPU or memory) the script is designed to handle. The valid values are `cpu` or `mem`.

- Name-value pairs from environment variables

  The `undopreacquire` subcommand takes several required input name-value pairs from environment values depending on the resource type that the script is designed to handle:

  – If the script is registered to handle CPU, see Table 3-4 on page 74.

  – If the script is registered to handle memory, see Table 3-5 on page 74.

  The `undopreacquire` subcommand also can take an optional input name-value pair from the environment value shown in Table 3-3 on page 73.

  Note: The force option should be ignored.

Output for the undopreacquire subcommand

The `undopreacquire` subcommand produces the following output data:

- Exit value

  The script must return an exit value, either 0 (success) or 1 (failure). If the exit value is 1, the DR_ERROR name-value pair must be set to describe the reason for failure, and the script must print it to the standard out.

- Name-value pairs to the standard output stream

  Optionally, the script can return the name-value pairs listed in Table 3-6 on page 75.

3.8 How to manage DLPAR scripts

The **`drmgr`** command must be used to manage DLPAR scripts. The function provided by the **`drmgr`** command does the following:

- Lists the registered DLPAR scripts and shows their information.
- Registers or uninstalls DLPAR scripts in the DLPAR script database.
- Changes the script install directory path. The default directory is /usr/lib/dr/scripts/all.

Note: The **`drmgr`** command is the only interface to manipulate the DLPAR script database. To use the **`drmgr`** command, you need the root authority.

In the following sections, typical **`drmgr`** command usage examples are provided.

3.8.1 List registered DLPAR scripts

To list registered DLPAR scripts and their information, type **drmgr -l**. If no scripts are registered, it returns the following output:

```
# drmgr -l
DR Install Root Directory: /usr/lib/dr/scripts/
Syslog ID: DRMGR
```

Example 3-1 on page 80 shows the example output of **drmgr -l** when DLPAR scripts are already registered.

3.8.2 Register a DLPAR script

To register a DLPAR script, type **drmgr -i script_file_name**. The script is copied into the script install path (the default value is /usr/lib/dr/scripts/all) and registered in the DLPAR script database, as shown in the following example:

```
# drmgr -l
DR Install Root Directory: /usr/lib/dr/scripts
Syslog ID: DRMGR
# ls /usr/samples/dr/scripts/IBM_template.sh
/usr/samples/dr/scripts/IBM_template.sh
# drmgr -i /usr/samples/dr/scripts/IBM_template.sh

DR script file /usr/samples/dr/scripts/IBM_template.sh installed successfully

# drmgr -l
DR Install Root Directory: /usr/lib/dr/scripts
Syslog ID: DRMGR
------------------------------------------------------------
/usr/lib/dr/scripts/all/IBM_template.sh          AIX DR ksh example script
        Vendor:IBM,     Version:1,      Date:10182002
        Script Timeout:10,      Admin Override Timeout:0
        Resources Supported:
                Resource Name: cpu       Resource Usage: cpu binding for
performance
                Resource Name: mem       Resource Usage: Shared(Pinned) memory
for app XYZ
------------------------------------------------------------
# ls /usr/lib/dr/scripts/all
total 32
-rwxr-xr-x   1 bin      bin           13609 Aug 28 22:30 IBM_template.sh
```

If the permission mode of the registered script is not appropriate, for example, no executable bits are set, then **drmgr -l** will not list the registered script name, even if the registration has been successfully completed. In this case, set the appropriate permission mode on the script and register it with the overwrite option **-f**, as shown in the following example:

```
# ls -l dr_IBM_template.sh
-rw-r--r--   1 root     system        14109 Nov 12 16:31 dr_IBM_template.sh
# drmgr -i dr_IBM_template.sh

DR script file dr_IBM_template.sh installed successfully
root@lpar01:/tmp [652] # drmgr -l
DR Install Root Directory: /usr/lib/dr/scripts
Syslog ID: DRMGR
# chmod a+x dr_IBM_template.sh
# drmgr -i dr_IBM_template.sh -f

DR script file dr_IBM_template.sh installed successfully

# drmgr -l
DR Install Root Directory: /usr/lib/dr/scripts
Syslog ID: DRMGR
------------------------------------------------------------
/usr/lib/dr/scripts/all/dr_IBM_template.sh              AIX DR ksh example
script
        Vendor:IBM,     Version:1,      Date:18102002
        Script Timeout:10,      Admin Override Timeout:0
        Resources Supported:
                Resource Name: cpu       Resource Usage: cpu binding for
performance
                Resource Name: mem       Resource Usage: Shared(Pinned) memory
for app XYZ
------------------------------------------------------------
```

3.8.3 Uninstall a registered DLPAR script

To uninstall a registered DLPAR script, type **drmgr -u script_file_name**. The script is unregistered from the DLPAR script database, as shown in the following example:

```
# drmgr -u IBM_template.sh

DR script file IBM_template.sh uninstalled successfully

# drmgr -l
DR Install Root Directory: /usr/lib/dr/scripts
Syslog ID: DRMGR
```

The uninstalled script file name is renamed in the script install path, as shown in the following example:

```
# ls -l /usr/lib/dr/scripts/all
total 32
-rw-r--r--   1 bin      bin           13609 Aug 28 22:30 .IBM_template.sh
```

Please note that a dot character is added in front of the original file name.

3.8.4 Change the script install path

To change the script install path, type **drmgr -R new_dir**. In the following example, the script install path is changed to the newly created directory /local/lpar01.itsc.austin.ibm.com®[6] from the default path /usr/lib/dr/scripts:

```
# drmgr -l
DR Install Root Directory: /usr/lib/dr/scripts
Syslog ID: DRMGR
# mkdir -p /local/`hostname`
# drmgr -R /local/`hostname`

0930-022 DR script ROOT directory set to:/local/lpar01.itsc.austin.ibm.com
successfully

# drmgr -l
DR Install Root Directory: /local/lpar01.itsc.austin.ibm.com
Syslog ID: DRMGR
```

Note: If you have changed the script install path, scripts that are already registered will not be referenced by the **drmgr** command.

3.8.5 The drmgr command line options

Table 3-12 on page 97 lists the **drmgr** command line options and their purpose. For further information about the **drmgr** command, type **man drmgr** on the command line prompt or refer to *AIX 5L Version 5.2 Reference Documentation: Commands Reference*, available at:

http://techsupport.services.ibm.com/server/library

[6] In this example, the **hostname** command returns the fully qualified host name (FQDN), lpar01.itsc.austin.ibm.com on one of our test partitions.

Table 3-12 The drmgr command line options

Command option	Brief description	Detailed description
-i script_name Other associated options: **[-D install_directory]** **[-w timeout]** **[-f]**	Installs a DLPAR script to the default or specified directory.	The system administrator should use the **-i** flag to install a DLPAR script. The script's file name is used as input. Unless the **-D** flag is used, the scripts are installed into the /usr/lib/dr/scripts/all/ directory under the root install directory (see **-R** flag). Permissions for the DLPAR script are the same as the script_name file. If a script with the same name is already registered, the install will fail with a warning unless the force option is used.
-w timeout	Timeout value in minutes.	This option is used in conjunction with the **-i** option. The **drmgr** command will override the timeout value specified by the LPAR script with the new-user defined timeout value.
-f	Forces an override.	During the installation of a script, the **-f** option can be set to force an override of a duplicate DLPAR script name.
-u script_name Other associated options: **[-D host_name]**	Uninstalls a DLPAR script.	The system administrator invokes this command to uninstall a DLPAR script. The script file name is provided as an input. The user can specify the directory from where the script should be removed by using the **-D** option. If no directory is specified, the command will try to remove the script from the all directory under the root directory (see the **-R** option). If the script is registered using the **-D** option, it will only be invoked on a system with that host name. If no file is found, the command will return with an error.
-R base_directory_path	Sets the root directory where the DLPAR scripts are installed.	The default value is /usr/lib/dr/scripts/. The installer looks at the all or hosts directory under this root directory. (/usr/lib/dr/scripts/all/).
-d debug_level	Sets the debug level.	This option sets the DR_DEBUG environment variable, which controls the level of debug messages from the DLPAR scripts.

Command option	Brief description	Detailed description
-l	Lists DLPAR scripts.	This option lists the details of all DLPARR scripts currently active on the system.
-b	Rebuilds DLPAR script database.	This option rebuilds the DLPAR script database by parsing through the entire list of DLPAR script install directories.
-S syslog_chan_id_str	Specifies a syslog channel.	This option enables the user to specify a particular channel to which the syslog messages have to be logged from the DLPAR script by the **drmgr** command (see "The syslog facility" on page 118).

3.8.6 Sample output examples from a DLPAR script

Although, the syslog facility can be used to record debug information, we decided to write our example DLPAR script to send the debug information to /tmp/dr_api_template.pl.dbg for readability reasons.

After registering the DLPAR script written in Perl (see Example B-2 on page 409), we initiated the following DLPAR operations on the HMC:

- 2 GB memory addition
- 1 GB memory removal
- 1 CPU addition
- 2 CPU removal

To perform a DLPAR operation using the graphical user interface on the HMC, refer to 8.1, "Dynamic logical partitioning" on page 256. If you use the command line interface on the HMC, refer to 9.3, "Dynamic logical partitioning operations using chhwres" on page 285.

Sample output: 2 GB memory addition

Example 3-2 shows the sample output of a 2 GB memory addition DLPAR event. You will notice that there are three line blocks for the check, pre, and post phases.

Example 3-2 Sample output: 2 GB memory addition

```
-- start checkacquire phase --
DR_DRMGR_INFO=DRAF architecture Version 1
DR_FORCE=FALSE
DR_FREE_FRAMES=0x30a
DR_MEM_SIZE_COMPLETED=0x0
DR_MEM_SIZE_REQUEST=0x80000000
DR_PINNABLE_FRAMES=0x54a55
DR_TOTAL_FRAMES=0x80000
mem resources: 0x80000000
-- end checkacquire phase --
-- start preacquire phase --
DR_DRMGR_INFO=DRAF architecture Version 1
DR_FORCE=FALSE
DR_FREE_FRAMES=0x30a
DR_MEM_SIZE_COMPLETED=0x0
DR_MEM_SIZE_REQUEST=0x80000000
DR_PINNABLE_FRAMES=0x54a55
DR_TOTAL_FRAMES=0x80000
-- end preacquire phase --
-- start undopreacquire phase --
DR_DRMGR_INFO=DRAF architecture Version 1
DR_FORCE=FALSE
DR_FREE_FRAMES=0x30a
DR_MEM_SIZE_COMPLETED=0x800
DR_MEM_SIZE_REQUEST=0x80000000
DR_PINNABLE_FRAMES=0x54a55
DR_TOTAL_FRAMES=0x80000
-- end undopreacquire phase --
```

Before the DLPAR operation, the partition had 2 GB memory assigned, as shown in the following example:

```
# lsattr -El mem0
size     2048 Total amount of physical memory in Mbytes  False
goodsize 2048 Amount of usable physical memory in Mbytes False
```

After the completion of the DLPAR operation, the memory size has been increased to 4 GB, as shown in the following example:

```
# lsattr -El mem0
size     4096 Total amount of physical memory in Mbytes  False
goodsize 4096 Amount of usable physical memory in Mbytes False
```

Sample output: 1 GB memory removal

Example 3-3 shows the sample output of a 1 GB memory removal DLPAR event. You will see there are three line blocks for the check, pre, and post phases.

Example 3-3 Sample output: 1 GB memory removal

```
-- start checkrelease phase --
DR_DRMGR_INFO=DRAF architecture Version 1
DR_FORCE=FALSE
DR_FREE_FRAMES=0x7e2f5
DR_MEM_SIZE_COMPLETED=0x0
DR_MEM_SIZE_REQUEST=0x40000000
DR_PINNABLE_FRAMES=0xac3f3
DR_TOTAL_FRAMES=0x100000
-- end checkrelease phase --
-- start prerelease phase --
DR_DRMGR_INFO=DRAF architecture Version 1
DR_FORCE=FALSE
DR_FREE_FRAMES=0x7e2f5
DR_MEM_SIZE_COMPLETED=0x0
DR_MEM_SIZE_REQUEST=0x40000000
DR_PINNABLE_FRAMES=0xac3f3
DR_TOTAL_FRAMES=0x100000
-- end prerelease phase --
-- start postrelease phase --
DR_DRMGR_INFO=DRAF architecture Version 1
DR_FORCE=FALSE
DR_FREE_FRAMES=0x7e2f5
DR_MEM_SIZE_COMPLETED=0x400
DR_MEM_SIZE_REQUEST=0x40000000
DR_PINNABLE_FRAMES=0xac3f3
DR_TOTAL_FRAMES=0x100000
-- end postrelease phase --
```

Before the DLPAR operation, the partition had 4 GB memory assigned, as shown in the following example:

```
# lsattr -El mem0
size     4096 Total amount of physical memory in Mbytes  False
goodsize 4096 Amount of usable physical memory in Mbytes False
```

After the completion of the DLPAR operation, the memory size has been increased to 3 GB, as shown in the following example:

```
# lsattr -El mem0
size     3072 Total amount of physical memory in Mbytes  False
goodsize 3072 Amount of usable physical memory in Mbytes False
```

Sample output: 1 CPU addition

Example 3-4 shows the sample output of a 1 CPU addition DLPAR event. You will see there are three line blocks for the check, pre, and post phases for the CPU ID 2.

Example 3-4 Sample output: 1 CPU addition

```
-- start checkacquire phase --
DR_BCPUID=2
DR_DRMGR_INFO=DRAF architecture Version 1
DR_FORCE=FALSE
DR_LCPUID=2
cpu resources: logical 2, bind 2
-- end checkacquire phase --
-- start preacquire phase --
DR_BCPUID=2
DR_DRMGR_INFO=DRAF architecture Version 1
DR_FORCE=FALSE
DR_LCPUID=2
-- end preacquire phase --
-- start undopreacquire phase --
DR_BCPUID=2
DR_DRMGR_INFO=DRAF architecture Version 1
DR_FORCE=FALSE
DR_LCPUID=2
-- end undopreacquire phase --
```

Before the DLPAR operation, the partition had two processors assigned, as shown in the following example:

```
# lsdev -Cc processor -S Available
proc6 Available 00-06 Processor
proc7 Available 00-07 Processor
```

After the completion of the DLPAR operation, the number of active processors has been increased to three, as shown in the following example:

```
# lsdev -Cc processor -S Available
proc6  Available 00-06 Processor
proc20 Available 00-20 Processor
proc7  Available 00-07 Processor
```

Sample output: 2 CPU removal

Example 3-5 shows the sample output of a 2 CPU removal DLPAR event. You will see there are three line blocks for the check, pre, and post phases for each CPU (ID 2 and 3).

Example 3-5 Sample output: 2 CPU removal

```
-- start checkrelease phase --
DR_BCPUID=3
DR_DRMGR_INFO=DRAF architecture Version 1
DR_FORCE=FALSE
DR_LCPUID=3
-- end checkrelease phase --
-- start prerelease phase --
DR_BCPUID=3
DR_DRMGR_INFO=DRAF architecture Version 1
DR_FORCE=FALSE
DR_LCPUID=3
-- end prerelease phase --
-- start postrelease phase --
DR_BCPUID=3
DR_DRMGR_INFO=DRAF architecture Version 1
DR_FORCE=FALSE
DR_LCPUID=3
-- end postrelease phase --
-- start checkrelease phase --
DR_BCPUID=2
DR_DRMGR_INFO=DRAF architecture Version 1
DR_FORCE=FALSE
DR_LCPUID=2
-- end checkrelease phase --
-- start prerelease phase --
DR_BCPUID=2
DR_DRMGR_INFO=DRAF architecture Version 1
DR_FORCE=FALSE
DR_LCPUID=2
-- end prerelease phase --
-- start postrelease phase --
DR_BCPUID=2
DR_DRMGR_INFO=DRAF architecture Version 1
DR_FORCE=FALSE
DR_LCPUID=2
-- end postrelease phase --
```

Before the DLPAR operation, the partition had four processors assigned, as shown in the following example:

```
# lsdev -Cc processor -S Available
proc6  Available 00-06 Processor
proc20 Available 00-20 Processor
proc7  Available 00-07 Processor
proc21 Available 00-21 Processor
```

After the completion of the DLPAR operation, the number of active processes has been decreased to two, as shown in the following example:

```
# lsdev -Cc processor -S Available
proc6 Available 00-06 Processor
proc7 Available 00-07 Processor
```

3.9 API-based DLPAR event handling

AIX 5L Version 5.2 has introduced a new system call, dr_reconfig(), and a new signal, SIGRECONFIG. The applications modified to adequately use the new system call and signal are considered DLPAR-safe.

A properly written DLPAR-aware application registers a signal handler that calls dr_reconfig(). When a DLPAR event occurs, the application receives SIGRECONFIG signals from the kernel in order to notify the DLPAR event.

Note that the SIGRECONFIG signal is also sent (along with the SIGCPUFAIL signal for backward compatibility) in the case of a CPU Guard[7] event. Therefore, this API-based method can also be utilized by CPU Guard-aware applications.

The signal is sent twice (check and pre phases) before the actual resource change (addition or removal), and sent once (post phase) after the resource change.

In the first release of DLPAR support, DLPAR events for I/O slots do not notify applications using the dr_config() system call.

3.9.1 The dr_reconfig system call

The dr_reconfig() system call is provided to query the information of the current DLPAR event. The system call must be called from a registered signal handler in order for the application be notified from the kernel when a DLAR event occurs. The sigaction() system call is used to register a signal handler.

[7] CPU Guard is also known as dynamic processor deallocation.

To use dr_reconfig() in your C language application, you need to add the following compiler directive line that instructs the preprocessor to include the /usr/include/sys/dr.h file:

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

Example 3-6 shows the prototype definition of dr_reconfig.

Example 3-6 The dr_reconfig system call usage

```
int dr_reconfig(int flags, dr_info_t *info);

0 is returned for success; otherwise,
-1 is returned, and the errno is set to the appropriate value.
```

The dr_reconfig() system call takes two parameters. The flags determine what the system call does. The info parameter is a structure that contains DLPAR-specific data that the signal handler uses to process DLPAR events accordingly. Table 3-13 shows the supported flags.

Table 3-13 The dr_reconfig flag parameters[8]

Flags	**Description**
DR_QUERY	This flag identifies the current DLPAR event. It also identifies any actions, if any, that the application should take to comply with the current DLPAR event. Any pertinent information is returned in the second parameter.
DR_EVENT_FAIL	This flag fails the current DLPAR event. It requires root authority.
DR_RECONFIG_DONE	This flag is used in conjunction with the DR_QUERY flag. The application notifies the kernel that the actions it took to comply with the current DLPAR request are now complete. The dr_info structure identifying the DLPAR request that was returned earlier is passed as an input parameter.

The other parameter is a pointer to a structure that hold DLPAR-specific information. The signal handler must allocate space for the dr_info_t data structure. The AIX kernel will populate this data structure and return it to the signal handler.

[8] The DR_RECONFIG_DONE flag is available on AIX 5L Version 5.2 plus 5200-01 Recommended Maintenance Level and later.

Example 3-7 shows the definition of the dr_info structure.

Example 3-7 The dr_reconfig info parameter

```
typedef struct dr_info {
        unsigned int        add : 1;    /* add operation                     */
        unsigned int        rem : 1;    /* remove operation                  */
        unsigned int        cpu : 1;    /* target resource is a cpu          */
        unsigned int        mem : 1;    /* target resource is memory         */
        unsigned int      check : 1;    /* check phase in effect             */
        unsigned int        pre : 1;    /* pre phase in effect               */
        unsigned int       doit : 1;    /* doit phase in effect              */
        unsigned int       post : 1;    /* post phase in effect              */
        unsigned int  posterror : 1;    /* post error phase in effect        */
        unsigned int      force : 1;    /* force option in effect            */
        unsigned int   bindproc : 1;    /* process has bindprocess dependency*/
        unsigned int   softpset : 1;    /* process has soft processor set
                                           dependency */
        unsigned int   hardpset : 1;    /* process has hard processor set
                                           dependency */
        unsigned int      plock : 1;    /* process has plock'd memory        */
        unsigned int       pshm : 1;    /* process has pinned shared memory  */

        /* The following fields are filled out for cpu based requests        */

        int     lcpu;                   /* logical id of target cpu          */
        int     bcpu;                   /* bind-id of target cpu             */

        /* The following fields are filled out for memory based requests     */

        size64_t  req_memsz_change;     /* Request size in bytes             */
        size64_t  sys_memsz;            /* System Memory size */
        rpn64_t   sys_free_frames;      /* Number of free frames in the
                                           system */
        rpn64_t   sys_pinnable_frames;  /* Number of pinnable frames in
                                           system */
        rpn64_t   sys_total_frames;     /* Total number of frames in system */

        int  reserved[12];
} dr_info_t;
```

The bindproc and bindpset bits are only set if the request is to remove a CPU. If the bindproc is set, then the process has a bindprocessor() attachment that must be resolved before the operation is allowed. If the bindpset bit is set, the application has processor set attachment, which can be lifted by calling the appropriate processor set interface.

The plock and pshm bits are only set if the DLPAR request is to remove memory and the process has plock() memory or is attached to a pinned shared memory segment. If the plock bit is set, the application calls plock() to unpin itself. If the pshm bit is set, the application detaches its pinned memory segments. The memory remove request may succeed, even if the pshm bit is set, as long as there is enough pinnable memory in the partition. Therefore, an action may not be required for the pshm bit to be set, but it is strongly recommended. The sys_pinnable_frames field provides the necessary information if the system has enough excess pinnable memory.

For further detailed information about the dr_reconfig system call, please refer to *AIX 5L Version 5.2 Technical Reference: Base Operating System and Extensions*, available at:

http://techsupport.services.ibm.com/server/library

Programming implications of CPU DLPAR events

At boot time, CPUs are configured in the kernel. In AIX 5L Version 5.2, a processor is identified by three different identifications, namely:

- The physical CPU ID, which is derived from the Open Firmware device tree and used to communicate with RTAS.
- The logical CPU ID, which is a ppda[9]-based index of online and offline CPUs.
- The bind CPU ID, which is the index of online CPUs.

The logical and bind CPU IDs are consecutive and have no holes in the numbering. No guarantee is given across boots that the CPUs will be configured in the same order, or even that the same CPUs will be used in a partitioned environment at all.

At system startup, the logical and bind CPU IDs are both consecutive and have no holes in the numbering; however, DLPAR operations can remove a processor from the middle of the logical CPU list. The bind CPU IDs remain consecutive because they refer only to online CPUs, so the kernel has to explicitly map these IDs to logical CPU IDs (containing online and offline CPU IDs).

The range of logical CPU IDs is defined to be 0 to M-1, where M is the maximum number of CPUs that can be activated within the partition. M is derived from the Open Firmware device tree. The logical CPU IDs name both online and offline CPUs. The rset[10] APIs are predicated on the use of logical CPU IDs.

[9] Per processor description area.
[10] Resource set.

The range of bind CPU IDs is defined to be 0 to N-1; however, N is the current number of online CPUs. The value of N changes as processors are added and removed from the system by either DLPAR or CPU Guard. In general, new processors are always added to the Nth position. Bind CPU IDs are used by the system call bindprocessor and by the kernel service switch_cpu.

The number of potential CPUs can be determined by:

- _system_configuration.max_ncpus
- _system_configuration.original_ncpus
- var.v_ncpus_cfg
- sysconf(_SC_NPROCESSORS_CONF)

The number of online CPUs can be determined by:

- _system_configuration.ncpus
- var.v_ncpus
- sysconf(_SC_NPROCESSORS_ONLN)

The _system_configuration structure is defined in the /usr/include/sys/systemcfg.h header file, and those members can be accessed from your application, as shown in the following code fraction example:

```
#include <sys/systemcfg.h>
printf("_system_configuration.original_ncpus=%d\n"
    , _system_configuration.original_ncpus);
```

The var structure is defined in the /usr/include/sys/var.h header file and populated by the sysconfig system call. The following code fraction example demonstrates how to retrieve var.v_ncpus:

```
#include <sys/types.h>
#include <sys/sysconfig.h>
#include <sys/var.h>
struct var myvar;
rc = sysconfig(SYS_GETPARMS, &myvar, sizeof(struct var));
if (rc == 0)
    printf("var.v_ncpus = %d\n", myvar.v_ncpus);
```

The number of online CPUs can also be determined from the command line. The following commands are provided by AIX:

- **`bindprocessor -q`**
- **`lsrset -a`**

As previoulsy mentioned, AIX supports two programming models for CPUs: the bindprocessor model that is based on bind CPU IDs and the rset API model that is based on logical CPU IDs. Whenever a program implements any of these programming models, it should be DLPAR-aware.

A detailed description of these kernel subroutines is given in *AIX 5L Version 5.2 Technical Reference: Kernel and Subsystems*, available at:

`http://techsupport.services.ibm.com/server/library`

Also several programming examples using the rset API are shown in Chapter 11, “Resource sets” on page 341.

The following new interfaces (system calls and kernel services) are provided to query bind and logical CPU IDs and the mapping between them:

- mycpu(): Returns bind CPU ID of the process
- my_lcpu(): Returns bind CPU ID of the process
- b2lcpu(): Returns the bind to logical CPU ID mapping
- l2bcpu(): Returns the logical to bind CPU ID mapping

3.9.2 A sample code using the dr_reconfig system call

We have developed a sample application written in the C language using the dr_reconfig() system call (see Example B-4 on page 434). Because the source code is long, we excerpt the important part with annotations from the example.

Basically, this application does nothing voluntary, except for the signal handler registration. It just does the busy loop in the while loop in main and waits until the SIGRECONFIG signal is delivered. You must implement your application logic in the while loop, specified by the comment `Your application logic goes here`.

The behavior of the application is briefly explained in the following:

1. Register a signal handler, dr_func(), in main (indicated as #A in the comment). The signal handler is registered in order to react to the SIGRECONFIG signal when it is delivered to the application process.

   ```
   if ((rc = sigaction(SIGRECONFIG, &sigact, &sigact_save)) != 0) { /* #A */
   ```

2. Wait in the busy loop in main (#B) until the SIGRECONFIG signal is sent:

   ```
   while (1) { /* #B */
       ;
       /* your application logic goes here. */
   }
   ```

3. After the SIGRECONFIG signal is delivered by the kernel, the signal handler, dr_func(), is invoked.

4. The handler calls dr_reconfig() in order to query the dr_info structure data. The dr_info structure is used to determine what DLPAR operation triggers this signal (#C).

   ```
   l_rc = dr_reconfig(DR_QUERY, &dr_info);              /* #C */
   ```

   Note: You must include the following preprocessor directive line to use the dr_reconfig() system call:

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

5. The handler parses the dr_info structure to determine the DLPAR operation type:

   – If the dr_info.add member is set, this signal is triggered by a DLPAR resource addition request (#D):

   ```
   if (dr_info.add) {                                   /* #D */
   ```

   – If the dr_info.rem member is set, this signal is triggered by a DLPAR resource removal request (#E):

   ```
   if (dr_info.rem) {                                   /* #E */
   ```

6. The handler again parses the dr_info structure to determine the DLPAR resource type:

   – If the dr_info.cpu member is set, this signal is triggered by a DLPAR CPU resource addition or removal request (#F):

   ```
   if (dr_info.cpu) {                                   /* #F */
   ```

   – If the dr_info.mem member is set, this signal is triggered by a DLPAR memory resource addition or removal request (#G):

   ```
   } else if (dr_info.mem) {                            /* #G */
   ```

7. Invoke the corresponding function based on the information determined:

   – If the requested DLPAR resource type is CPU, call the function pointer stored in the l_currentPhase->cpu_ptr array (#H):

   ```
   l_rc = l_currentPhase->cpu_ptr();                /* #H */
   ```

   – If the requested DLPAR resource type is memory, call the function pointer stored in the l_currentPhase->mem_ptr array (#I):

   ```
   l_rc = l_currentPhase->mem_ptr();                /* #I */
   ```

   Note: You must modify the functions included in the definedPhase array (#J) by adding your own logic in order to react against DLPAR operation phases. The comment `Perform actions here` specifies the location where you will modify the functions.

3.9.3 Sample output examples from a DLPAR-aware application

Although, the syslog facility can be used to record debug information, we decided to write our example application to send the debug information to /tmp/dr_api_template.C.dbg for readability reasons.

After compiling the C source code (see “Sample DLPAR-aware application using a signal handler” on page 433), we ran the application and initiated several DLPAR operations on the HMC.

The following several examples exhibit the internal behaviors of the following DLPAR operations:

- 1 GB memory addition
- 1 GB memory removal
- 2 CPU addition
- 1 CPU removal

To perform a DLPAR operation using the graphical user interface on the HMC, refer to 8.1, “Dynamic logical partitioning” on page 256. If you use the command line interface on the HMC, refer to 9.3, “Dynamic logical partitioning operations using chhwres” on page 285.

Sample output: 1 GB memory addition

Example 3-8 shows the sample output of a 1 GB memory addition DLPAR event. You will notice that there are three line blocks for the check, pre, and post phases.

Example 3-8 Sample output: 1 GB memory addition

```
---Start of Signal Handler---
An add request for
   ** check phase **
Resource is Memory.
   requested memory size (in bytes) = 1073741824
   system memory size = 2147483648
   number of free frames in system = 29434
   number of pinnable frams in system = 339916
   total number of frames in system = 524288
*****Entered CheckedAcquire_mem*****
---end of signal handler---

---Start of Signal Handler---
An add request for
   ** pre  phase **
Resource is Memory.
   requested memory size (in bytes) = 1073741824
   system memory size = 2147483648
   number of free frames in system = 29434
   number of pinnable frams in system = 339916
   total number of frames in system = 524288
*****Entered PreeAcquire_mem*****
---end of signal handler---

---Start of Signal Handler---
An add request for
   ** post phase **
Resource is Memory.
   requested memory size (in bytes) = 1073741824
   system memory size = 2147483648
   number of free frames in system = 284761
   number of pinnable frams in system = 516763
   total number of frames in system = 786432
*****Entered PostAcquire_mem*****
---end of signal handler---
```

Sample output: 1 GB memory removal

Example 3-9 shows the sample output of a 1 GB memory removal DLPAR event. You will see there are three line blocks for the check, pre, and post phases.

Example 3-9 Sample output: 1 GB memory removal

```
---Start of Signal Handler---
A remove request for
   ** check phase **
Resource is Memory.
   requested memory size (in bytes) = 1073741824
   system memory size = 3221225472
   number of free frames in system = 284771
   number of pinnable frams in system = 516763
   total number of frames in system = 786432
*****Entered CheckeRelease_mem*****
---end of signal handler---

---Start of Signal Handler---
A remove request for
   ** pre phase **
Resource is Memory.
   requested memory size (in bytes) = 1073741824
   system memory size = 3221225472
   number of free frames in system = 284770
   number of pinnable frams in system = 516763
   total number of frames in system = 786432
*****Entered PreRelease_mem*****
---end of signal handler---

---Start of Signal Handler---
A remove request for
   ** post phase **
Resource is Memory.
   requested memory size (in bytes) = 1073741824
   system memory size = 3221225472
   number of free frames in system = 29043
   number of pinnable frams in system = 339916
   total number of frames in system = 524288
*****Entered PostReleasee_mem*****
---end of signal handler---
```

Sample output: 2 CPU addition

Example 3-10 shows the sample output of a 2 CPU addition DLPAR event. You will see there are three line blocks for the check, pre, and post phases for each CPU (CPU ID 2 or 3).

Example 3-10 Sample output: 2 CPU addition

```
---Start of Signal Handler---
An add request for
   ** check phase **
Resource is CPU .
   logical CPU ID = 2
   Bind CPU ID = 2
*****Entered CheckedAcquire_cpu*****
---end of signal handler---

---Start of Signal Handler---
An add request for
   ** pre  phase **
Resource is CPU .
   logical CPU ID = 2
   Bind CPU ID = 2
*****Entered PreAcquire_cpu*****
---end of signal handler---

---Start of Signal Handler---
An add request for
   ** post phase **
Resource is CPU .
   logical CPU ID = 2
   Bind CPU ID = 2
*****Entered PostAcquire_cpu*****
---end of signal handler---

---Start of Signal Handler---
An add request for
   ** check phase **
Resource is CPU .
   logical CPU ID = 3
   Bind CPU ID = 3
*****Entered CheckedAcquire_cpu*****
---end of signal handler---

---Start of Signal Handler---
An add request for
   ** pre  phase **
Resource is CPU .
   logical CPU ID = 3
   Bind CPU ID = 3
```

```
*****Entered PreAcquire_cpu*****
---end of signal handler---

---Start of Signal Handler---
An add request for
   ** post phase **
Resource is CPU .
   logical CPU ID = 3
   Bind CPU ID = 3
*****Entered PostAcquire_cpu*****
---end of signal handler---
```

Sample output: 1 CPU removal

Example 3-11 shows the sample output of a 1 CPU removal DLPAR event. You will see there are line three blocks for the check, pre, and post phases for the CPU ID 3.

Example 3-11 Sample output: 1 CPU removal

```
---Start of Signal Handler---
A remove request for
   ** check phase **
Resource is CPU .
   logical CPU ID = 3
   Bind CPU ID = 3
*****Entered CheckRelease_cpu*****
---end of signal handler---

---Start of Signal Handler---
A remove request for
   ** pre phase **
Resource is CPU .
   logical CPU ID = 3
   Bind CPU ID = 3
*****Entered PreRelease_cpu*****
---end of signal handler---

---Start of Signal Handler---
A remove request for
   ** post phase **
Resource is CPU .
   logical CPU ID = 3
   Bind CPU ID = 3
*****Entered PostRelease_cpu*****
---end of signal handler---
```

3.9.4 DLPAR-aware kernel extensions

Like applications, most kernel extensions are DLPAR-safe by default. However, some are sensitive to the system configuration and might need to be registered with the kernel in order to be notified of DLPAR events.

To register and unregister from the kernel to be notified in the case of DLPAR events, the following kernel services are available:

- reconfig_register
- reconfig_unregister
- reconfig_complete

For further information about these kernel services, please refer to the following publications, available at:

`http://techsupport.services.ibm.com/server/library`

- *AIX 5L Version 5.2 General Programming Concepts: Writing and Debugging Programs*
- *AIX 5L Version 5.2 Technical Reference: Kernel and Subsystems*

3.10 Error handling of DLPAR operations

Knowing what errors the **drmgr** command may return is fundamental to creating a comprehensive DLPAR script or DLPAR-aware application. This section covers the methods AIX provides to help perform error analysis on failed DLPAR operations. We also discuss some actions that should be taken when an error occurs.

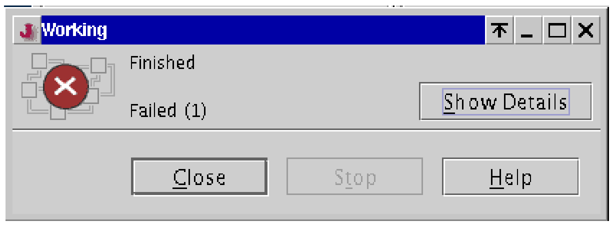
3.10.1 Possible causes of DLPAR operation failures

A DLPAR operation request can fail for various reasons. The most common of these is that the resource is busy, or that there are not enough system resources currently available to complete the request. In these cases, the resource is left in a normal state as if the DLPAR event never happened.

The following are possible causes of DLPAR operation failures:

- The primary cause of processor removal failure is processor bindings. The operating system cannot ignore processor bindings and carry on DLPAR operations or applications might not continue to operate properly. To ensure that this does not occur, release the binding, establish a new one, or terminate the application. The CPUs that are impacted is a function of the type of binding that is used.
- The primary cause of memory removal failure is that there is not enough pinned memory available in the system to complete the request. This is a system-level issue and is not necessarily the result of a specific application. If a page in the memory region to be removed has a pinned page, its contents must be migrated to another pinned page, while automatically maintaining its virtual to physical mappings. The failure occurs when there is not enough pinnable memory in the system to accommodate the migration of the pinned data in the region that is being removed. To ensure that this does not occur, lower the level of pinned memory in the system. This can be accomplished by destroying pinned shared memory segments, terminating programs that implement the plock system call, or removing the plock on the program.
- The primary cause of PCI slot removal failure is that the adapters in the slot are busy. Note that device dependencies are not tracked. For example, the device dependency might extend from a slot to one of the following: an adapter, a device, a volume group, a logical volume, a file system, or a file. In this case, resolve the dependencies manually by stopping the relevant applications, unmounting file systems, varying off volume groups, and unconfiguring the device drivers associated with the adapters in the target slot.

If an error occurs in a DLPAR operation, you will see the error message dialog box on the HMC, as shown in Figure 3-4.

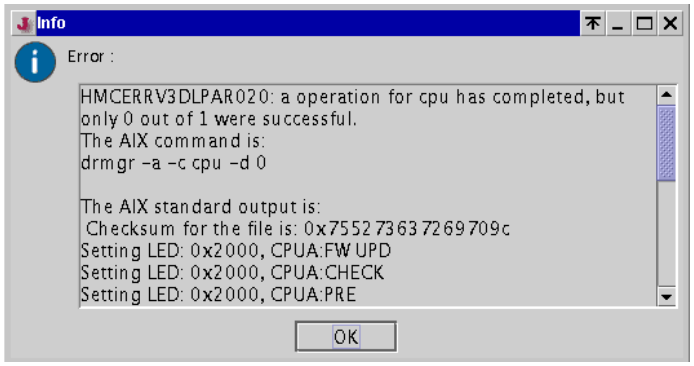


Finished

Failed (1)

Figure 3-4 DLPAR operation failed message

The HMC also displays the information message dialog box, as shown in Figure 3-5 on page 117.



```
HMCERRV3DLPAR020: a operation for cpu has completed, but
only 0 out of 1 were successful.
The AIX command is:
drmgr -a -c cpu -d 0

The AIX standard output is:
 Checksum for the file is: 0x7552 7363 7269 709c
Setting LED: 0x2000, CPUA:FW UPD
Setting LED: 0x2000, CPUA:CHECK
Setting LED: 0x2000, CPUA:PRE
```

Figure 3-5 DLPAR operation failure detailed information

Note: If the registered DLPAR scripts have bugs, they are also the cause of failure. You must carefully code the scripts and test them on a test partition before the deployment on the production partition.

3.10.2 Error analysis facilities

AIX provides the user the following facilities to help isolate DLPAR operation failures:

- Virtual operator panel
- The syslog facility
- AIX system trace facility
- AIX error log facility
- Kernel debugger (KDB)

These facilities can be also used in the event that a script or DLPAR-aware application fails. Moreover, by learning how to use these, you can modify your programs to automatically handle some of the possible errors.

Virtual operator panel

The virtual operator panel displayed on the HMC is used by AIX to display LED codes and character string messages (see Figure 5-7 on page 162). The values displayed on the virtual operator panel indicate several important actions that the kernel is performing while a DLPAR event is in progress. Table 3-14 on page 118 shows the possible values on the virtual operator panel.

Table 3-14 Virtual operator panel DLPAR operation indicators

Progress indicator code	Text string	Description
2000	CPUA	DLPAR CPU addition
2001	CPUR	DLPAR CPU removal
2002	MEMA	DLPAR memory addition
2003	MEMR	DLPAR memory removal

While a DLPAR operation is taking place, the appropriate code is displayed. These indicators are on the virtual operator panel for the duration of the operation. The virtual operator panel also indicates which phase it is in by appending the appropriate phase identifier, as shown in Table 3-15.

Table 3-15 Virtual operator panel phase indicators

Identifier	Description
:CHECK	The check phase
:PRE	The pre phase
:POST	The post phase

For example, if you see the following text string on the virtual operator panel, the partition is performing the post phase of a memory removal DLPAR event:

```
2003    MEMR:POST
```

In the case of DLPAR operations that use a large amount of memory, which could take a long time, this tells you that the operation is still in progress. Another useful instance is when a system failure occurs. Because the progress indicator is left on the panel when a system failure occurs, you can easily see that a DLPAR operation was in progress when the machine failed.

The syslog facility

The syslog facility is another useful tool to help isolate DLPAR-related errors. It can be used to keep a record of the progress of DLPAR events. The syslog entries come with a time stamp to indicate when all the DLPAR events occurred.

On AIX, the syslog facility is not enabled by default. To enable recording DLPAR events using syslog, do the following:

1. Edit the /etc/syslog.conf file as the root user.

2. Add the required syslog entries to the end of the file.

   For example, add the following:

   ```
   *.debug /var/adm/syslog.log rotate size 10k
   ```

   This directive line instructs the syslog facility to log all messages of priority debug (LOG_DEBUG) and above to the /var/adm/syslog.log file. The /var/adm/syslog.log file is automatically rotated to limit the maximum file size to 10 KB.
3. Create the file explicitly:

   ```
   # touch /var/adm/syslog.log
   ```

4. Restart the syslogd subsystem:

   ```
   # stopsrc -s syslogd
   # startsrc -s syslogd
   ```

In "Using the syslog facility" on page 450, we include the following syslog output examples:

- "CPU addition" on page 450
- "CPU removal" on page 451
- "Memory addition" on page 452
- "Memory removal" on page 453

When you register your DLPAR scripts, if you explicitly specify a channel ID string other than the default value DRMGR by using **`drmgr -S`**, you can quickly search the corresponding information that is produced by your DLPAR scripts. The default channel ID, DRMGR, is shown in several syslog output examples in "Using the syslog facility" on page 450.

AIX system trace facility

The AIX system trace facility is a tool that can trace many kernel internal activities by specifying trace hook IDs. In case of CPU- and memory-related DLPAR events, the trace hook ID is 38F. After capturing the trace of DLPAR events, you can generate a trace report in order to examine the results.

To use AIX system trace facility in order to capture DLPAR events, do the following as the root user:

1. Start the trace:

   ```
   # trace -a -j 38f
   ```

2. Perform the desired DLPAR operations.
3. Stop the trace:

   ```
   # trcstop
   ```

4. Analyze the trace:

   ```
   # trcrpt
   ```

You can also use SMIT to do the same activities (you need the root authority):

1. Invoke **smit** and select the following panels, and then press Enter:

   ```
   Problem Determination
       Trace
           START Trace
   ```

2. Type 38F in the ADDITIONAL event IDs to trace field, as shown in Example 3-12, and then press Enter.

Example 3-12 START Trace panel

```
                                   START Trace

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

                                                      [Entry Fields]
  EVENT GROUPS to trace                            []                      +
  ADDITIONAL event IDs to trace                    [38F]                   +
  Event Groups to EXCLUDE from trace               []                      +
  Event IDs to EXCLUDE from trace                  []                      +
  Trace MODE                                       [alternate]             +
  STOP when log file full?                         [no]                    +
  LOG FILE                                         [/var/adm/ras/trcfile]
  SAVE PREVIOUS log file?                          [no]                    +
  Omit PS/NM/LOCK HEADER to log file?              [yes]                   +
  Omit DATE-SYSTEM HEADER to log file?             [no]                    +
  Run in INTERACTIVE mode?                         [no]                    +
  Trace BUFFER SIZE in bytes                       [131072]                 #
  LOG FILE SIZE in bytes                           [1310720]                #
  Buffer Allocation                                [automatic]             +
```

3. Perform the desired DLPAR operations.
4. Invoke **smit** and select the following panels, and then press Enter:

   ```
   Problem Determination
       Trace
           STOP Trace
   ```

5. Invoke **smit**, select the following panels, select **1 filename (defaults stdout)**, and then press Enter:

   ```
   Problem Determination
       Trace
           Generate a Trace Report
   ```

6. Select the following values in the SMIT panel shown in Example 3-13, and then press Enter:

```
Show PROCESS IDs for each event?                    yes
Show THREAD IDs for each event?                     yes
```

Example 3-13 Generate a Trace Report panel

```
                           Generate a Trace Report

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

                                                       [Entry Fields]
  Show exec PATHNAMES for each event?               [yes]                   +
  Show PROCESS IDs for each event?                  [yes]                   +
  Show THREAD IDs for each event?                   [yes]                   +
  Show CURRENT SYSTEM CALL for each event?          [yes]                   +
  Time CALCULATIONS for report                      [elapsed only]          +
  Event Groups to INCLUDE in report                 []                      +
  IDs of events to INCLUDE in report                []                      +X
  Event Groups to EXCLUDE from report               []                      +
  ID's of events to EXCLUDE from report             []                      +X
  STARTING time                                     []
  ENDING time                                       []
  LOG FILE to create report from                    [/var/adm/ras/trcfile]
  FILE NAME for trace report (default is stdout)    []
```

In “Using the AIX system trace facility” on page 454, we include the following trace output examples:

- “CPU addition trace output” on page 454
- “CPU removal trace output” on page 457
- “Memory addition trace output” on page 461
- “Memory removal trace output” on page 464

AIX error log facility

The **drmgr** command can generate error log messages in the few cases involving kernel, kernel extensions, or platform failures that have been caused by a DLPAR event. Table 3-16 on page 122 shows a list of the possible errors that could be found in the system error log.

Table 3-16 AIX error logs generated by DLPAR operations

Error log entry	Description
DR_SCRIPT_MSG	Application script error or related messages, or both. Entry incudes failing script name and DLPAR phase where the error occurred.
DR_RECONFIG_HANDLER_MSG	Kernel extension reconfiguration handler error. Entry includes failing handler's registration name.
DR_MEM_UNSAFE_USE	Non-DLPAR aware kernel extension's use of physical memory is not valid. The result is that the affected memory is not available for DLPAR removal. Entry includes: ► The affected logical memory address ► An address corresponding to the kernel extension's load module ► The kernel extension load module's path name
DR_DMA_MEM_MIGRATE_FAIL	Memory removal failure due to DMA activity. The affected LMB had active DMA mappings that could not be migrated by the platform. The entry includes: ► The logical memory address within the LMB ► Hypervisor migration return code ► Logical bus number of the slot owning the DMA mapping ► The DMA address
DR_DMA_MEM_MAPPER_FAIL	Memory removal failure due to a kernel extension responsible for controlling DMA mappings error. The entry includes: ► DMA mapper handler return code ► An address corresponding to the DMA mapper's kernel extension load module ► The DMA mapper's kernel extension load module's path name

Kernel debugger (KDB)

The AIX kernel debugger (KDB) helps isolate DLPAR operation errors. KDB can be especially useful to diagnose errors found in the kernel extensions. For further information about the use of KDB, please refer to *AIX 5L Version 5.2 Technical Reference: Kernel and Subsystems*, available at:

`http://techsupport.services.ibm.com/server/library`

3.10.3 AIX error log messages when DLPAR operations fail

Several different AIX error messages may be generated when a DLPAR event failure occurs. We list these error messages and the action that should be taken. The following tables show all the AIX error messages in relation to DLPAR operations.

Table 3-17 indicates general error messages that may be displayed when a DLPAR event failure occurs and the recommended actions to take.

Table 3-17 General AIX error messages

Error message	Recommended action
`You must have root authority to run this command.`	Log in as the root user.
`Failed to set the ODM data.`	Contact an IBM service representative.
`Consult AIX error log for more information.`	Open the AIX error log and look for error log entries with the DR_ prefix.
`Resource identifier out of range.`	Consult AIX syslog and HMC logs.

Table 3-18 describes the possible errors that can be generated by the **drmgr** command.

Table 3-18 drmgr-specific AIX error messages

Error message	Recommended action
`Error building the DLPAR script information.`	Check system resources, such as free space in the /var file system. If the problem persists, contact an IBM service representative.
`Aborting DLPAR operation due to Check Phase failure.`	Examine the AIX syslog. Contact the script/application owner.

Error message	Recommended action
`Error: Specified DLPAR script file already exists in the destination directory.`	Examine the AIX syslog. Contact the script/application owner to change the script name. Use the force flag (**`drmgr -f`**) to overwrite the pre-existing script.
`Error: Specified DLPAR script file does not exist in directory.`	File could not be found for uninstallation. Check the file name specified.
`The DLPAR operation is not supported.`	The machine or configuration does not support that operation. Upgrade the system firmware or operating system software, or both.
`Invalid parameter.`	Contact an IBM service representative.

While a DLPAR operation is taking place, an error may occur. Table 3-19 indicates some of these error messages caused by AIX during a DLPAR operation.

Table 3-19 DLPAR operation-specific AIX error messages

Error message	Recommended action
`DLPAR operation failed because of timeout.`	Increase the timeout value, or try again later. Also, try the DLPAR operation without a timeout specified.
`DLPAR operation failed. Kernel busy with another DLPAR operation.`	Only perform one DLPAR operation at a time. Try again later.
`DLPAR operation failed due to kernel error.`	Examine the AIX syslog or contact an IBM service representative, or both.
`The DLPAR operation could not be supported by one or more kernel extensions.`	Find the corresponding AIX error log entry DR_RECONFIG_HANDLER_MSG and contact the kernel extension owner.
`DLPAR operation failed since resource is already online.`	Examine the AIX syslog and HMC log.
`DLPAR operation timed out.`	Increase the timeout, or try again later. Also, try initiating the DLPAR operation without a timeout specified.

Finally, there are several DLPAR errors resulting from resource events. Table 3-20 on page 125 displays these types of errors.

Table 3-20 DLPAR resource-specific AIX error messages

Error message	**Recommended action**
`The specified connector type is invalid, or there is no dynamic reconfiguration support for connectors of this type on this system.`	Examine the AIX syslog and HMC log. Valid connector types are listed in Table 3-10 on page 81.
`Insufficient resource to complete operation.`	Try again later. Free up resources and try again.
`CPU could not be started.`	Examine the AIX syslog and HMC log.
`Memory could not be released. DLPAR operation failed since a kernel extension controlling DMA mappings could not support the operation.`	Examine the AIX error log and look for the DR_DMA_MAPPER_FAIL entry. The logical memory block or address of the message should correspond to the logical memory address in the error log entry. The LR value in the error log is an address within the failing kernel extension. The Module Name in the error log is the path name of the kernel extension load module. Unconfigure the kernel extension and retry. Contact the kernel extension owner.
`Resource could not be found for the DLPAR operation.`	Examine the AIX syslog and HMC log.
`Resource is busy and cannot be released.`	Examine the AIX syslog. Quiesce activity using the resource and try again.
`Memory in use by a non DLPAR-safe kernel extension and hence cannot be released.`	Examine the AIX error log and look for the DR_MEM_UNSAFE_USE entry. The logical memory block or address in the message should correspond to the logical memory address in the error log. The LR value in the error log is an address within the owning kernel extension. The Module Name in the error log is the path name of the kernel extension load module. Unconfigure the kernel extension and retry. Contact the kernel extension owner.
`Memory could not be released because system does not contain enough resources for optimal pinned memory and large page memory ratios.`	Reduce pinned or large page memory requirements, or both, and try again.

In "Using the AIX error log facility" on page 467, we provided several AIX error log output examples.

Part 2

Systems Management

4

HMC graphical user interface

This chapter describes the HMC graphical user interface in the following sections:

- "Login and logout" on page 130
- "HMC graphical user interface at a glance" on page 130
- "HMC application overview" on page 138
- "Server and Partition" on page 139
- "Virtual terminal window" on page 146
- "Open xterm to access remote system using telnet" on page 149

Before proceeding to later chapters, you should be familiar with the terms and concepts used in the HMC graphical user interface, as explained in this chapter.

4.1 Login and logout

After power-on, the HMC shows the graphical login panel prompting for the user ID and the password. The HMC is supplied with a predefined user ID `hscroot` and the default password `abc123`. Both the user ID and password are case sensitive and must be typed exactly as shown. After the successful login, the HMC graphical user interface opens, as shown in Figure 4-1 on page 131.

To log out from the HMC graphical user interface, do the following:

1. From the menu bar, select **Console** → **Exit**.

   At this point, you can choose to save the state of the console for the next session by selecting the check box next to the option.

2. Select **Exit Now**.
3. When you exit from your HMC session, you have to choose from the following three logout modes:[1]

Shutdown Console	Powers off the HMC system.
Reboot Console	Shuts down the HMC system and then reboots it to the login prompt.
Logout	Returns the user to the login prompt without shutting down the HMC system.

   In either mode, the managed systems are not affected by these operations.

4.2 HMC graphical user interface at a glance

The HMC graphical user interface has the same appearance, key concepts, and basic tasks and tools as the AIX 5L Version 5.2 Web-based System Manager. For further information about the Web-based System Manager, refer to *AIX 5L Version 5.2 System Management Guide: AIX 5L Version 5.2 Web-based System Manager Administration Guide*, available at:

`http://techsupport.services.ibm.com/server/library`

[1] You can also use the **`hmcshutdown`** command to shut down or reboot your HMC.

The HMC graphical user interface is composed of several elements, as shown in Figure 4-1.

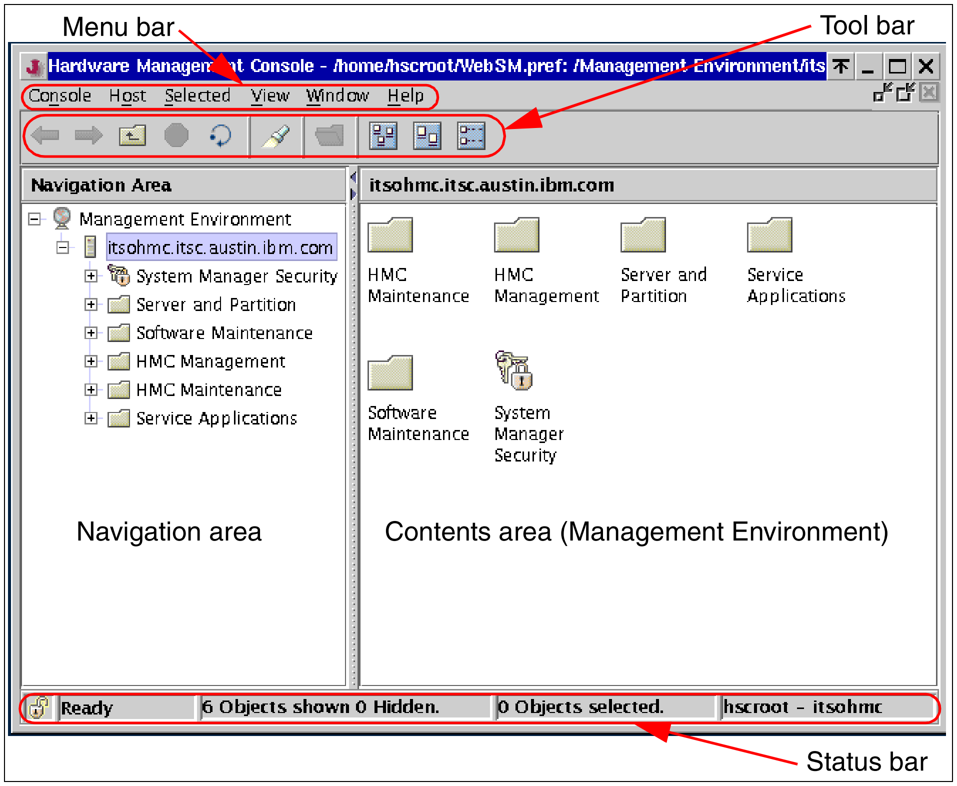


Figure 4-1 HMC graphical user interface

Table 4-1 shows the relevant section number for each element indicated in Figure 4-1.

Table 4-1 Elements in the HMC graphical user interface

Element	**Relevant section number**
Navigation area	4.2.1
Contents area	4.2.2
Menu bar	4.2.3
Tool bar	4.2.4
Status bar	4.2.5

4.2.1 Navigation area

The left side of the HMC graphical user interface is the Navigation area. It displays a hierarchy of items ordered in a tree structure. The root of the tree is the Management Environment. It contains the name of the HMC that you are currently logged in to. For example, you can see the icon with the host name itsohmc.itsc.austin.ibm.com in the Navigation area in Figure 4-1 on page 131. It is the host name of the HMC from which this panel image has been taken. In this example, there is only one host system, the HMC itsohmc.itsc.austin.ibm.com.

The Management Environment is a set of host systems that can be managed from the HMC. The host systems can be the HMC into which you are currently logged, the other remote HMCs, and also AIX systems managed by their Web-based System Manager interface.

To add a host system under the Management Environment, do the following:

1. From the menu bar, select **Console** → **Add** → **Hosts**.

   You have two options here: you can add a single host or multiple hosts. For a single host, add the host name of the system that you want to add; for multiple hosts, provide the path name of the file that contains the hosts to be added.

   You also have the option to verify whether the hosts added are on the network by selecting the option provided to you.

To remove a host system under the Management Environment, do the following[2]:

1. From the menu bar, select **Console** → **Remove**.
2. Select the host name from the displayed list that you want to remove, and then confirm in the next panel that you want to remove the designated host.

Note: The managed system itself never appears in the Navigation area unless you manage AIX instances running in partitions managed by the Web-based System Manager.

Every folder contains different HMC applications used in the specific management task, such as Server and Partition or Software Maintenance, as shown in the Navigation area in Figure 4-1 on page 131. If you choose one of these HMC applications, it provides its own submenus and objects in the Contents area determined by the application context.

[2] This remove operation does not affect the managed system deleted from the HMC application.

4.2.2 Contents area

The right side of the panel is the Contents area. It displays managed objects and related tasks. You can choose different views in the Contents area: large icons, small icons, or details in the form of a list.

Note: The label of the Contents area is changed depending on the application context. For example, if you select **Management Environment** in the Navigation area, the label is changed to Management Environment, as shown in Figure 4-1 on page 131.

4.2.3 Menu bar

The following six menu items are provided in the menu bar:

Console The Console menu contains choices that control the console. It enables you to add and remove managed systems, other HMCs, or other AIX systems managed by Web-based System Manager from the management environment. It also enables you to change themes on the desktop, change font sizes, open an outbound Telnet terminal session using an IP address or a host name, and exit the console.

Object The title of the Object menu changes to indicate the type of resource managed by the current HMC application. For example, when the Server Management application is selected, the Object menu title becomes Server Management. The Object menu contains general choices and actions for a HMC application that do not require the selection of specific objects to act on. The find function is also located in the Object menu. The contents of the Object menu are updated when a new HMC application is selected. In the case where you are managing an AIX system remotely, the AIX 5L Version 5.2 Web-based System Manager applications appear here.

Selected The Selected menu contains the set of actions that are applicable to the object selected in the Contents pane. The contents of the Selected menu are updated based on which object you select. It is disabled when Overview and Launch applications are loaded. The open tab in the Selected menu expands the view of a managed system in the Navigation area.

View The View menu contains choices for navigating, such as Back, Forward, and Up One Level. It also includes choices for customizing the console in the Show submenu. For example, you can select to show or hide the tool bar and status bar. This menu

	also includes options that control how objects are presented. For example, if the Contents area content provides a choice of views, such as Large Icon, Small Icon, Details, and Tree, these choices are listed here. If the content has only a single view, no view choices are listed. When the content displays an icon or Details view, the View menu includes choices for sorting and filtering the container.
Window	The Window menu contains actions for managing subpanels in the console workspace. The new virtual terminal creates a new console subpanel in the workspace. Other choices control how all console subpanels are placed. For example, you can choose to have the panels completely cover the workspace-like tiles, or have them stacked in a cascade style.
Help	The Help menu lists user assistance choices. Different options enable you to view help contents, search for help on a particular topic, and view help information about shortcut keys.

4.2.4 Tool bar

The tool bar lists commonly used actions that are available when the current plug-in application is loaded. It includes navigation controls, Find and View choices (if available), and a refresh option of the HMC graphical user interface. The tool bar also provides tool tip help when the pointer remains over a tool bar icon for a few seconds.

Reload button

The HMC graphical user interface provides the Reload button in the tool bar shown in Figure 4-2.[3] If the HMC does not display the operation task result correctly, you can click this button to reload the latest information.

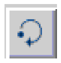


Figure 4-2 Reload button

Details, Tree, Tree-Details buttons

The HMC graphical user interface provides the Details, Tree, and Tree-Details buttons in the tool bar, shown in Figure 4-3.[4] Once one of these buttons is selected, the selected view is preserved across the power recycle of HMC.

[3] The function can be also be selected from **View** → **Reload** or pressing the F5 key.
[4] Same functions are available in the **View** menu.

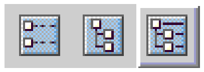
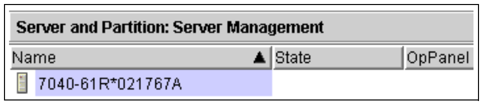


Figure 4-3   Details, Tree, Tree-Details buttons

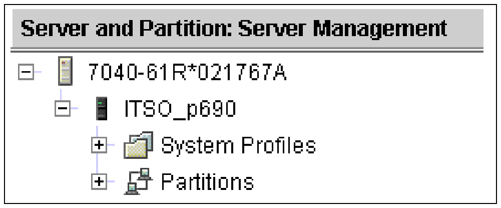
For example, if you click the Details button while a managed system is selected in the Server Management application, the Contents area will show the detailed information about the selected managed system itself, as shown in Figure 4-4.


Server and Partition: Server Management

Name ▲	State	OpPanel
7040-61R*021767A		

Figure 4-4   Detailed view

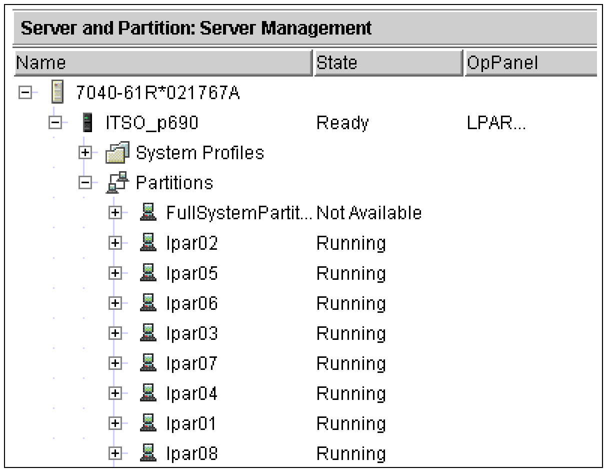
If you click the Tree button, the Contents area will show a tree that represents objects belonging to the managed system, as shown in Figure 4-5.


Server and Partition: Server Management

```
⊟ 7040-61R*021767A
   ⊟ ITSO_p690
      ⊞ System Profiles
      ⊞ Partitions
```

Figure 4-5   Tree view

If you click the Tree-Details button, the Contents area will show a tree that represents objects belonging to the managed system as well as the detailed information for each object as shown in Figure 4-6 on page 136.


Server and Partition: Server Management

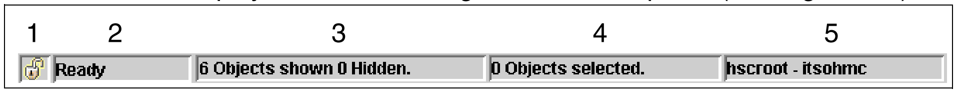
Name	State	OpPanel
7040-61R*021767A		
ITSO_p690	Ready	LPAR...
System Profiles		
Partitions		
FullSystemPartit...	Not Available	
lpar02	Running	
lpar05	Running	
lpar06	Running	
lpar03	Running	
lpar07	Running	
lpar04	Running	
lpar01	Running	
lpar08	Running	

Figure 4-6 Tree-Details view

Note: We recommend that you select this view while you are managing partitions.

4.2.5 Status bar

The status bar displays at the lower edge of a console panel (see Figure 4-7).



Ready | 6 Objects shown 0 Hidden. | 0 Objects selected. | hscroot - itsohmc

Figure 4-7 Status bar

It has the following five fields ordered from left to right for displaying status information:

1. Padlock icon

   The padlock icon is open when secure communications are not active. When locked, the padlock icon indicates that the Web-based System Manager client on the HMC is running in secure mode. In this case, the communication between the Web-based System Manager client on the HMC and the connected Web-based System Manager server on the other system is encrypted using Secure Sockets Layer (SSL). A Web-based System Manager server is always running on the HMC itself, and can be running on the following remote systems:

   – The other remote HMCs
   – AIX systems (including on partitions)

The padlock icon indicates whether the Web-based System Manager client on the HMC is running in secure mode. It does not indicate whether the Web-based System Manager server on the HMC is running in secure mode. Therefore, while you are logging in to your HMC from its local console, the padlock icon is locked only when:

  - You are accessing to other manageable systems, including the other HMCs.
  - A public key ring file that was generated on the remote system has been already copied onto the local HMC.[5]

2. Plug-in loading status

   When a plug-in application is loaded, the text `Ready` is present. When an application is in the process of loading, a graphic bounce bar is displayed.

3. Number of objects visible in the Contents area

   Objects can be present on the managed system but hidden from the view by the view filter.

4. Number of objects selected in the Contents area
5. Security context (user name and host name)

   This displays the administrator user name and the HMC host name for the currently active HMC.

   The status bar can be hidden or shown by clearing or checking the Status Bar option in the Show submenu under View.

The HMC also provides a pop-up menu (it is also called context menu) for quick access to menu choices. To use pop-up menus with a mouse, point to an object, and then right-click. The pop-up menu lists the actions found in the Selected and Object menus for the current object or objects.

Reset the current HMC graphical user interface session

If your HMC graphical user interface session hangs, it means that even if you wait 10 minutes after an operation, and the pointer is still a clock-shaped icon, you can restart the X server to reset the session. You can reset your hung session by pressing the Ctrl+Alt+Backspace key combination. The X server restarts and displays the HMC login prompt. All messages and panels regarding the hung session will be lost.

[5] This operation is performed by the “Copy another Certificate Authority’s Public Key Ring File from diskette” task.

4.3 HMC application overview

As shown in Figure 4-8, seven application folders are provided in the Navigation area in the HMC graphical user interface.

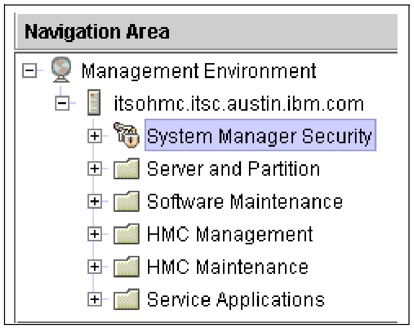


Figure 4-8 HMC application folders in the Navigation area

These folders contain several applications to be used for different system management tasks on the HMC and managed systems as shown in Table 4-2.

Table 4-2 HMC application folders

Folder name	**Description**
System Manager Security	This folder contains several applications that enable a secure network connection from other Web-based System Manager clients for the remote control of an HMC in client/server mode.
Server and Partition	This folder contains only one application, Server Management, which provides all partition-related tasks. It is used to create, maintain, activate, and delete logical partitions and affinity partitions.
Software Maintenance	This folder contains three applications (Frame, HMC, and Software Maintenance) that enable you to perform software level management tasks on a frame as well as on an HMC. The available tasks for the HMC are: save and back up important HMC-related information, format removable media, save upgrade data, and install corrective fixes.
HMC Management	This folder contains only one application, Users, which controls user access to the HMC and enables the user to perform different tasks in the HMC environment depending on the different roles assigned to each user you create.
HMC Maintenance	This folder contains several applications that enable you to set the console's date and time, modify and view HMC network information, view console events, and schedule routine backups. It is also used to enable and disable remote command execution and secure shell access, change the language locale, and configure the serial adapter.

Folder name	Description
Service Applications	This folder contains several applications to be used for service-related tasks, such as Inventory Scout, Service Agent, and Service Focal Point.

In this redbook, we focus on the Server and Partition folder and the Server Management application under the folder. The following section explains how the objects are represented in the application. See Chapter 6, “Creating and managing partitions” on page 165 for the application usage.

4.4 Server and Partition

The Server and Partition folder contains only the Server Management application, which provides all partition-related tasks.

It is important to understand how to select the managed system that you are going to manage using the Server Management application. If this application is selected, you will see the object hierarchy illustrated in Figure 4-9 in the content area.

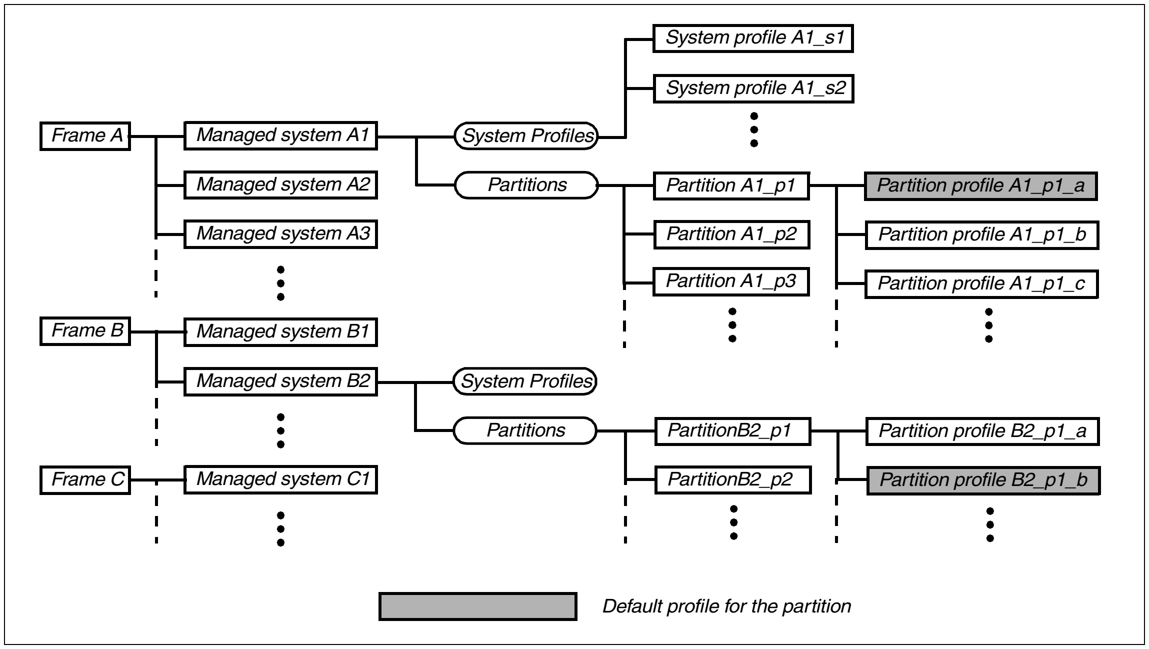


Figure 4-9 Object hierarchy for the Server Management application

The object hierarchy is summarized as follows:

- Multiple frames can exist in the content area.
- A frame can contain multiple managed systems.
- A partitioning-capable managed system always has two branch nodes:
  - System Profiles
  - Partitions
- The System Profiles branch node can contain multiple system profiles.
- The Partitions branch node can contain multiple partitions.
- A partition can contain multiple partition profiles.
- One of the partition profiles is designated as the default profile for the partition; if a partition has only one partition profile, that profile is always treated as the default partition profile.

For example, a frame icon (7040-61R*021767A) is shown in the content area in Figure 4-10 on page 141, where 7040-61R is the machine type and model for the 24” system frame for pSeries 690, pSeries 670, and pSeries 655. In this frame, there is only one managed system ITSO_p690 is shown, which has two branch nodes: System Profiles and Partitions.

Under the Partitions branch node, nine partitions are shown, including the Full System Partition. While the lpar02 partition has two partition profiles, lpar01 has only one; therefore aix51_64 is the default profile for lpar01.

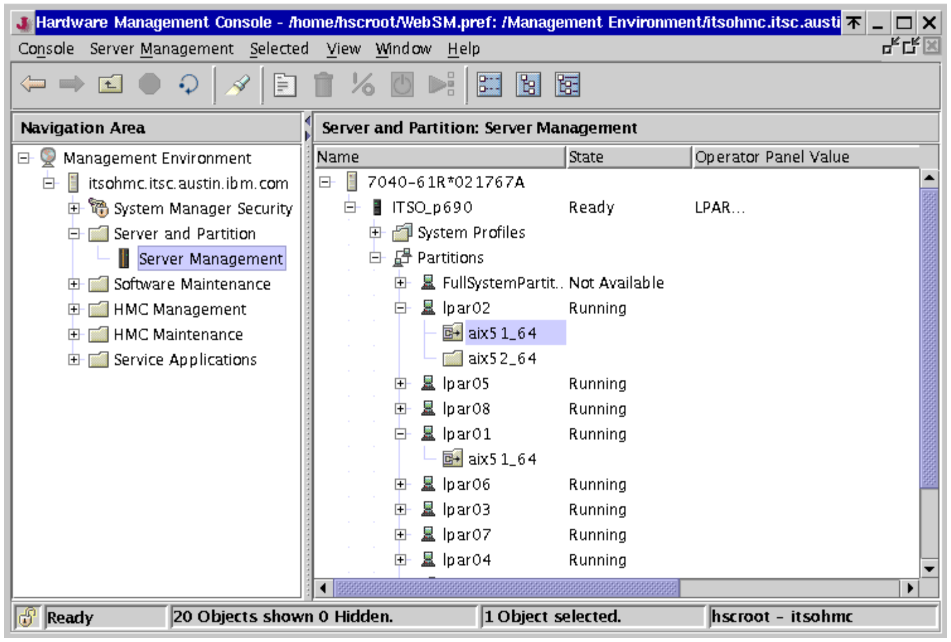
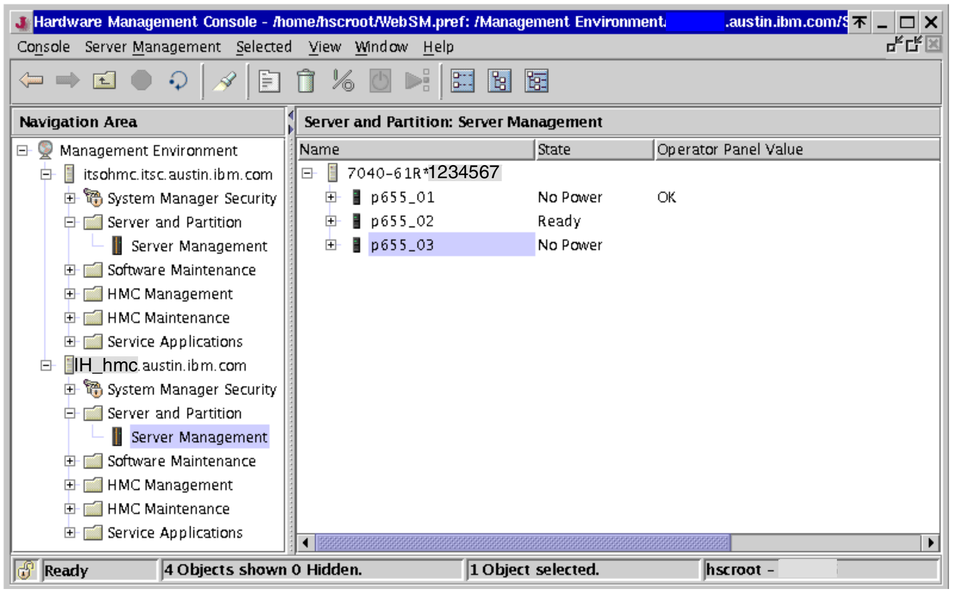


Figure 4-10 Server Management (one managed system)

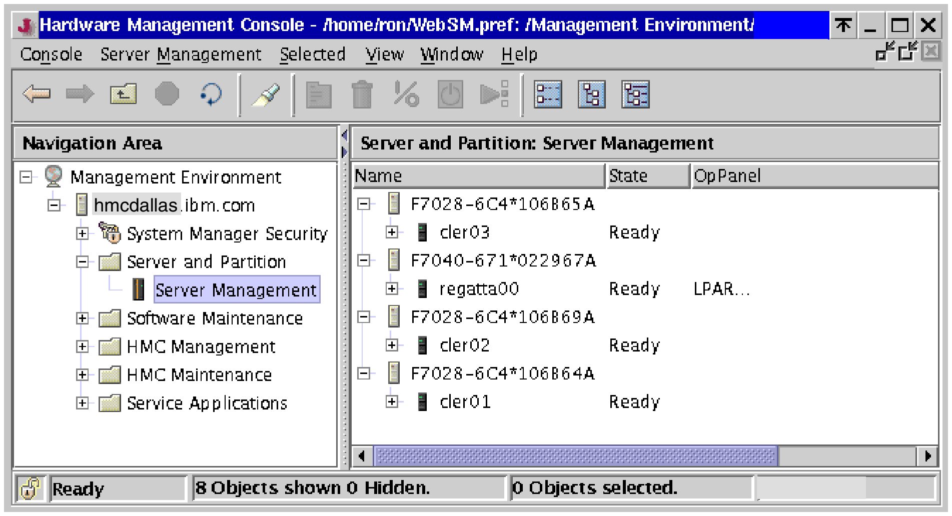
We show somewhat more complex examples for the graphical user interface on the HMC that manages multiple managed systems:

- Figure 4-11 on page 142 shows a frame (7040-61R*1234567) in the Content area that contains three managed systems (three pSeries 655 servers).
- Figure 4-12 on page 142 shows four frames in the Content area. The first frame contains a pSeries 670, whereas the other frames contain a pSeries 630 Model 6C4 each.



Name	State	Operator Panel Value
7040-61R*1234567		
p655_01	No Power	OK
p655_02	Ready	
p655_03	No Power	

Figure 4-11   Server Management (three pSeries 655 servers)[6]



Name	State	OpPanel
F7028-6C4*106B65A		
cler03	Ready	
F7040-671*022967A		
regatta00	Ready	LPAR...
F7028-6C4*106B69A		
cler02	Ready	
F7028-6C4*106B64A		
cler01	Ready	

Figure 4-12   Server Management (four managed systems)[7]

[6] This screen shot is taken from an HMC that manages three pSeries 655 nodes being used for the internal test purpose. Therefore, the HMC host name is purposely hidden.
[7] The frame icon descriptions start with F, not from MT-MDL, in Figure 4-12, since those systems are installed with V3.0 system firmware.

Note:

- A single 7040-61R frame can accommodate only one pSeries 670 or pSeries 690 server.
- A single 7040-61R frame can accommodate multiple pSeries 655 servers.
- Although multiple pSeries 650 Model 6M2 servers, and pSeries 630 Models 6C4 and 6E4 servers can be physically accommodated in a single 19-inch rack, those servers always appear in separate frames in the Content area of Server Management.

4.4.1 Connect and disconnect managed systems

You can connect or disconnect managed systems on the HMC using the following procedures.

Connect to the managed systems

The first time you connect a managed system to the HMC, a predefined name of the managed system appears in the Contents area of Partition Management. You can change this name by selecting the managed system in the Contents area, selecting the **Select** option in the menu bar, and clicking the **Properties** menu option (see Figure 5-1 on page 153).

Disconnect from the managed systems

You can delete managed systems from the HMC graphical user interface if you no longer want to manage a particular system.

Note: Do not physically disconnect the serial connection before performing the procedures explained here.

To delete the managed system from the Contents area, do the following:

1. In the Contents area, select the managed system.
2. From the menu bar, choose **Selected** → **Delete**.
3. Click **Yes** to delete the managed system from the Contents area.
4. Disconnect the serial cable from the managed system.

4.4.2 Server Management

The Server Management application is used to create, maintain, activate, and delete logical partitions and affinity partitions. It is also used to power on and power off the managed system and partitions, open and close virtual terminal windows for the partitions, view properties of the managed system, perform backups, restore profile data, and rebuild the managed system.

Note: You can create, view, and remove partitions (including affinity partitions), system profiles, and partition profiles using the **`mksyscfg`**, **`lssyscfg`**, and **`rmsyscfg`** commands. The **`chsyscfg`** command can also be used to modify the already created objects.

4.4.3 Server Management menus

As shown in Figure 4-13, the following 10 menus are available in this application if you select a managed system.

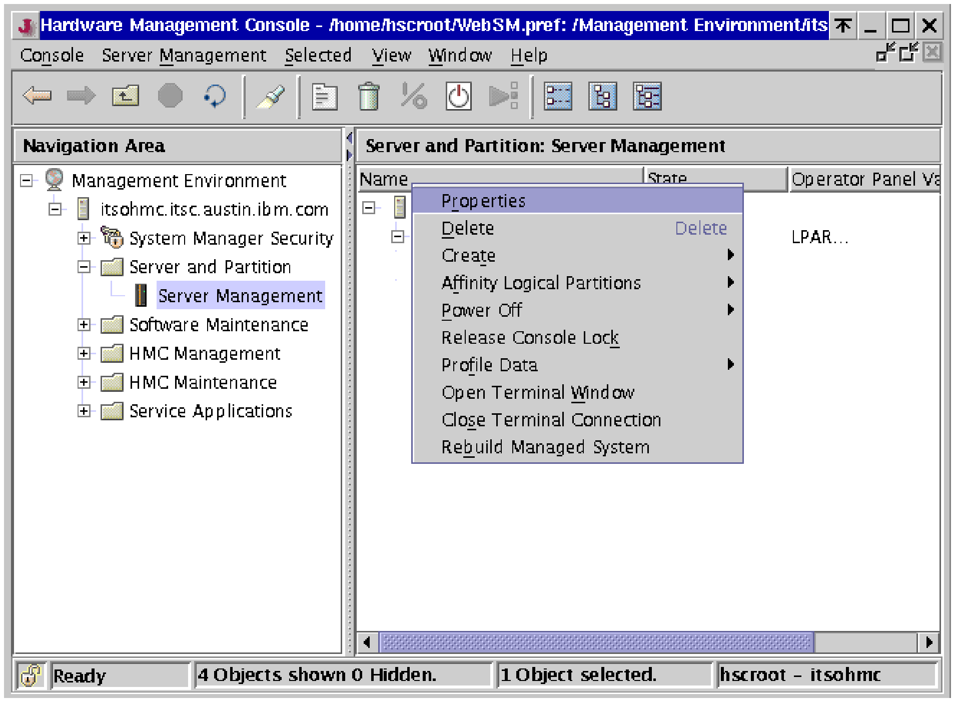


Figure 4-13 Server Management options

Properties

This menu enables you to see the properties of the managed system. The application queries several attributes and capabilities and displays them in the machine, processors, memory, I/O slot, and policy attributes in the property window (see 5.1, “Viewing properties of the managed system” on page 152).

Delete

This menu enables a user with System Administrator role, such as hscroot, to delete a selected managed system that is controlled from this HMC.

Create

This menu enables a user with System Administrator role, for example hscroot, to create logical partitions or system profiles. The system profile option is dimmed on a system that has no logical partition profiles defined.

Affinity Logical Partitions

This menu is used to set up partitions that have a predefined affinity for processors and memory. You can set up these affinity logical partitions with either a four-way processor MCM configuration or with an eight-way processor MCM configuration. The application setup wizard automatically defines the number of affinity partitions that can be defined based on the systems processor configuration. Affinity partitions cannot run with normal partitions on a single managed system at the same time.

Note: This menu is only available on the pSeries 670 and pSeries 690.

Power On and Off

This menu enables the user to toggle the power states of the managed system. Only one option is available based on the state of the managed system. If the state is *powered on*, the *Power Off* option is available, and if the state is *powered off*, the *Power On* option is available.

We explain this menu in 5.2, “Power on the managed system” on page 157 and 5.3, “Power off the managed system” on page 161.

Release Console Lock

This menu provides a way to manually override the HMC operations lock held in the service processor on a managed system, which coordinates activities between two HMCs.

To release an HMC lock, do the following:[8]

1. In the Contents area, select the managed system.
2. From the menu bar, choose **Selected** → **Release Console Lock**.

Note: This menu should normally only be needed if there have been HMC failures that left the lock on.

[8] The `rmsplock` command can also be used to remove leftover locks.

Profile Data

This menu enables the hscroot user to restore, initialize, back up, and remove profile data.

Open Terminal Window

This menu enables the opening of a virtual terminal window to the partition. This connection is necessary to define the default console and the network interface for the partition when it is created.

Close Terminal Connection

This menu enables the closing of the virtual terminal window to the partition.

Note: Clicking the X at the top-right of the opened virtual terminal window is not enough to close the terminal connection. You must explicitly select this menu in order to close the opened virtual terminal connection.

Rebuild managed system

This menu instructs the HMC to retrieve the information from the NVRAM in the managed system and then refresh the graphical user interface using the retrieved information (the **`chsysstate`** command can also be used to rebuild the managed system.

4.5 Virtual terminal window

AIX needs a console for installation and some service activities. The native serial ports on the managed system are only assignable together to one partition. The virtual terminal window provides virtual terminal console access to every partition without a physical device assigned.

4.5.1 Virtual terminal window concept

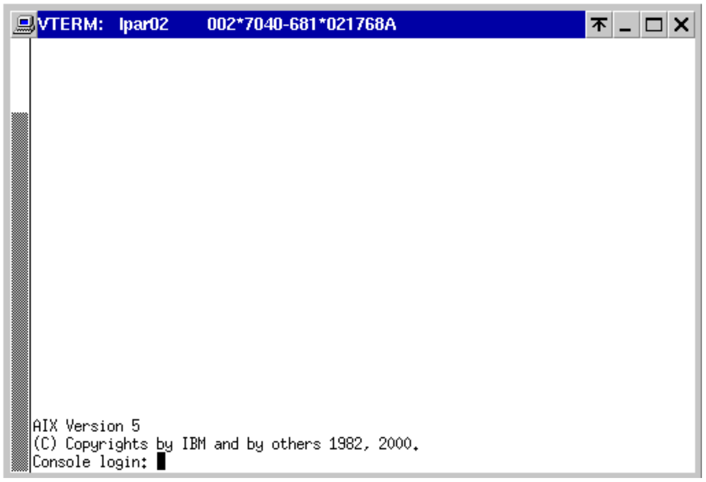
A virtual terminal window is available for each partition or Full System Partition of the managed system. Some functions are limited, and the performance cannot be guaranteed because of the limited bandwidth of the serial connection between the HMC and the managed system.

To open the virtual terminal window, do the following (the **`mkvterm`** and **`rmvterm`** commands can also be used to open and close a virtual terminal to the specified partition):

1. Expand the **System and Partition** folder in the navigation area.
2. Select the Server Management application.

3. Select the frame in which the target managed system resides.
4. Select the managed system on which the target partition is running.
5. Expand the **Partitions** tree.
6. Select the target partition and right-click on it.
7. Select the **Open Terminal** menu.

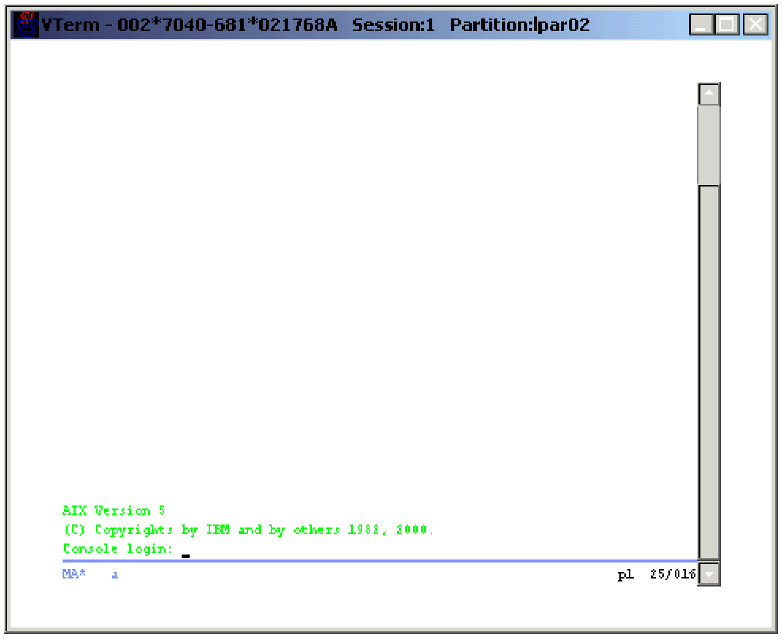
If you have done this operation on the HMC local console, you will see the virtual terminal window shown in Figure 4-14.



```
AIX Version 5
(C) Copyrights by IBM and by others 1982, 2000.
Console login: █
```

Figure 4-14 Virtual terminal window on the HMC

If you have done this operation on the remote Web-based System Manager client, you will see the virtual terminal window shown in Figure 4-15 on page 148.



```
AIX Version 5
(C) Copyrights by IBM and by others 1982, 2000.
Console login: _
MA*   1                                              pl  25/016
```

Figure 4-15 Virtual terminal window on the remote WebSM client

Note: In Figure 4-14 on page 147 and Figure 4-15, the title bar displays the machine type and model name (7040-681), the serial number of the pSeries 690 (021768A), and the partition name (lpar02) to which the virtual terminal window is connected.

The virtual terminal window should only be used for installation and service purposes. For AIX configuration and management, we recommend you use a network adapter assigned to the partition exclusively. The virtual terminal window does not support the following:

- Printing to a virtual terminal
- Transparent print services
- Modem connection for the virtual console port
- Real-time applications

The virtual terminal window supports the AIX **`smitty`** and other *curses-driven* applications. The virtual terminal window emulates a VT320 terminal. To set the terminal type on a virtual terminal window session, you can use the AIX **`export TERM=vt320`** command on the Korn shell prompt.

The following operations are available for the virtual terminal window:

- Open a virtual terminal window

  From the HMC graphical user interface, select a partition or the Full System Partition in the Contents area using Partition Management, right-click it, and select **Open Terminal Window**.

- Close a virtual terminal window

  To close a virtual terminal window, click the X in the top-right corner of the panel. To force a virtual terminal window to close, select the partition, right-click it, and then select **Close Terminal Connection**.

4.5.2 Virtual terminal window in the Full System Partition

When you open a virtual terminal window to the Full System Partition, the output of the native S1 serial port is redirected to the virtual terminal window. Then, the output of any command is directed from the serial port S1 to the virtual terminal window. After closing the virtual terminal window, the serial port S1 is normally accessible.

In the No Power state, you can access the service processor of a managed system with a virtual terminal window.

4.5.3 Partition virtual terminal windows

You can open a virtual terminal window at any time, regardless of the state of a partition, but only one per partition. The virtual terminal window is blank until the partition is activated. After you activate one of partitions, you cannot connect a virtual terminal window to the service processor of the managed system. In a partitioned environment, the native serial port S1 is not redirected to the virtual terminal of that partition.

4.6 Open xterm to access remote system using telnet

You can open xterm windows to connect to the other hosts (except for the HMC itself) using `telnet` over the network in order to access the other hosts, including partitions.

To use this function, do the following from the HMC graphical user interface:

1. From the menu bar, select **Console** → **Open Terminal**.
2. Enter the host name or the IP address, then click **OK**.

To access a partition using this function, the partition has to be assigned at least one network adapter, and the adapter has to be configured with an IP address that can be accessible from the HMC.

Note: The **Open Terminal** menu is available only on the local HMC console.

5

Basic managed system operation tasks

This chapter explains in the following sections how to start, stop, and reset an operating system on a managed system in both a partitioned environment and Full System Partition using the Server Management application:

- “Viewing properties of the managed system” on page 152
- “Power on the managed system” on page 157
- “Power off the managed system” on page 161
- “Operating the managed system with the HMC” on page 161

In addition, the Contents area of the Server Management application provides status information about the managed system and the partitions and displays the operator panel value of the managed system and the partitions.

5.1 Viewing properties of the managed system

To view the properties of your managed system, select the managed system in the Contents area, and from the menu bar choose **Selected** → **Properties**. Or select the managed system in the Contents area, right-click, and select **Properties**. The property panel shown in Figure 5-1 on page 153 opens.

The properties panel includes the five property tabs of the managed system shown in Table 5-1.

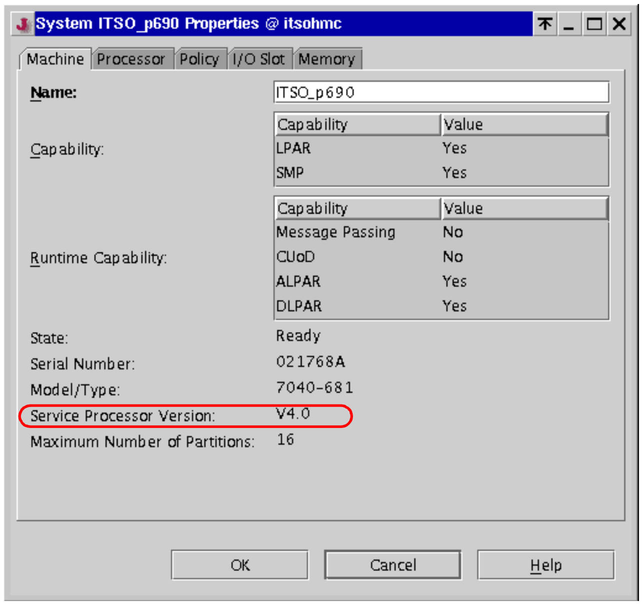
Table 5-1 Properties of the managed system

Property name	Figure number
Machine	Figure 5-1 on page 153
Processor	Figure 5-2 on page 154
Policy	Figure 5-3 on page 155
I/O Slot	Figure 5-4 on page 156
Memory	Figure 5-5 on page 157

5.1.1 Machine property

The Machine property tab displays the following information, as shown in Figure 5-1:

- Capability
- Runtime Capability
- State
- Serial Number
- Model/Type
- Service Processor Version



Name: ITSO_p690

Capability:

Capability	Value
LPAR	Yes
SMP	Yes

Runtime Capability:

Capability	Value
Message Passing	No
CUoD	No
ALPAR	Yes
DLPAR	Yes

State:	Ready
Serial Number:	021768A
Model/Type:	7040-681
Service Processor Version:	V4.0
Maximum Number of Partitions:	16

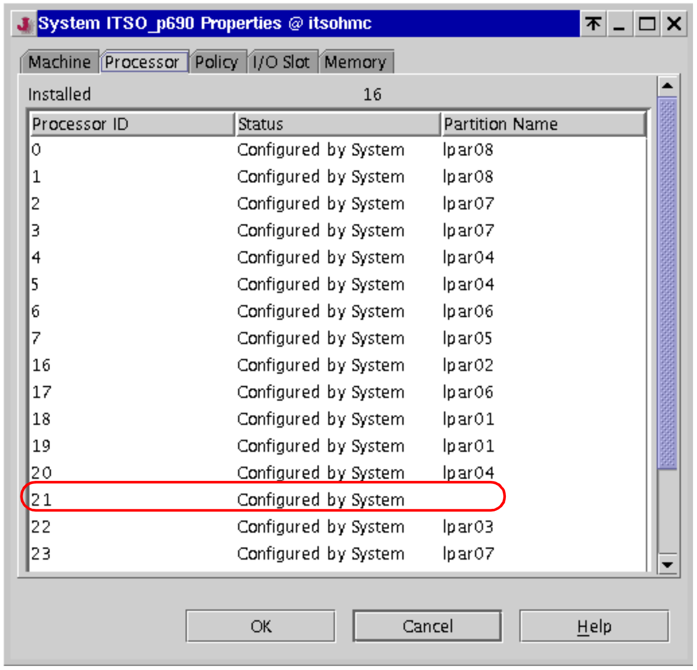
Figure 5-1 System properties: Machine[1]

Note: The Service Processor Version field (highlighted in Figure 5-1) shows the system firmware version on your managed system.

[1] The same information can be obtained using the `lssyscfg` command.

5.1.2 Processor property

The Processor property tab displays information about the installed processors, identified by their processor ID[2] and their assignment to partitions, as shown in Figure 5-2.



Installed 16

Processor ID	Status	Partition Name
0	Configured by System	lpar08
1	Configured by System	lpar08
2	Configured by System	lpar07
3	Configured by System	lpar07
4	Configured by System	lpar04
5	Configured by System	lpar04
6	Configured by System	lpar06
7	Configured by System	lpar05
16	Configured by System	lpar02
17	Configured by System	lpar06
18	Configured by System	lpar01
19	Configured by System	lpar01
20	Configured by System	lpar04
21	Configured by System	
22	Configured by System	lpar03
23	Configured by System	lpar07

Figure 5-2 System properties: Processor[3]

Processor 21 is not assigned to any partitions in Figure 5-2.

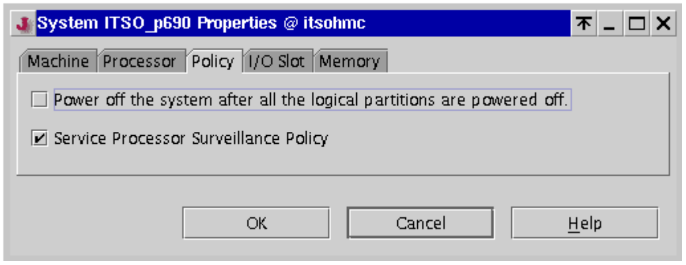
[2] This is the physical processor ID.
[3] The same information can be obtained using the **`lshwres`** command.

5.1.3 Policy property

In the Policy tab, you can choose to switch these two options on or off, as shown in Figure 5-3:

- Power off the system after all the logical partitions are powered off.
- Service Processor Surveillance Policy.

The Service Processor Surveillance Policy is a program that monitors the managed system. If the managed system is not responding, and the Service Processor Surveillance Policy is set, the state of the managed system changes from `Ready` to `No Connection` on the HMC graphical user interface.


System ITSO_p690 Properties @ itsohmc

☐ Power off the system after all the logical partitions are powered off.

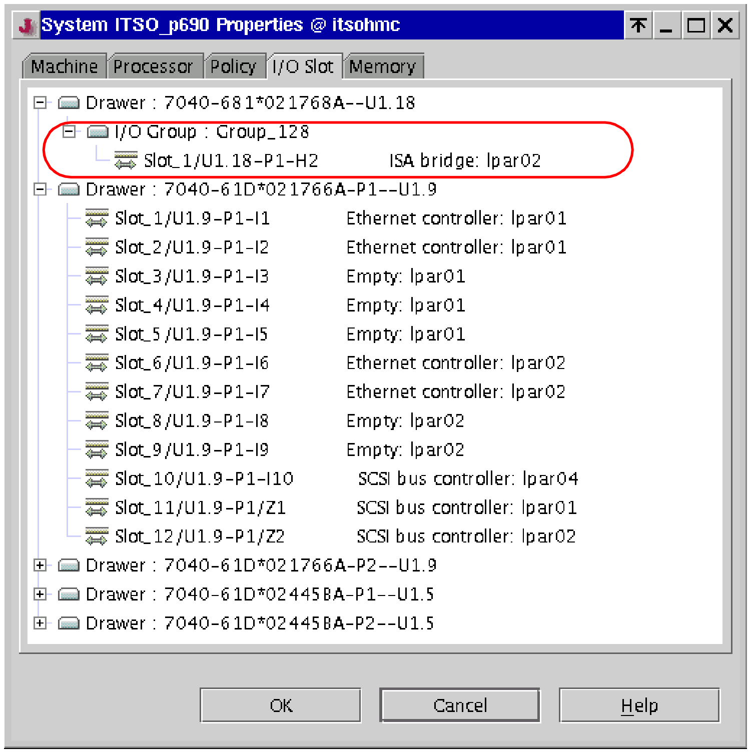
☑ Service Processor Surveillance Policy

Figure 5-3 System properties: Policy[4]

[4] Figure 5-3 shows the default setting.

5.1.4 I/O Slot property

The I/O Slot property tab displays the assignment of I/O slots to partitions and the adapter-type information grouped by drawers, as shown in Figure 5-4.



```
Drawer : 7040-681*021768A--U1.18
  I/O Group : Group_128
    Slot_1/U1.18-P1-H2        ISA bridge: lpar02
Drawer : 7040-61D*021766A-P1--U1.9
  Slot_1/U1.9-P1-I1           Ethernet controller: lpar01
  Slot_2/U1.9-P1-I2           Ethernet controller: lpar01
  Slot_3/U1.9-P1-I3           Empty: lpar01
  Slot_4/U1.9-P1-I4           Empty: lpar01
  Slot_5/U1.9-P1-I5           Empty: lpar01
  Slot_6/U1.9-P1-I6           Ethernet controller: lpar02
  Slot_7/U1.9-P1-I7           Ethernet controller: lpar02
  Slot_8/U1.9-P1-I8           Empty: lpar02
  Slot_9/U1.9-P1-I9           Empty: lpar02
  Slot_10/U1.9-P1-I10           SCSI bus controller: lpar04
  Slot_11/U1.9-P1/Z1            SCSI bus controller: lpar01
  Slot_12/U1.9-P1/Z2            SCSI bus controller: lpar02
Drawer : 7040-61D*021766A-P2--U1.9
Drawer : 7040-61D*02445BA-P1--U1.5
Drawer : 7040-61D*02445BA-P2--U1.5
```

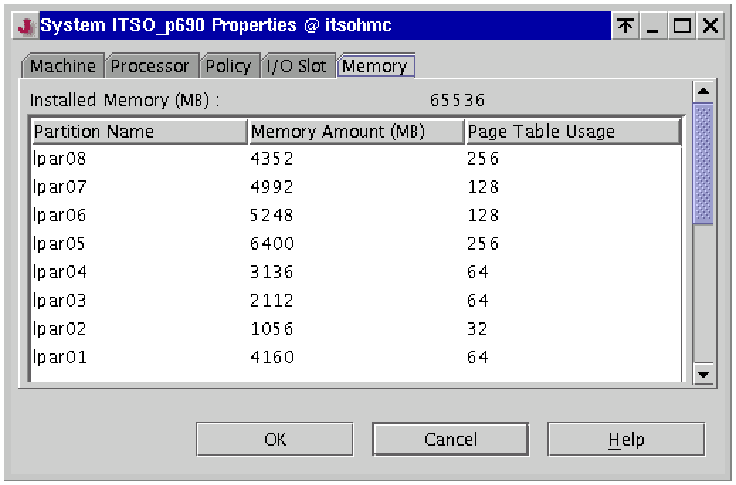
Figure 5-4 System properties: I/O Slot[5]

Note: ISA devices are not supported by DLPAR operations. The I/O slot Slot_1/U1.18-P1-H2 in Figure 5-4 represents a group of ISA devices, such as the diskette drive and native serial ports, on the pSeries 670 and pSeries 690.

[5] The same information can be obtained using the `lshwres` command.

5.1.5 Memory property

The Memory property tab displays the assigned memory amount to partitions and the page table usage information, as shown in Figure 5-5. It also shows the total installed physical memory size.



Installed Memory (MB) : 65536

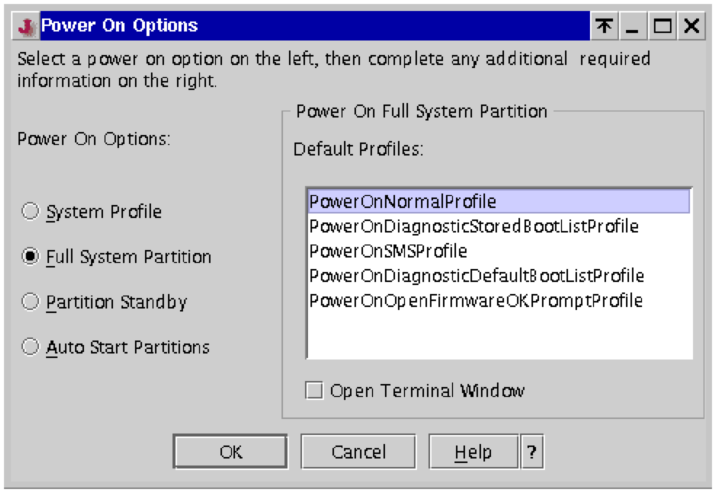
Partition Name	Memory Amount (MB)	Page Table Usage
lpar08	4352	256
lpar07	4992	128
lpar06	5248	128
lpar05	6400	256
lpar04	3136	64
lpar03	2112	64
lpar02	1056	32
lpar01	4160	64

Figure 5-5 System properties: Memory[6]

5.2 Power on the managed system

To power on the managed system, open the Server Management application from the Server Management folder and select the managed system in the Contents area. From the menu bar choose **Selected** → **Power On**. A panel opens that offers the four power-on modes shown in Figure 5-6 on page 158: System Profile, Full System Partition, Partition Standby, and Auto Start Partitions.[7]

[6] The same information can be obtained using the `lshwres` command.
[7] The `chsysstate` command can be also used to power on and off the managed systems.


Power On Options

Select a power on option on the left, then complete any additional required information on the right.

Power On Options:

- ( ) System Profile
- (●) Full System Partition
- ( ) Partition Standby
- ( ) Auto Start Partitions

Power On Full System Partition

Default Profiles:

```
PowerOnNormalProfile
PowerOnDiagnosticStoredBootListProfile
PowerOnSMSProfile
PowerOnDiagnosticDefaultBootListProfile
PowerOnOpenFirmwareOKPromptProfile
```

☐ Open Terminal Window

Figure 5-6 Power On Modes panel

The following modes are available:

System Profile The managed system activates partition profiles in the order listed in the given system profiles.

Full System Partition Only one AIX operating system image is activated that has access to all resources of the managed system. The operator panel on the media drawer displays all progress codes during the boot process. The Full System Partition has predefined profiles, as shown in Figure 5-6. You cannot change, add, or delete them. The predefined profiles are as follows:

Power On Normal

Boots an operating system from the designated boot device.

Power On Diagnostic Stored Boot List

Causes the system to perform a service mode boot using the service mode boot list saved on the managed system. If the system boots AIX from the disk drive, and AIX diagnostics are loaded on the disk drive, AIX boots to the diagnostics menu. Using this option to boot the system is the preferred way to run online diagnostics.

Power On SMS

Boots to the System Management Services (SMS) menus. The SMS menus include Password Utilities, Display Error Log, Remote Initial Program Load Setup, SCSI Utilities, Select Console, MultiBoot, Select Language, and the OK Prompt.

Power On Diagnostic Default Boot List

Similar to Power On Diagnostic Stored Boot List Profile, except the system boots using the Default Boot List that is stored in the system firmware. This is normally used to try to boot diagnostics from the CD-ROM drive. Using this option to boot the system is the preferred way to run stand-alone diagnostics.

Power On Open Firmware OK Prompt

Used only by service personnel to obtain additional debug information. When this selection is enabled, the system boots to the Open Firmware prompt.

Partition Standby This power-on mode provides two actions:

– Creating partitions

– Activation of individual partitions

When the Partition Standby power-on is completed, the operator panel on the managed system displays `LPAR…`, indicating that the managed system is ready for you to use the HMC to partition its resources or to activate configured partitions.

In Partition Standby power-on mode, the state of the Full System Partition is shown as `Not Available`.

Auto Start Partitions[8] Powers on the **managed system** to partition standby mode and then activates all partitions that have been powered on by the HMC at least once. For example, if you create a partition with four processors, use DLPAR operation to remove one processor, then shut down the system, the Auto Start Partitions power-on mode activates this partition with three processors. This is because the three-processor configuration was the last configuration used, and the HMC ignores whatever you have specified in the partition's profile. Using this option, the activated partitions boot the operating system using a normal mode boot, even if the default profile for the partition specifies the other modes, such as boot to SMS.

[8] This power-on mode is available on the HMC software release beginning with Release 3, Version 2.

5.2.1 Operation states of a managed system

This attribute of the managed system is displayed in the content area of the HMC window under the State label (see Table 5-2).

Table 5-2 Operating states of managed systems

State	Description
Initializing	The managed system is powered on and is initializing. The initialization time may vary depending on the hardware and partition configuration of the managed system.
Ready	The managed system is powered on and is operating normally.
No Power	The managed system is powered off.
Error	The operating system or the hardware of the managed system is experiencing errors.
Incomplete	The HMC cannot gather complete partition, profile, or resource information from the managed system. To rebuild the managed system, see 5.2.2, "Rebuild the managed system in the HMC" on page 160.
No Connection	The HMC cannot contact the managed system. Check the serial cable or delete and configure the managed system again.
Recovery	The partition and profile data stored in the managed system must be refreshed.
Version Mismatch	The managed system's service processor level is later than the code level of the HMC.
CUOD CTA	You must accept the CUoD license.

5.2.2 Rebuild the managed system in the HMC

The Rebuild managed system function downloads the data stored in the NVRAM of the managed system to the HMC. The NVRAM contains the properties of the managed system, the partition, system profile information, and the current states. Rebuilding the managed system is useful when the operating state indicator of a managed system in the Contents area is shown as `Incomplete`. This operation is different from performing a reload of the local HMC panel. In this operation, the HMC reloads from the information that is stored on the local database on the HMC.

To rebuild the managed system, select the managed system in the Contents area, and from the menu bar choose **Selected** → **Rebuild** managed system. When the operation finishes, the current system information of the managed system appears.

5.3 Power off the managed system

Before powering off the managed system, ensure that all partitions or the Full System Partition have been shut down and their states have changed from `Running` to `Ready`. To shut down a partition, you can use a virtual terminal window to run the **`shutdown`** command or Telnet into the partition and issue the **`shutdown`** command.

To power off the managed system, select the managed system in the Contents area, and from the menu bar choose **Selected** → **Power Off**. If you attempt to power off a system that has active partitions, you will receive a warning to that effect, but you will still be able to power off the managed system.

5.4 Operating the managed system with the HMC

Although the managed system is designed not to put any dependency on the HMC, except the specific system management operation, you should not plan to run the managed system without an HMC. The HMC is required for the operations, such as to set up or change the partition configurations, and is also a key element in configuring the Service Applications. Without an HMC, the Service Applications will not be able to provide the extended RAS capabilities that are available on the partioning-capable pSeries servers. The call home feature available with Service Focal Point provides this function through the HMC.

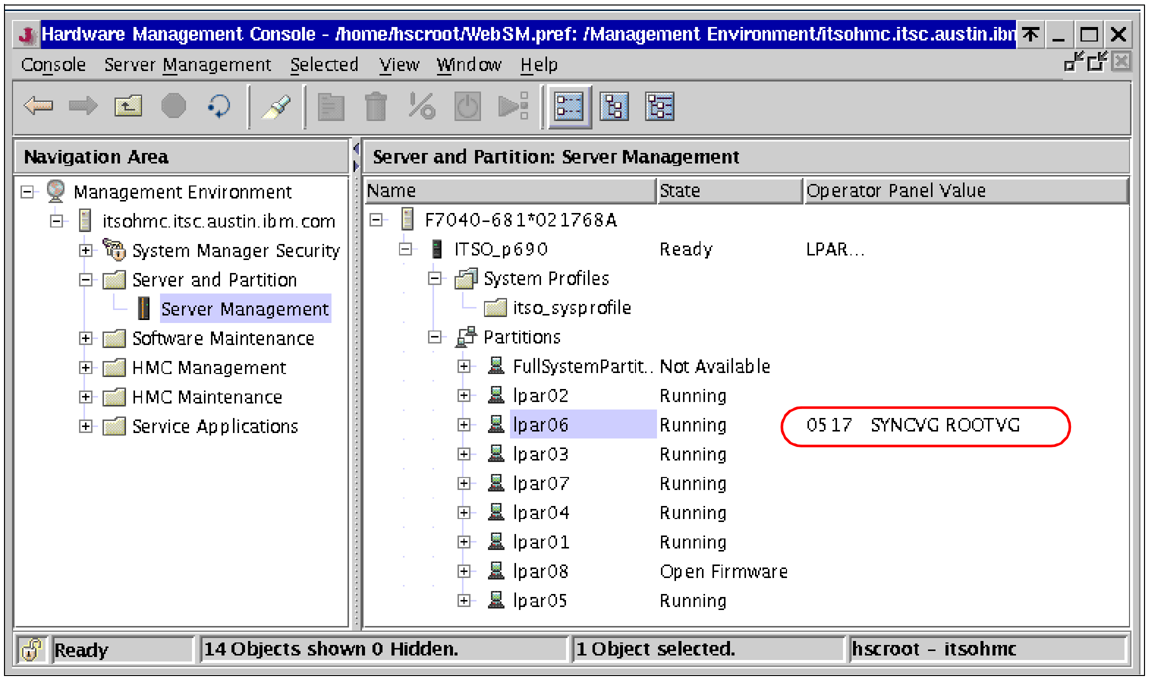
Without an HMC, it is still possible to bring up a managed system in its last configured partition state, including a boot of defined partitions, by pressing the power button on the operator panel. However, running partitions can be rebooted and restarted using the **`shutdown`** command, even if the HMC is not present.

5.4.1 Operator panel

The operator panel[9] is used to track the progress of the system unit's self tests and configuration program, to display codes when the operating system comes to an abnormal end, and to display system messages. In a logical partitioned environment, the operator panel displays an error code for most hardware or

[9] The operator panel is physically located in the media drawer in the pSeries 670 and pSeries 690.

firmware problems, but you need the HMC to display any error information written to a partition's virtual operator panel. The operator panel values of the partitions are displayed in the HMC main menu for every partition, as shown in Figure 5-7 on page 162.


Hardware Management Console - /home/hscroot/WebSM.pref: /Management Environment/itsohmc.itsc.austin.ibn
Console Server Management Selected View Window Help
Navigation Area
Management Environment
itsohmc.itsc.austin.ibm.com
System Manager Security
Server and Partition
Server Management
Software Maintenance
HMC Management
HMC Maintenance
Service Applications
Server and Partition: Server Management
Ready
14 Objects shown 0 Hidden.
1 Object selected.
hscroot - itsohmc

Name	State	Operator Panel Value
F7040-681*021768A		
ITSO_p690	Ready	LPAR...
System Profiles		
itso_sysprofile		
Partitions		
FullSystemPartit..	Not Available	
lpar02	Running	
lpar06	Running	0517 SYNCVG ROOTVG
lpar03	Running	
lpar07	Running	
lpar04	Running	
lpar01	Running	
lpar08	Open Firmware	
lpar05	Running	

Figure 5-7 Hardware Management Console operator panel codes

Note: AIX 5L Version 5.2 displays a detailed description and a four-digit LED value at the operating system boot phase, as highlighted in Figure 5-7.

5.4.2 Power button

The white power button in the operator panel acts in a logical partitioned environment and in Full System Partition, such as in a conventional pSeries machine. The managed system will come back up in the same mode in which it was previously booted. If the managed system was previously booted in a Partition Standby mode, all partitions will automatically start and run. To power off the whole system, press the button twice: once to indicate action, the second time to confirm. We recommend you shut down the operating system instances in the partitions before powering off the system.

5.4.3 Reset button

The reset button functions only in the Full System Partition mode. In Partition Standby mode, the reset button is not active. To reset a partition, use the operating system reset function of the HMC.

6

Creating and managing partitions

This chapter explains how to create, modify, and manage partitions on partioning-capable pSeries servers by providing the following sections:

- 6.1, “Partition and system profile tasks” on page 166
- 6.2, “Affinity logical partitions” on page 183
- 6.3, “Activate partitions” on page 192
- 6.5, “Reset the operating system in a partition” on page 198
- 6.3, “Activate partitions” on page 192

For more information about the concept of partition and system profiles, see 2.2.1, “Partition and system profiles” on page 31. We also provide detailed information about resource assignment to partitions in 2.2, “Partition resources” on page 31.

We recommend you accurately plan the partitions before starting your configuration work on the system.

6.1 Partition and system profile tasks

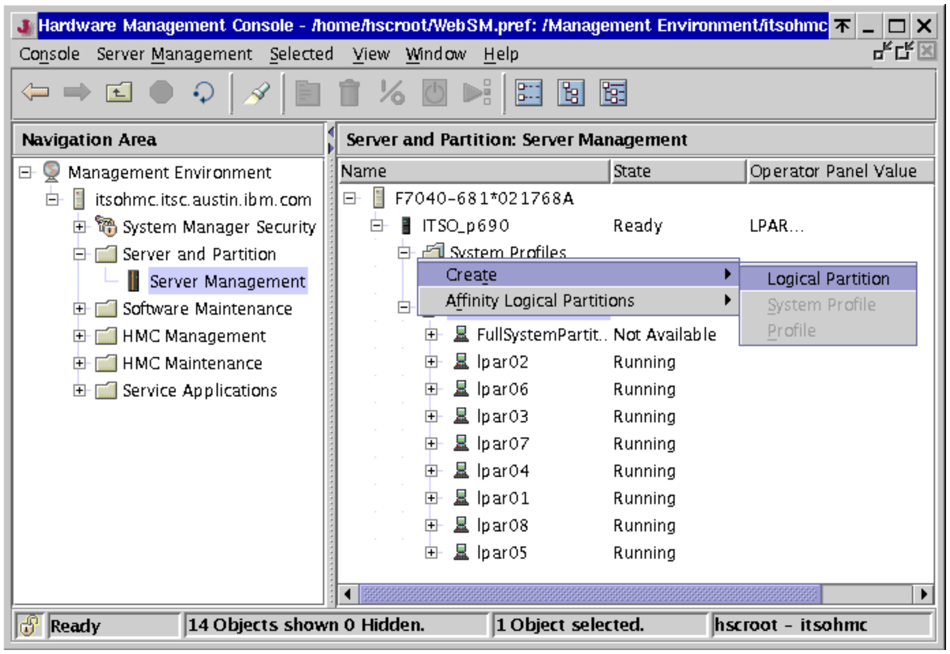
In this section, we explain how to create, change, and delete partition and system profiles.

6.1.1 Create logical partitions and partition profiles

To create a new partition or partition profiles, the managed system is required to be powered on in the Partition Standby mode (see 5.2, “Power on the managed system” on page 157).

To create logical partitions and partition profiles, do the following:

1. To create a partition, select the managed system. From the menu bar, select **Selected** → **Create**. Or select the managed system, right-click, and select **Create** → **Logical Partition**, as shown in Figure 6-1 on page 166.



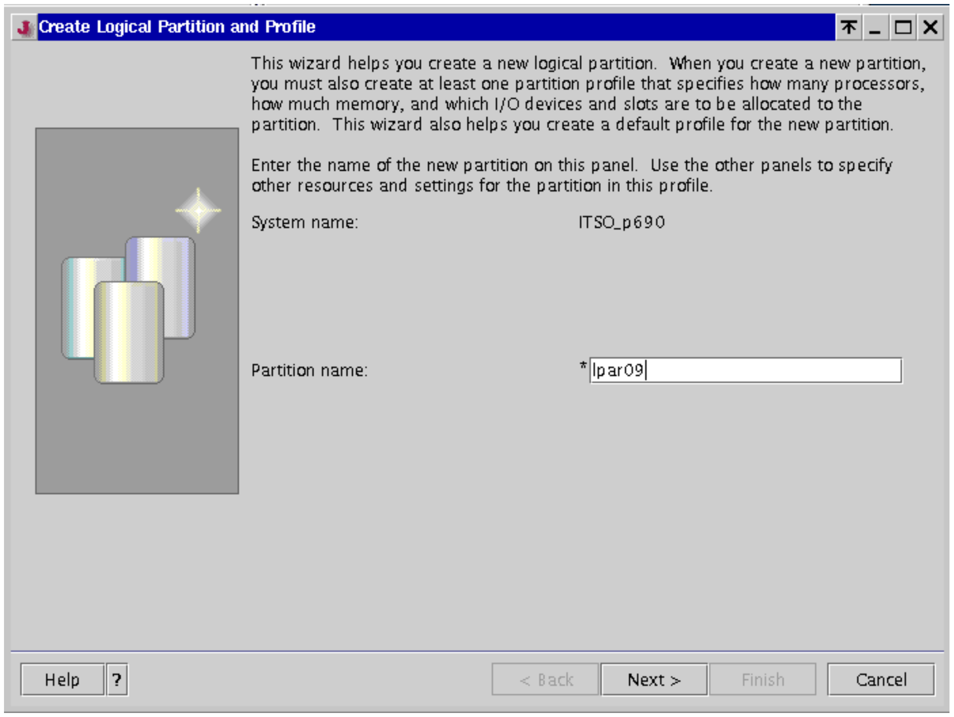
Name	State	Operator Panel Value
F7040-681*021768A		
ITSO_p690	Ready	LPAR...
FullSystemPartit..	Not Available	
lpar02	Running	
lpar06	Running	
lpar03	Running	
lpar07	Running	
lpar04	Running	
lpar01	Running	
lpar08	Running	
lpar05	Running	

Ready | 14 Objects shown 0 Hidden. | 1 Object selected. | hscroot - itsohmc

Figure 6-1 Create a logical partition

2. After you select **Logical Partition**, the Create Logical Partition and Profile wizard opens (it is shown from Figure 6-2 on page 167 to Figure 6-6 on page 172). It enables you to create a new partition with at least one default partition profile.

3. First, you have to enter a name for the partition that you are creating. In Figure 6-2, we typed `lpar09` for the partition name. The partition name has a limit of 31 characters and must be a unique name within the managed system. Click **Next**.



This wizard helps you create a new logical partition. When you create a new partition, you must also create at least one partition profile that specifies how many processors, how much memory, and which I/O devices and slots are to be allocated to the partition. This wizard also helps you create a default profile for the new partition.

Enter the name of the new partition on this panel. Use the other panels to specify other resources and settings for the partition in this profile.

System name: ITSO_p690

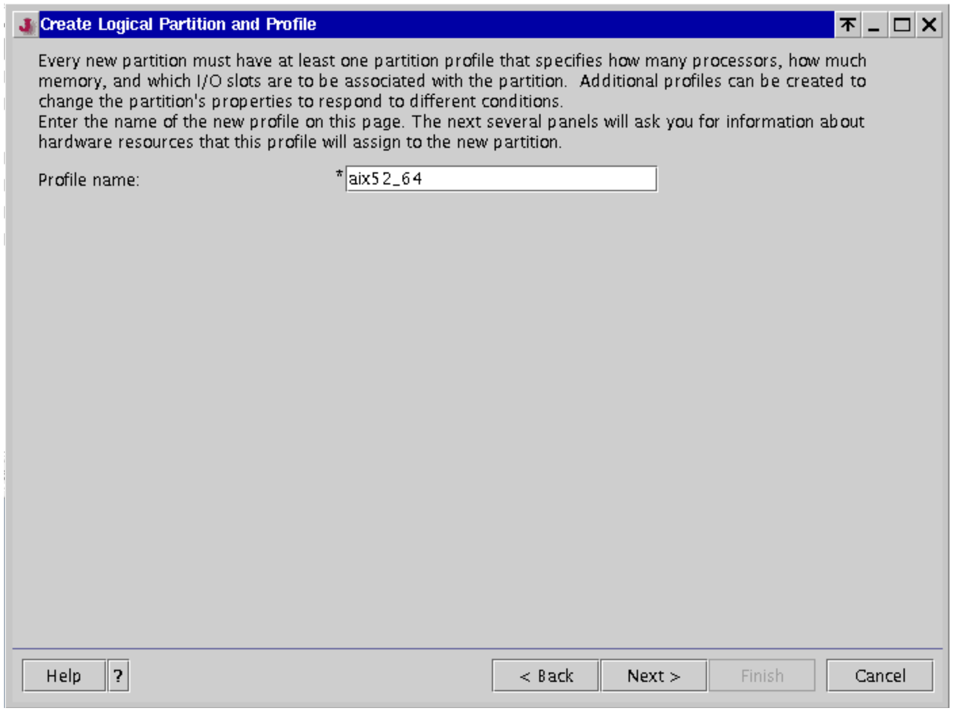
Partition name: * lpar09

Figure 6-2 Panel 1: Partition name

4. You have to enter a name for the partition profile that you are creating for this partition. In Figure 6-3, we typed `aix52_64` for the partition profile name. The partition profile name has a limit of 31 characters and must be unique within the partition.

   Note: We strongly suggest you name the profile name in accordance with the purpose of partition usage.

   Click **Next**.

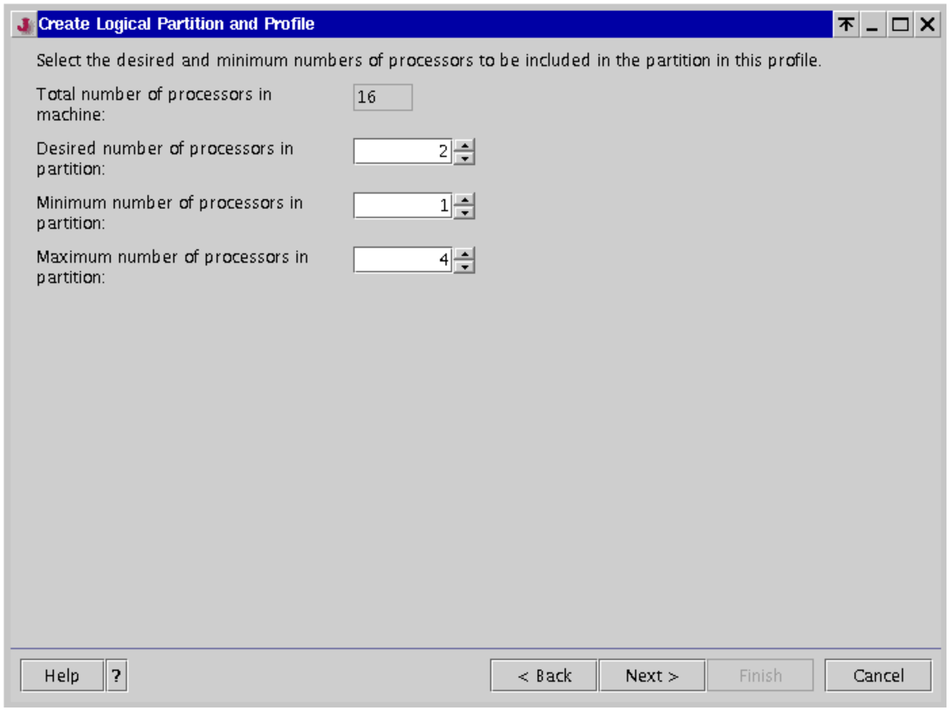


Every new partition must have at least one partition profile that specifies how many processors, how much memory, and which I/O slots are to be associated with the partition. Additional profiles can be created to change the partition's properties to respond to different conditions.
Enter the name of the new profile on this page. The next several panels will ask you for information about hardware resources that this profile will assign to the new partition.

Profile name: * aix52_64

Figure 6-3 Panel 2: Profile name

5. The wizard prompts you to enter the desired, minimum, and maximum number of processors that can be associated with this profile. We chose 2, 1, and 4 for desired, minimum, and maximum processors, as shown in Figure 6-4.


Create Logical Partition and Profile

Select the desired and minimum numbers of processors to be included in the partition in this profile.

Total number of processors in machine:	16
Desired number of processors in partition:	2
Minimum number of processors in partition:	1
Maximum number of processors in partition:	4

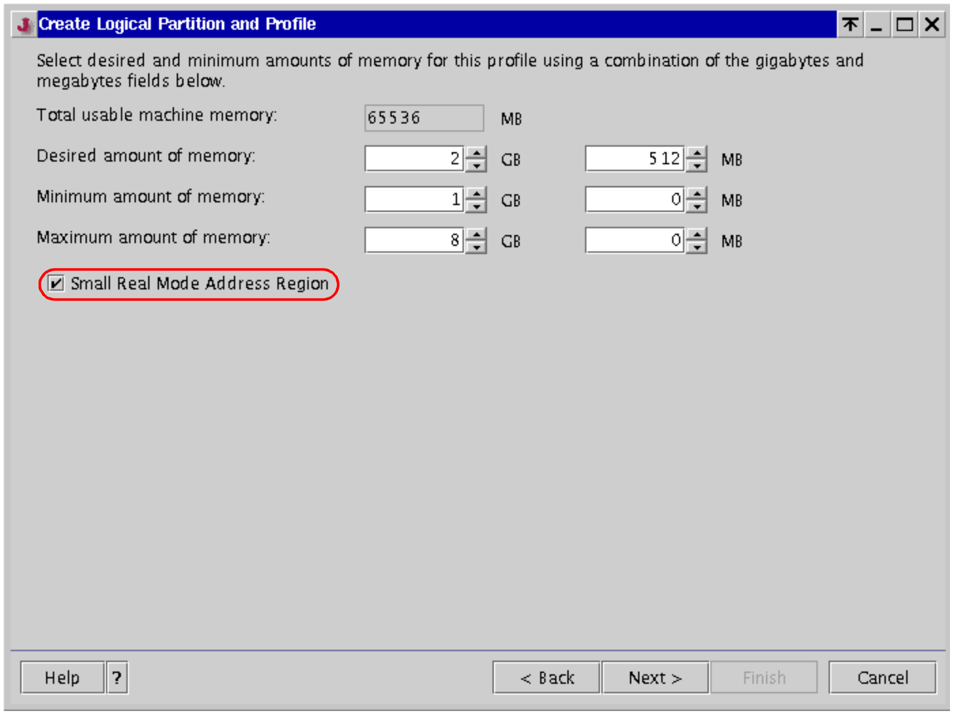
Figure 6-4 Panel 3: Number of processors

Note: You do not see the Maximum number of processors in partition field on the HMC if the HMC software level is earlier than Release 3, Version 1.

The number of processors installed in the managed system is also displayed on this panel. In Figure 6-4, you can see the number `16` shown in the Total number of processors in machine field.

Click **Next**.

6. The wizard prompts you to enter your desired, minimum, and maximum memory amounts for this partition profile, as shown in Figure 6-5. Enter the amount of minimum, desired, and required memory in 1 gigabyte (GB) increments and 256 megabyte (MB) increments. In Figure 6-5 on page 170, we entered 2.5 GB for Desired amount of memory, 1 GB for Minimum amount of memory, and 8 GB for Maximum amount of memory. You must have a minimum of 256 MB memory for each partition. The rules and limitations of the memory assignment to partitions are provided in 2.2, "Partition resources" on page 31.



Select desired and minimum amounts of memory for this profile using a combination of the gigabytes and megabytes fields below.

Total usable machine memory:	65536	MB		
Desired amount of memory:	2	GB	512	MB
Minimum amount of memory:	1	GB	0	MB
Maximum amount of memory:	8	GB	0	MB

[✓] Small Real Mode Address Region

Figure 6-5 Panel 4: Amount of memory

Note:

- The minimum amount of memory should be no less than 1/64 of the maximum amount of memory. Otherwise, AIX will not boot.
- Select the **Small Real Mode Address Region** check box only when you will run AIX 5L Version 5.2 or Linux in this partition. See 2.2.5, "Physical memory allocation to partitions" on page 38 for the further detailed information.

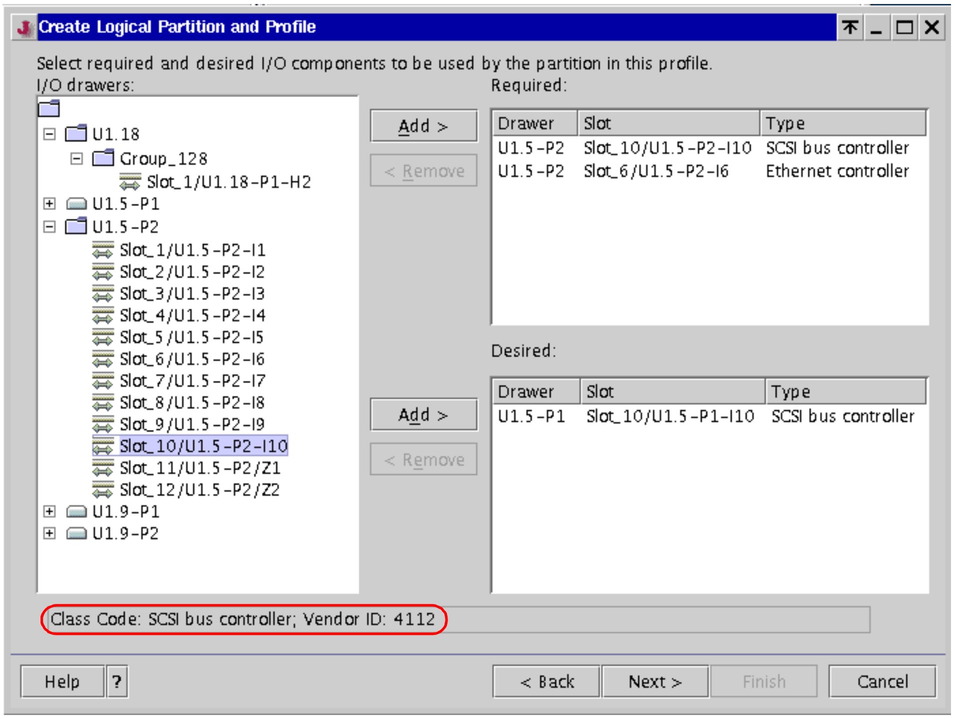
You will see the Maximum amount of memory field if the system microcode Version 3.0 or later is loaded on your pSeries 670 or pSeries 690 and the HMC software level is Release 3, Version 1 or later. Otherwise, you will not see the Maximum amount of memory field. If the HMC software level is earlier than Release 3, Version 1, either both values have to be 16 GB or less, or both values have to be 16.25 GB or greater. This means that you cannot set 17 GB for the desired amount of memory and 14 GB for the minimum amount of memory size.

The wizard also shows the total amount of memory configured for use by the system in the Total usable machine memory field. In Figure 6-5 on page 170, it shows 65536 MB (64 GB) in total.

Click **Next**.

7. The wizard shows you the I/O components assignment panel, shown in Figure 6-6. The left side of the panel displays the I/O drawers available and configured for use. You can expand the I/O tree to show the individual slots in each drawer by clicking the icon next to each drawer.

Note: The information shown in Figure 6-6 is highly dependant on our test environment explained in Appendix A, “Test environment” on page 389. If your managed system is other than pSeries 670 or pSeries 690, please consult with the appropriate publications shipped with the system.


Create Logical Partition and Profile

Select required and desired I/O components to be used by the partition in this profile.

I/O drawers:

```
⊟ U1.18
   ⊟ Group_128
        Slot_1/U1.18-P1-H2
⊞ U1.5-P1
⊟ U1.5-P2
     Slot_1/U1.5-P2-I1
     Slot_2/U1.5-P2-I2
     Slot_3/U1.5-P2-I3
     Slot_4/U1.5-P2-I4
     Slot_5/U1.5-P2-I5
     Slot_6/U1.5-P2-I6
     Slot_7/U1.5-P2-I7
     Slot_8/U1.5-P2-I8
     Slot_9/U1.5-P2-I9
     Slot_10/U1.5-P2-I10
     Slot_11/U1.5-P2/Z1
     Slot_12/U1.5-P2/Z2
⊞ U1.9-P1
⊞ U1.9-P2
```

Required:

Drawer	Slot	Type
U1.5-P2	Slot_10/U1.5-P2-I10	SCSI bus controller
U1.5-P2	Slot_6/U1.5-P2-I6	Ethernet controller

Desired:

Drawer	Slot	Type
U1.5-P1	Slot_10/U1.5-P1-I10	SCSI bus controller

Class Code: SCSI bus controller; Vendor ID: 4112

Figure 6-6 Panel 5: I/O components

Because some I/O slots are grouped by default, if you attempt to assign a member of one of these grouped slots to a partition profile, the entire group will be assigned. If your managed system is pSeries 670 or pSeries 690, you will see the special group, Group_128, in the U1.18 physical location code. This is composed of ISA devices available on the pSeries 670 or pSeries 690, such as the diskette drive and native serial ports.

Note: ISA devices are not supported by dynamic logical partitioning operations.

If you click the slot, it displays the details about the physical location code of the slot. If an adapter is installed in that slot, it also shows the type of the adapter (you can see the text Class Code: SCSI bus controller, Vendor ID : 4112) Please note that the slots in the I/O drawers field are not listed in sequential order.

You might be curious why you see 12 slots in the 7040-61D I/O drawer in Figure 6-6 on page 172 even though there are 20 PCI slots available for each I/O drawer. To better understand why, you have to understand the following points:

– An I/O drawer (7040-61D) is composed of two physically symmetrical planar boards. Each planar board has 10 PCI slots available. Therefore, an I/O drawer has 20 PCI slots available in total.

– Each planar board is displayed as a separate entity in Figure 6-6 on page 172. In this example, the left side of the planar board of the second I/O drawer (viewed from the rear side of the CEC) is represented as U1.5-P1. The right side is represented as U1.5-P2.

– Each planar has two built-in SCSI controllers, in addition to having 10 PCI slots available. These controllers are also displayed as I/O slots in Figure 6-6 on page 172. Therefore, 12 I/O slots in total are displayed under the tree.

– Each built-in SCSI controller is connected to the different 4-disk pack placed in the front of the I/O drawer. The first built-in SCSI controller is displayed as Slot_11 (physical location code is P1/Z1 or P2/Z1); the second controller is displayed as Slot_12 (physical location code P1/Z2 or P2/Z2).

See “pSeries 670- and pSeries 690-dependent information” on page 396 for the hardware-specific information about the pSeries 670 and pSeries 690.

Select the slot you want to assign to this partition profile and click **Add**. If you want to add another slot, repeat this process. Slots are added individually to the profile; you cannot add more than one slot at a time.

There are also desired and required I/O slots fields in this panel. In Figure 6-6 on page 172, you can see two slots are assigned as required I/O slots, and one slot is assigned as desired.

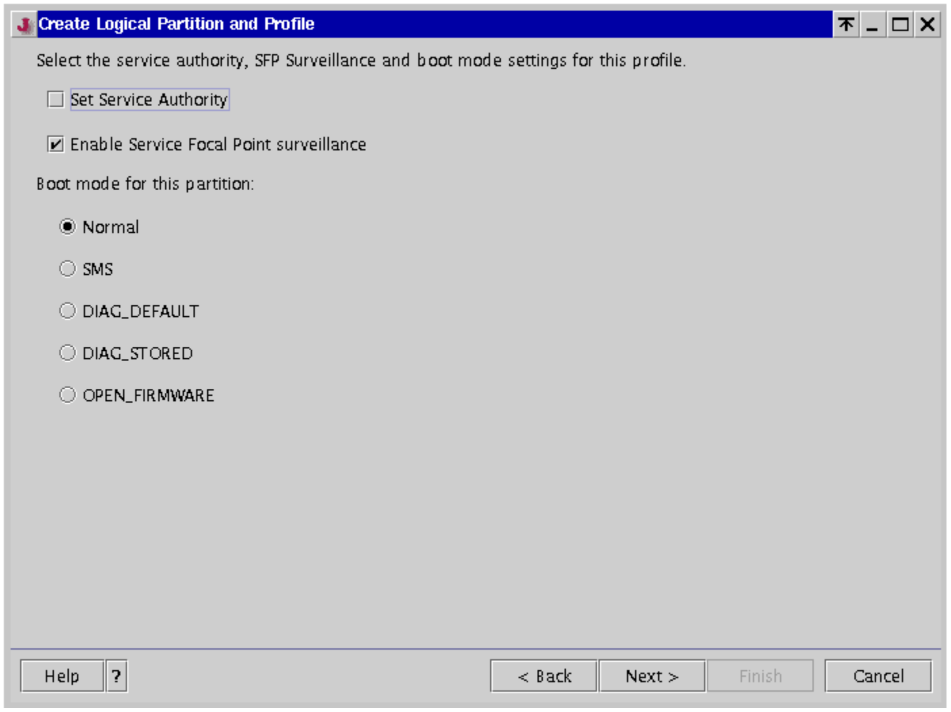
The real allocation of assigned I/O slots will occur when the partition is activated. The I/O slot is allocated to any partition on a "first-come, first-served" basis. The partition profile defines the assigned I/O slots for the partition, but it does not reserve any physical I/O slot in its definition until that partition profile is activated.

For example, if you assigned an I/O slot to two partitions as a required I/O slot, then only the partition you activated first will be activated, and the second partition will fail to activate due to the unavailability of a required resource. However, if you assigned this I/O slot to two partitions as a desired I/O slot, then one partition will be activated with the I/O slot assigned, and the other will be activated without the I/O slot assigned.

Click **Next**.

8. The next step, shown in Figure 6-7 on page 175, enables you to set service authority and boot mode policies for this partition profile. Select the **Set Service Authority** check box if you want this partition to be used by IBM customer support personnel to perform system firmware updates and set other system policy parameters (in this example, the service authority is not selected). Although it is technically possible to set the service authority to several partitions, only one partition will acquire the capability by "first-come, first-served" basis. If a partition with the service authority is activated when another service partition with the authority is already active, the activation of the partition would fail.[1]

[1] See Figure 6-21 on page 196.



Create Logical Partition and Profile

Select the service authority, SFP Surveillance and boot mode settings for this profile.

☐ Set Service Authority

☑ Enable Service Focal Point surveillance

Boot mode for this partition:

● Normal

○ SMS

○ DIAG_DEFAULT

○ DIAG_STORED

○ OPEN_FIRMWARE

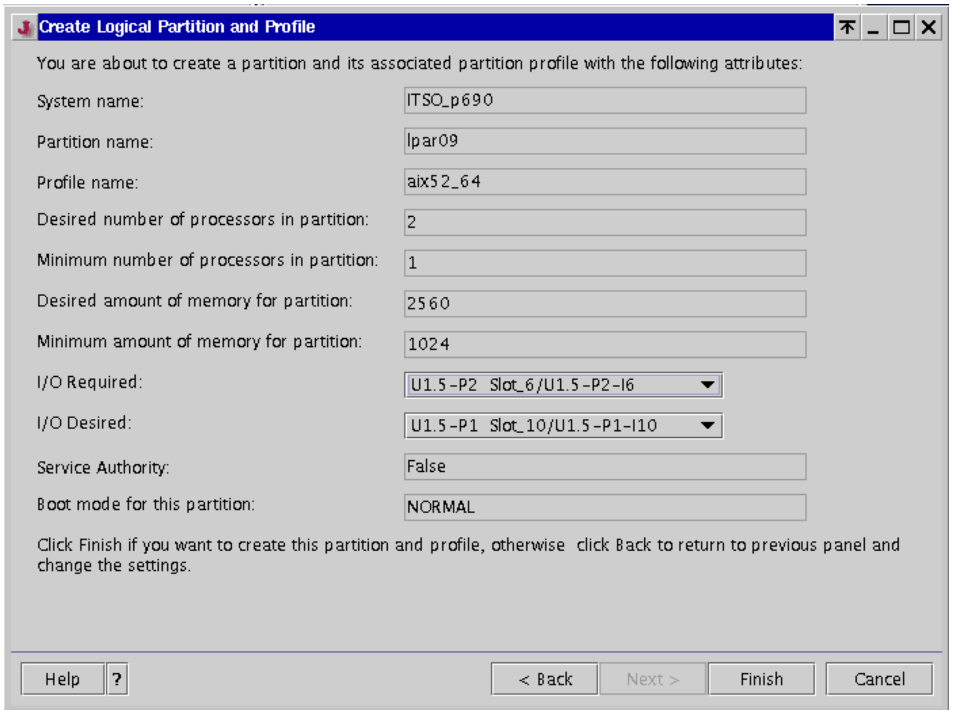
Figure 6-7 Panel 6: Set service authority and boot mode

You can also select the boot mode for this partition profile by selecting a value in the Boot mode for this partition field. For more information about the boot mode for the partition, see 5.2, “Power on the managed system” on page 157.

Click **Next**.

9. The last panel of this wizard is shown in Figure 6-8. It shows summary information for the partition profile you are creating. Review the information to ensure that you have the appropriate resources assigned to this partition.

   If you want to change the configuration, click **Back**. Otherwise, click **Finish** to create the partition and the partition profile.



You are about to create a partition and its associated partition profile with the following attributes:

System name:	ITSO_p690
Partition name:	lpar09
Profile name:	aix52_64
Desired number of processors in partition:	2
Minimum number of processors in partition:	1
Desired amount of memory for partition:	2560
Minimum amount of memory for partition:	1024
I/O Required:	U1.5-P2 Slot_6/U1.5-P2-I6
I/O Desired:	U1.5-P1 Slot_10/U1.5-P1-I10
Service Authority:	False
Boot mode for this partition:	NORMAL

Click Finish if you want to create this partition and profile, otherwise click Back to return to previous panel and change the settings.

Figure 6-8 Panel 7: Partition attributes

The new partition, along with the default profile you just created, appears underneath the managed system tree in the Contents area.

After you have created a partition, you can install an operating system on the partition. To install AIX on the partition, see Chapter 7, “Installing and migrating AIX in a partitioned environment” on page 201.

6.1.2 Create additional partition profiles

You can create new partition profiles for an existing partition that has a default partition profile defined. To create an additional partition profile, do the following:

1. In the Contents area, select the partition for which you want to create an additional profile. If you select the managed system, you will create a new partition, not an additional partition profile.
2. From the menu bar, select **Selected** → **Create** → **Profile**.

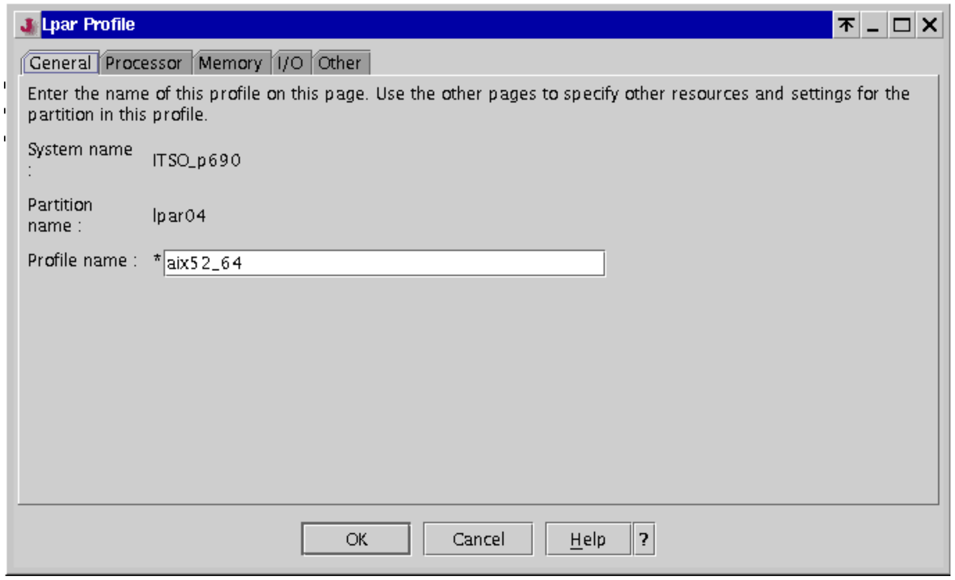
The Create Logical Partition and Profile wizard opens, and you now can begin to assign resources to the new partition profile. You can also create additional profiles by copying the default profile to another profile and editing the resources that are assigned to it.

6.1.3 View and modify partition profile properties

You can view or modify the properties of a partition profile using the following steps:

1. In the Contents area, select the profile.
2. From the menu bar, select **Selected** → **Properties**.

A window similar to the one shown in Figure 6-9 opens.



Enter the name of this profile on this page. Use the other pages to specify other resources and settings for the partition in this profile.

System name : ITSO_p690

Partition name : lpar04

Profile name : *aix52_64

Figure 6-9 Property of a partition profile

The window contains five tabs. Each tab has a panel very similar to the one shown in the partition creation process:

General	Shown in Figure 6-9 on page 177
Processor	Very similar to Figure 6-4 on page 169
Memory	Very similar to Figure 6-5 on page 170
I/O	Very similar to Figure 6-6 on page 172
Other	A panel very similar to Figure 6-7 on page 175

You can modify the partition profile while the partition is active. However, the modification takes effect after the reactivation of the partition.

6.1.4 Copy a partition profile

You can copy the contents of a partition profile you have already created to another partition profile. For example, you might decide that you need a partition profile that is similar to one that you have already created, but with a small change in resource allocation.

To copy a partition profile, do the following:

1. In the Contents area, select the existing partition profile you want to copy.
2. From the menu bar, select **Selected** → **Copy**.
3. Type a unique name for the new copy.
4. Click **OK**.

Note: You can copy partition profiles inside the partition only. You cannot copy a partition profile from a partition to other partitions.

6.1.5 Change default partition profiles

When you create a partition, the HMC requires that you create at least one profile called the *default* profile. In the Contents area, the default profile is marked in the list of the partition profiles with an icon. For example, you can distinguish the default partition profile aix51_64 from the other partition profiles for the lpar02 partition in Figure 4-10 on page 141, because the default partition profile has the following small icon with the character *c* (current).

You can choose a partition that is not the default partition when activating a partition using the following steps:

1. In the Contents area, select the partition.
2. From the menu bar, select **Selected** → **Change Default Profile**.
3. Select the profile that you want to make the default profile from the list.

Note: This step does not change the designation of a default profile.

6.1.6 Understand partition boot errors

You can use the following steps to determine the boot error value. This task is only available if a partition is in an error state after you attempted to activate it.

To determine the boot error value, do the following:

1. In the Contents area, right-click the partition that is in the Error state.
2. Select **Read Boot Error Value**. A panel opens that gives you more information about why the boot failed.

For more information about boot error values, please refer to *IBM Hardware Management Console for pSeries Installation and Operations Guide*, SA38-0590.

6.1.7 Delete partition profiles

You can delete a partition profile using the following steps:

1. In the Contents area, select the profile.

   Note: You should not select the partition itself to avoid deleting an entire partition.

2. From the menu bar, select **Selected** → **Delete**.

6.1.8 Create system profiles

We provide detailed information about the concept of system profiles in 2.2.1, “Partition and system profiles” on page 31. By using system profiles, you can activate some partitions in one operation or power on the managed system.

To create a system profile, do the following:

1. Select the managed system and select the **Selected** menu, or right-click and select **Create** → **System Profile**, as shown in Figure 6-10.

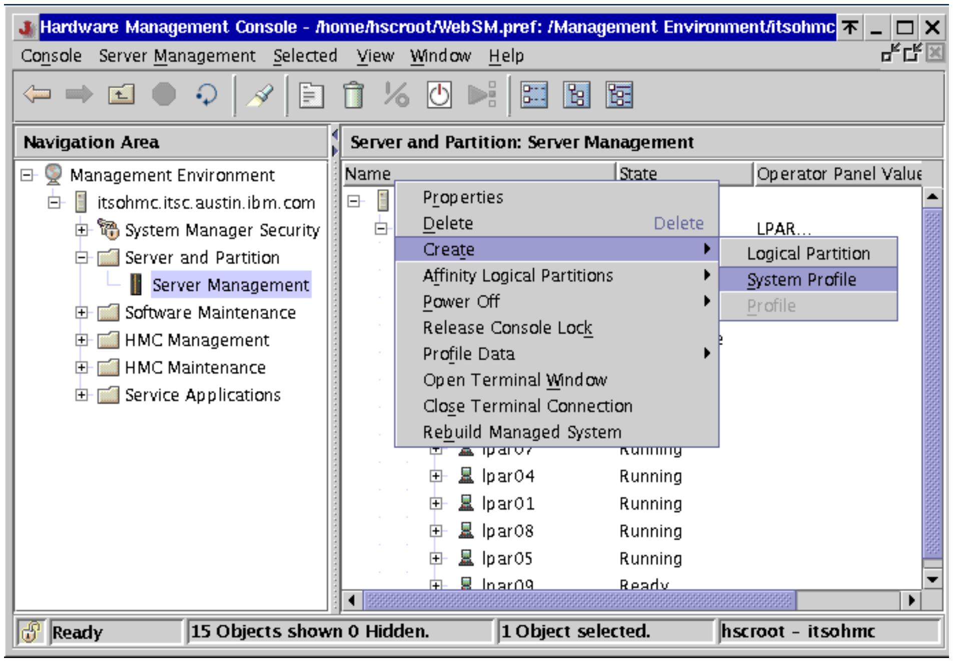
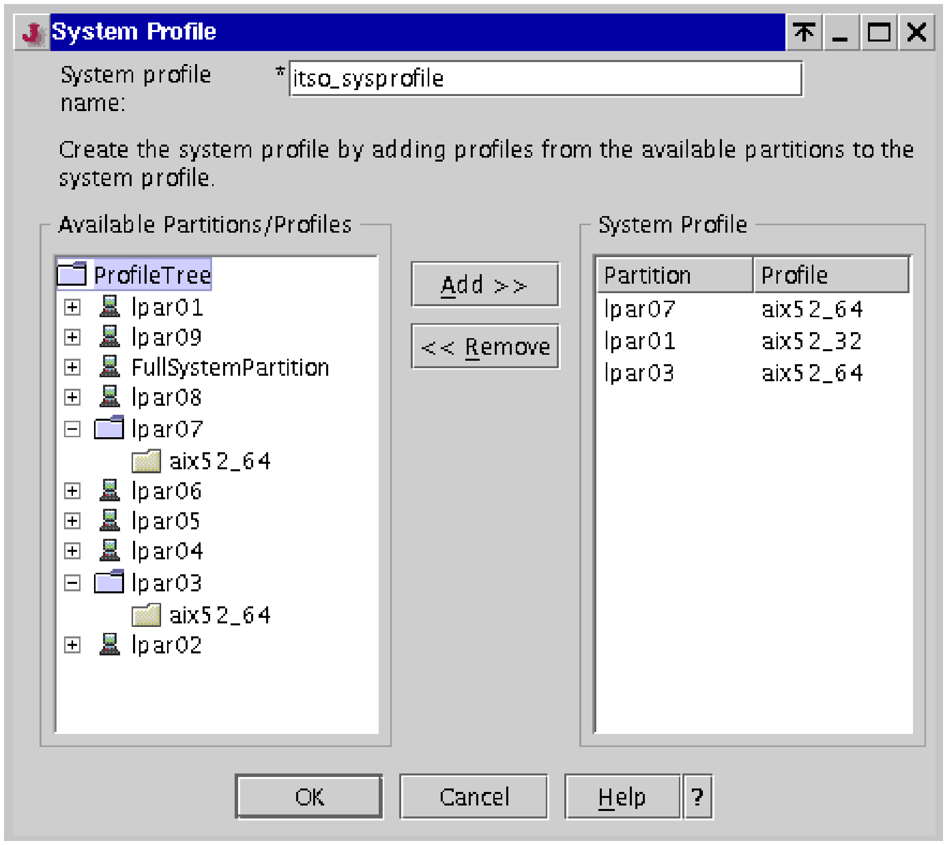


Figure 6-10 Creating system profiles

2. The System Profile panel opens, as shown in Figure 6-11 on page 181. In this example, we named a system profile itso_sysprofile and included three partition profiles.

   Click **OK**.



System profile name: * itso_sysprofile

Create the system profile by adding profiles from the available partitions to the system profile.

System Profile

Partition	Profile
lpar07	aix52_64
lpar01	aix52_32
lpar03	aix52_64

Figure 6-11 Setting up the system profiles menu

Once created, the system profile itso_sysprofile appears under the System Profile of the managed system (see Figure 5-7 on page 162).

6.1.9 View and modify system profile properties

You can view the properties of the system profile using the following steps:

1. In the Contents area, select the system profile.
2. From the menu bar, select **Selected** → **Properties**.

You can also modify the properties of the system profile by changing the system profile information as appropriate.

6.1.10 Copy a system profile

You can copy the contents of a system profile you have already created to a new system profile using the following steps:

1. In the Contents area, select the existing system profile that you want to copy.
2. From the menu bar, select **Selected** → **Copy**.
3. In the Copy Profile panel, type the new system profile name. Click **OK**.

6.1.11 Delete a system profile

You can delete a system profile using the following steps:

1. In the Contents area, select the system profile.
2. From the menu bar, select **Selected** → **Delete**.
3. The Delete System Profile panel opens. Click **Yes** to delete the system profile.

6.1.12 Activate a system profile

In order to activate a system profile, be sure to shut down the operating system for any participating active partition. The state has to show `Ready`, not `Running`. You can activate a system profile using the following steps:

1. In the Contents area, select the system profile.
2. From the menu bar, select **Selected** → **Activate**.

For example, if you activate the system profile itso_sysprofile in Figure 6-11 on page 181, the following three partitions shown in Table 6-1 will be activated in this order.

Table 6-1 Activation order example using the system profile

Activation order	Partition name	Partition profile name
1	lpar07	aix52_64
2	lpar01	aix52_32
3	lpar03	aix52_64

6.1.13 Power on using a system profile

You can power on your managed system using a predefined system profile. To learn more about powering on using a system profile you have already created, see 5.2, "Power on the managed system" on page 157.

6.2 Affinity logical partitions

Affinity logical partitions are a special group of partitions that can be created on the pSeries 670 and pSeries 690 depending on the configuration. The process of creating a group of affinity logical partitions is similar to the process of creating logical partitions. The only difference is that the system does the processor and memory assignment for you.

To determine if your managed system is capable of running affinity logical partitions, check your managed system's properties (see Figure 5-1 on page 153).

Note: A managed system can run only *regular* or affinity logical partitions at a time. The first activated partition type (regular or affinity) will determine whether the managed system runs regular or affinity logical partitions.

6.2.1 Create affinity logical partitions

To create affinity logical partitions, do the following:

1. In the Contents area, select the managed system.

2. From the Selected menu, select **Affinity Logical Partitions**, and then select **Setup**, as shown in Figure 6-12.

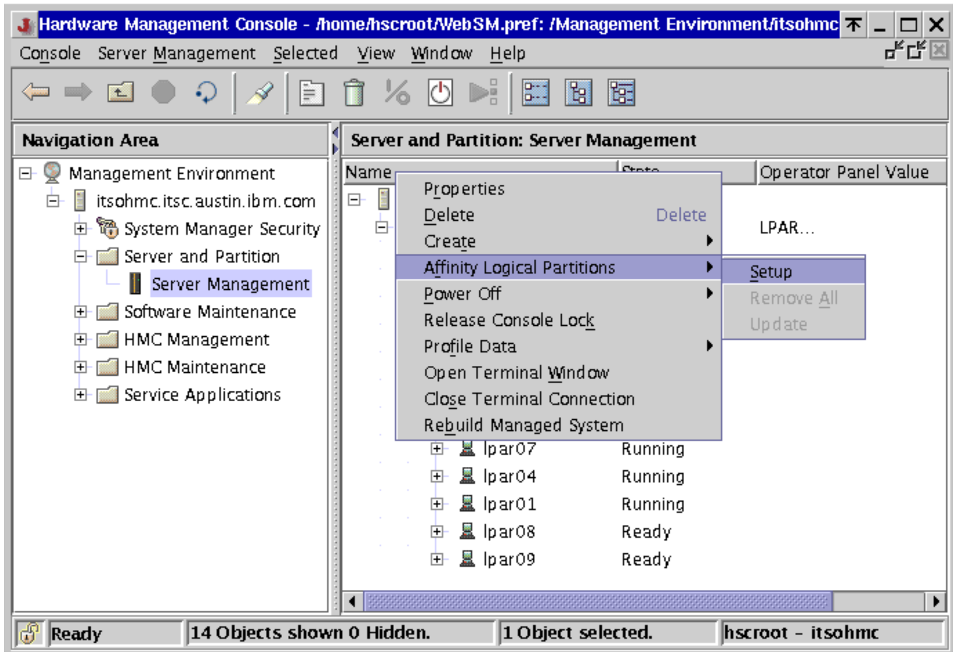
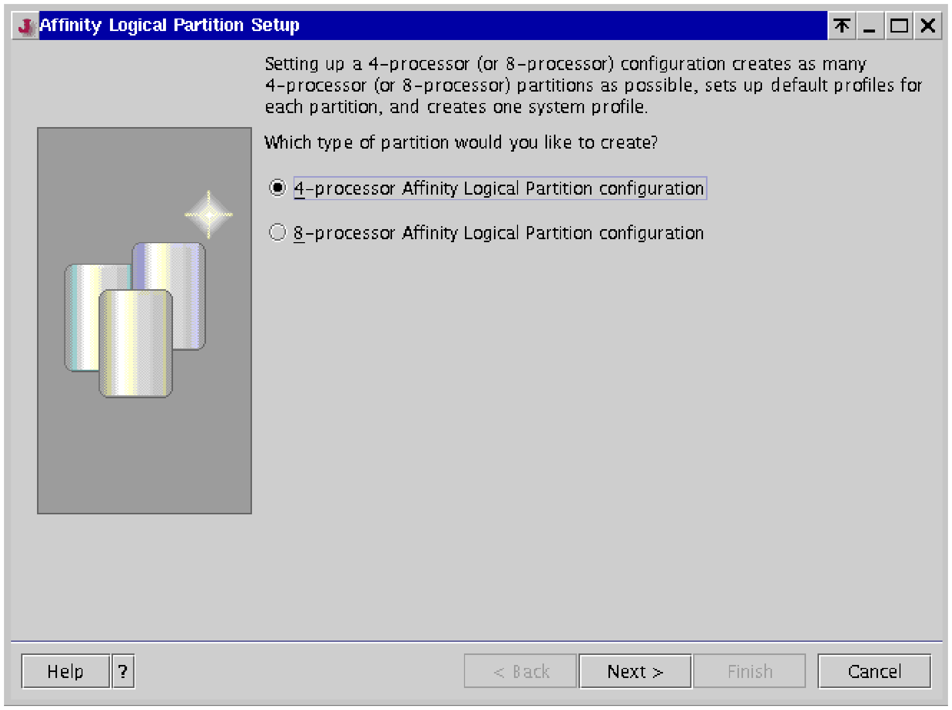


Figure 6-12 Affinity logical partitions setup

3. The Affinity Logical Partition Setup wizard opens, as shown in Figure 6-13.


Affinity Logical Partition Setup

Setting up a 4-processor (or 8-processor) configuration creates as many 4-processor (or 8-processor) partitions as possible, sets up default profiles for each partition, and creates one system profile.

Which type of partition would you like to create?

[✓ 4-processor Affinity Logical Partition configuration]

8-processor Affinity Logical Partition configuration

Figure 6-13 Selecting 4-processor or 8-processor ALPAR configuration

4. Select the type of affinity partition you want to create. You can select either of the following configurations:
   – 4-processor Affinity Logical Partition configuration
   – 8-processor Affinity Logical Partition configuration

Note: You cannot configure a combination of 4- and 8-processor affinity logical partitions on a managed system.

In this example, we selected 4-processor Affinity Logical Partition configuration. This selection creates four 4-processor affinity logical partitions, as shown in Figure 6-14 (see Appendix A, “Test environment” on page 389).

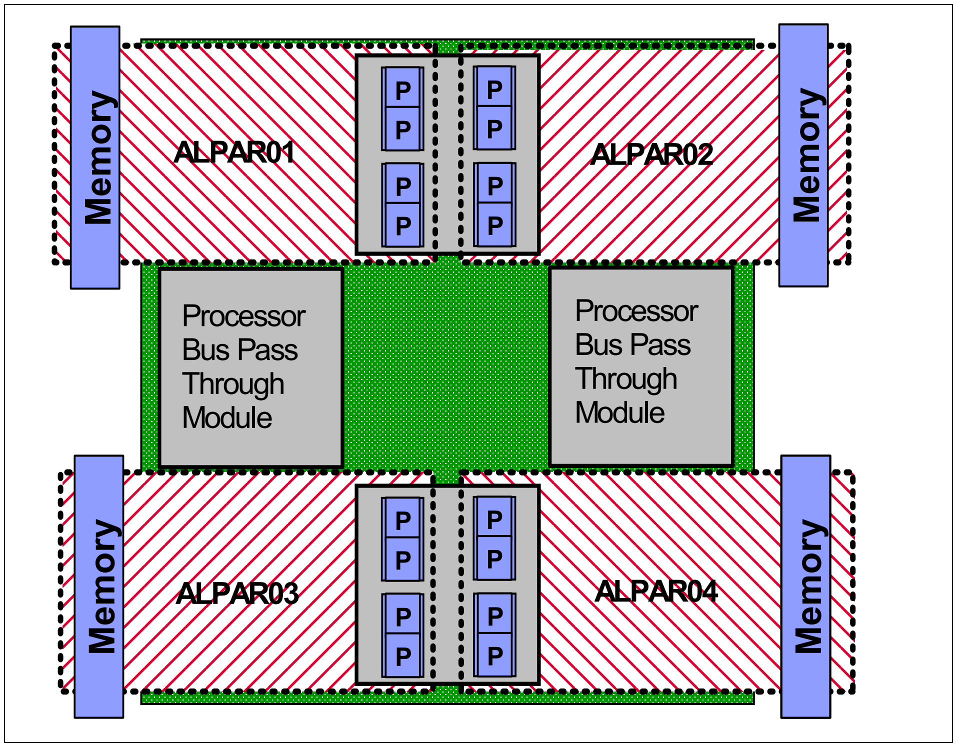
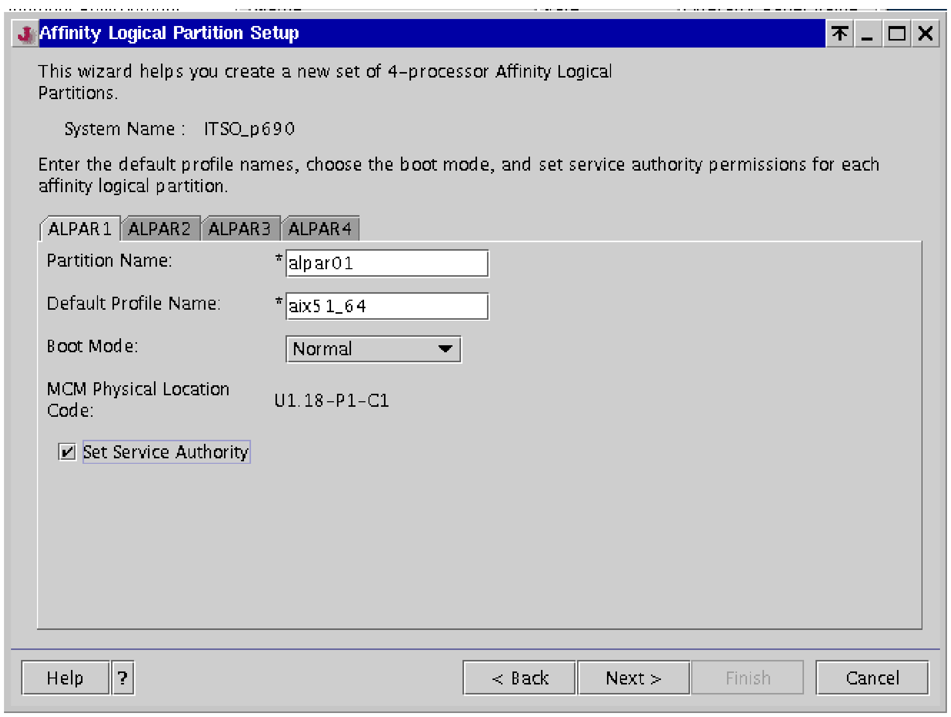


Figure 6-14 Four 4-processor ALPAR configuration

5. Specify the partition and partition profile names for all affinity logical partitions, as shown in Figure 6-15. Then, click **Next**.


Affinity Logical Partition Setup

This wizard helps you create a new set of 4-processor Affinity Logical Partitions.

System Name : ITSO_p690

Enter the default profile names, choose the boot mode, and set service authority permissions for each affinity logical partition.

Partition Name: * alpar01

Default Profile Name: * aix51_64

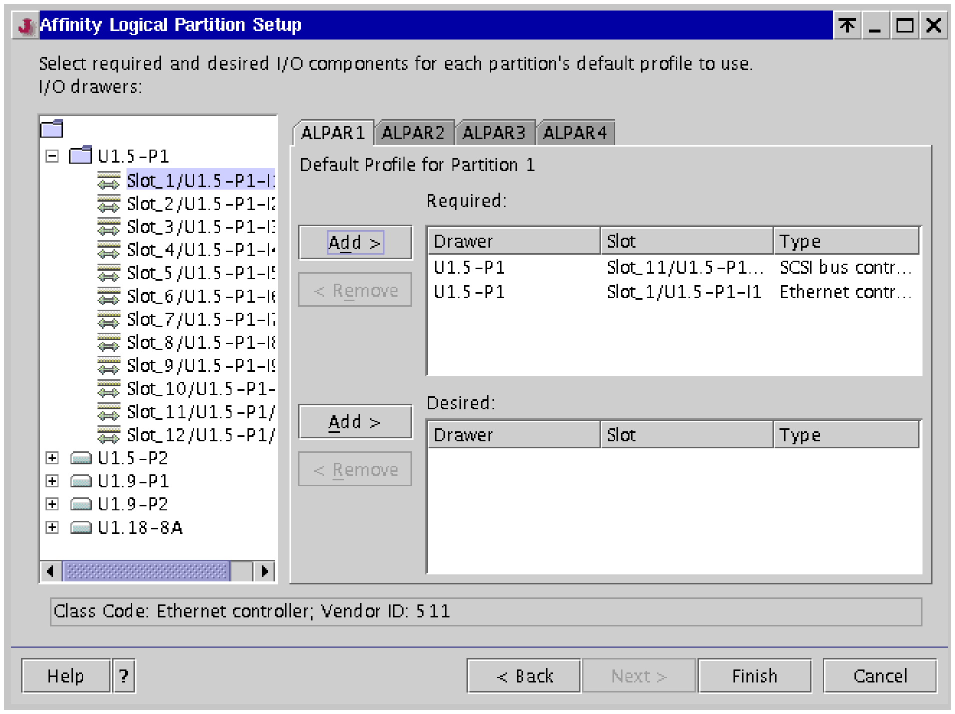
Boot Mode: Normal

MCM Physical Location Code: U1.18-P1-C1

☑ Set Service Authority

Figure 6-15 ALPAR partition and profile name

6. Assign I/O slots (adapters) to all affinity logical partitions, as shown in Figure 6-16. Click **Finish**.



Select required and desired I/O components for each partition's default profile to use.
I/O drawers:

```
⊟ U1.5-P1
    Slot_1/U1.5-P1-I
    Slot_2/U1.5-P1-I
    Slot_3/U1.5-P1-I
    Slot_4/U1.5-P1-I
    Slot_5/U1.5-P1-I
    Slot_6/U1.5-P1-I
    Slot_7/U1.5-P1-I
    Slot_8/U1.5-P1-I
    Slot_9/U1.5-P1-I
    Slot_10/U1.5-P1-
    Slot_11/U1.5-P1/
    Slot_12/U1.5-P1/
⊞ U1.5-P2
⊞ U1.9-P1
⊞ U1.9-P2
⊞ U1.18-8A
```

ALPAR1 | ALPAR2 | ALPAR3 | ALPAR4

Default Profile for Partition 1

Required:

Drawer	Slot	Type
U1.5-P1	Slot_11/U1.5-P1...	SCSI bus contr...
U1.5-P1	Slot_1/U1.5-P1-I1	Ethernet contr...

Desired:

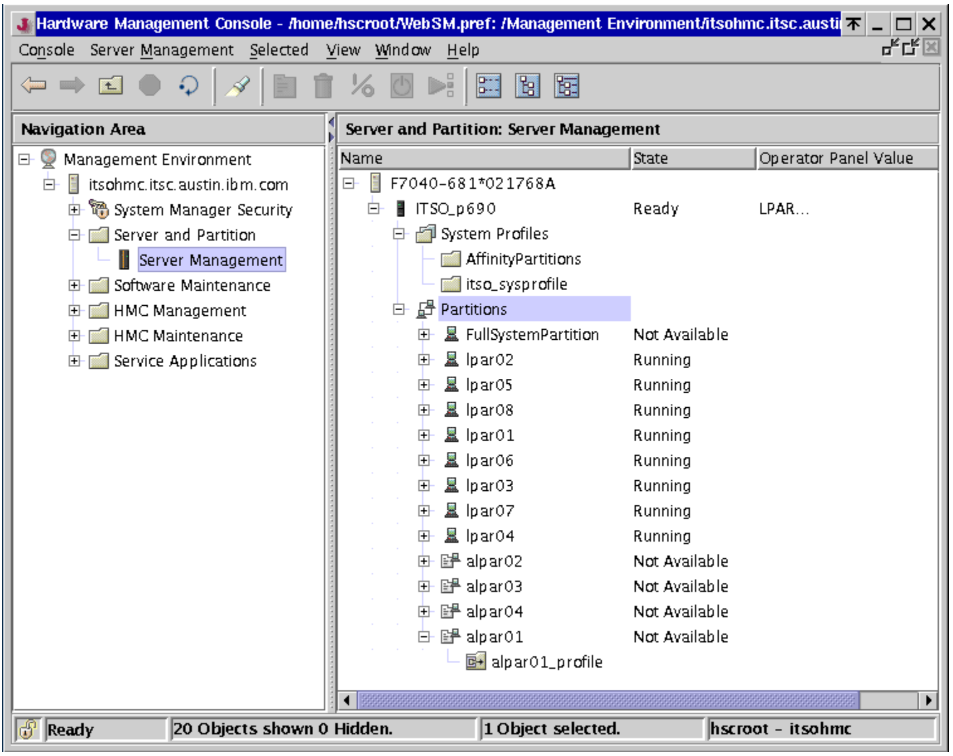
Drawer	Slot	Type

Class Code: Ethernet controller; Vendor ID: 511

Figure 6-16 ALPAR assigning I/O slots

Note: There is no memory assignment step when creating affinity logical partitions.

Four affinity logical partitions, alpar01 to alpar04, are created, as shown in Figure 6-17. The *AffinityPartitions* system profile, which includes all the created affinity logical partitions, is also created.


Hardware Management Console - /home/hscroot/WebSM.pref: /Management Environment/itsohmc.itsc.austi
Console
Server Management
Selected
View
Window
Help
Navigation Area
Server and Partition: Server Management

```
Management Environment
  itsohmc.itsc.austin.ibm.com
    System Manager Security
    Server and Partition
      Server Management
    Software Maintenance
    HMC Management
    HMC Maintenance
    Service Applications
```

Name	State	Operator Panel Value
F7040-681*021768A		
ITSO_p690	Ready	LPAR...
System Profiles		
AffinityPartitions		
itso_sysprofile		
Partitions		
FullSystemPartition	Not Available	
lpar02	Running	
lpar05	Running	
lpar08	Running	
lpar01	Running	
lpar06	Running	
lpar03	Running	
lpar07	Running	
lpar04	Running	
alpar02	Not Available	
alpar03	Not Available	
alpar04	Not Available	
alpar01	Not Available	
alpar01_profile		

Ready | 20 Objects shown 0 Hidden. | 1 Object selected. | hscroot - itsohmc

Figure 6-17 Created ALPAR

Note: The newly created affinity logical partitions cannot be activated until you reboot the managed system with the AffinityPartition system profile. If you boot the managed system with the AffinityPartition system profile, the regular partitions and the Full System Partition cannot be activated. In other words, activation of affinity logical partitions and regular partitions are mutually exclusive.

6.2.2 Manage resources in affinity logical partitions

To update affinity logical partitions after a service representative has added or removed resources on the managed system, do the following:

1. Log in to the HMC using either the System Administrator or Advanced Operator role.

2. In the Contents area, select the managed system.
3. From the Selected menu, select **Affinity Logical Partition** → **Update**.

The HMC then assesses what resources have been added or removed and asks you if you would like to add or remove affinity partitions, as appropriate.

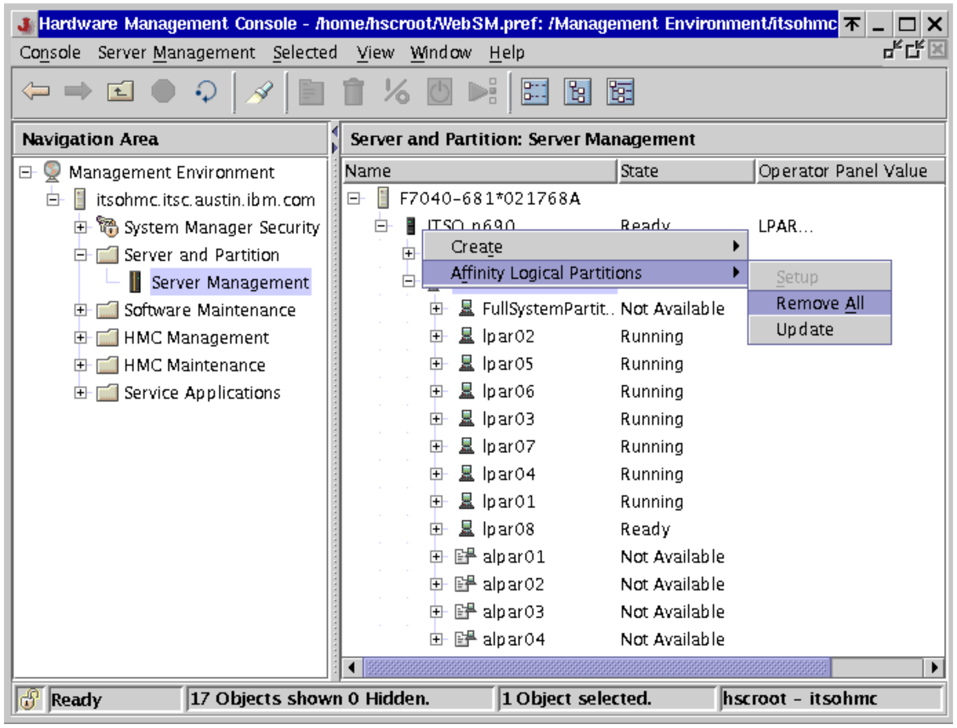
6.2.3 Delete all affinity logical partitions

To remove all the affinity logical partitions, do the following:

Note: You cannot remove affinity logical partitions individually.

1. In the Contents area, select the partition icon under the managed system. Right-click the partition icon.

2. Select **Affinity Logical Partitions**, and then **Remove All**, as shown in Figure 6-18.

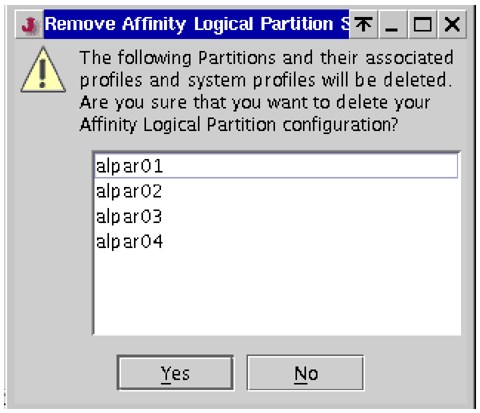

Hardware Management Console - /home/hscroot/WebSM.pref: /Management Environment/itsohmc
Console Server Management Selected View Window Help
Navigation Area
Management Environment
itsohmc.itsc.austin.ibm.com
System Manager Security
Server and Partition
Server Management
Software Maintenance
HMC Management
HMC Maintenance
Service Applications
Server and Partition: Server Management
Create
Affinity Logical Partitions
Setup
Remove All
Update

Name	State	Operator Panel Value
F7040-681*021768A		
ITSO_p690	Ready	LPAR...
FullSystemPartit..	Not Available	
lpar02	Running	
lpar05	Running	
lpar06	Running	
lpar03	Running	
lpar07	Running	
lpar04	Running	
lpar01	Running	
lpar08	Ready	
alpar01	Not Available	
alpar02	Not Available	
alpar03	Not Available	
alpar04	Not Available	

Ready | 17 Objects shown 0 Hidden. | 1 Object selected. | hscroot - itsohmc

Figure 6-18 Remove all affinity logical partitions

3. Click **Yes** in the Remove Affinity Logical Partition window, as shown in Figure 6-19.



The following Partitions and their associated profiles and system profiles will be deleted. Are you sure that you want to delete your Affinity Logical Partition configuration?

alpar01
alpar02
alpar03
alpar04

Figure 6-19 Confirm the removal of all the affinity logical partitions

This operation removes the AffinityPartition system profile and all the affinity logical partitions.

6.3 Activate partitions

If you *activate* a partition, you are virtually powering on the partition. To activate a partition, select the partition name and select activate by right-clicking. This opens a window that enables you to choose the profile that you want to activate for this partition. If the minimum and required resources you specified when you created the partition profile exceeds the amount of available resources, this partition will not be activated with the selected profile. Available resources are all resources currently not being used by other active partitions. It is important that you keep track of your system's resources at all times.

6.3.1 Change the default partition profile

When a partition is created, a profile also has to be created by default to define the resources associated with this partition. The application requires that you create at least one profile when a partition is created. The first profile created is the *default* profile. Additionally, the default partition profile is marked with an icon. The default partition profile can be changed at any time. To change the default partition profile, select the partition profile name in the Contents area, from the menu bar choose **Selected** → **Change Default Profile**, and select the profile name from the list that you want to make the default. This operation can also be completed by selecting the profile name, right-clicking, and following the menus.

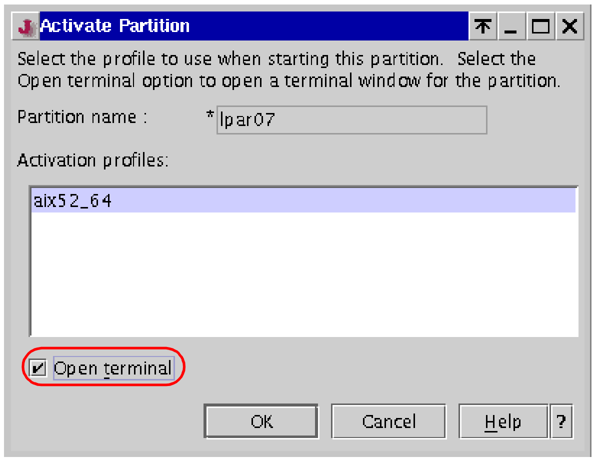
The default profile can be changed even when the partition is in the active state with the profile running.

6.3.2 Activate a specific partition profile

To activate a partition profile, select one of the partition profiles you created, and from the menu bar choose **Selected** → **Activate**. The profile name is highlighted. Click **OK** to activate the partition using this partition profile. If you want to activate using the other partition profiles, select another profile in the list and then click **OK**. This operation can also be accomplished by selecting the desired profile name, right-clicking, and selecting the **Activate** option.

6.3.3 Activate partitions without selecting a specific partition profile

To activate a partition without selecting a specific partition profile, select the partition in the Contents area, and from the menu bar choose **Selected** → **Activate**. The default profile name is highlighted as shown in Figure 6-20. Then, click **OK**.


Activate Partition

Select the profile to use when starting this partition. Select the Open terminal option to open a terminal window for the partition.

Partition name : * lpar07

Activation profiles:

aix52_64

☑ Open terminal

Figure 6-20 Activate a partition

If you select the **Open terminal** check box (highlighted in Figure 6-20), a virtual terminal window opens upon activation of the partition.

6.3.4 Reactivating a partition with a different partition profile

To reactivate a partition with a different profile, select the partition for which you want to change profiles in the Contents area. Open a virtual terminal window for that partition to log in to the operating system, and then issue an appropriate operating system shutdown command.[2] The system shuts down the operating system, and the partition's state changes from `Running` to `Ready` in the Contents area. In the Contents area, select the new partition profile you want to activate for that partition. From the menu bar choose **Selected** → **Activate**, or select the profile that you want to activate, right-click, and then select the **Activate** option.

6.3.5 Partition operating states

In the column to the right of the names of the partitions in the Contents area, the HMC indicates the operating status of the partitions. Table 6-2 lists all possible partition operating states.

Table 6-2 Operating states of partitions

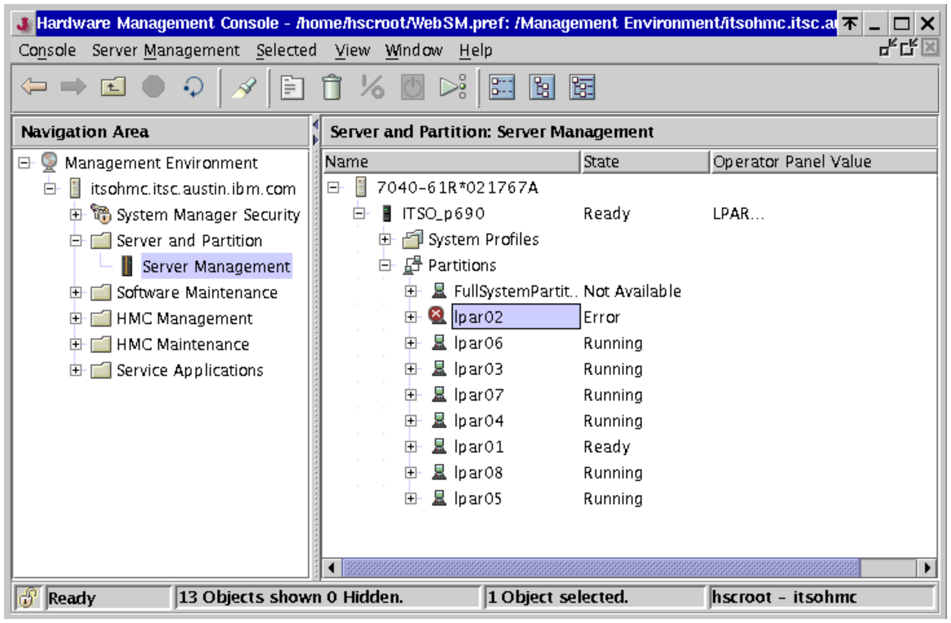
Operating state	**Description**
Ready	The partition is not active, but is ready to be activated.
Starting	The partition is activated and is undergoing booting routines.
Running	The partition has finished its booting routines. The operating system can be performing its booting routines or is in its normal running state.
Error	Activation of this partition failed due to a hardware or operating system error.

[2] As we will explain in 6.4, "Shut down the operating system in a partition" on page 196, the HMC software level Release 3, Version 2 provides the operating system shutdown menu, if the target partition is installed with AIX 5L Version 5.2 and 5200-01 Recommended Maintenance Level and later.

Operating state	Description
Not available	This partition is not available for use. Reasons can include: ► The managed system is powered off. ► The Full System Partition is not available when the managed system is powered on with the Partition Standby power-on option. ► Partitions are not available when the managed system is powered on with the Full System Partition power-on option. ► Affinity partitions are not available when the managed system is powered on and the non-affinity partitions are activated first. ► Non-affinity partitions are not available when the managed system is powered on and affinity partitions are powered on first.
Open Firmware	The partition was activated by a profile that specified an OPEN_FIRMWARE boot mode.

If the partition operation state is `Error` after you attempt to activate it, you can select **Read Boot Error Value** to understand why the partition gets an error during the boot.

For example, if you have set the service authority to more than one partition and tried to activate the second partition with the authority, then the activation would fail with the Error state as shown in Figure 6-21 on page 196.

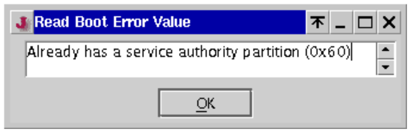

Hardware Management Console - /home/hscroot/WebSM.pref: /Management Environment/itsohmc.itsc.a
Console Server Management Selected View Window Help
Navigation Area
Management Environment
itsohmc.itsc.austin.ibm.com
System Manager Security
Server and Partition
Server Management
Software Maintenance
HMC Management
HMC Maintenance
Service Applications
Server and Partition: Server Management

Name	State	Operator Panel Value
7040-61R*021767A		
ITSO_p690	Ready	LPAR...
System Profiles		
Partitions		
FullSystemPartit..	Not Available	
lpar02	Error	
lpar06	Running	
lpar03	Running	
lpar07	Running	
lpar04	Running	
lpar01	Ready	
lpar08	Running	
lpar05	Running	

Ready | 13 Objects shown 0 Hidden. | 1 Object selected. | hscroot - itsohmc

Figure 6-21 Partition activation failure

In this case, the boot error message shown in Figure 6-22 explains the reason for the failure of this partition activation.



Already has a service authority partition (0x60)

Figure 6-22 Read Boot Error Values

6.4 Shut down the operating system in a partition

To shut down the operating system in a partition, do the following:

1. In the Contents area, select the partition you want to shut down.
2. From the menu bar choose **Selected** → **Operating System** → **Shutdown**.

This function is available when the following requirements are met:

- HMC is installed with software Release 3, Version 2 and later.[3]

[3] The menu does not exist on the HMC installed with software Release 3, Version 1 and before.

- The target partition is installed with AIX 5L Version 5.2 plus 5200-01 Recommended Maintenance Level and later.[4]

You can also perform this operation by selecting the partition name and right-clicking to display the window shown Figure 6-23.

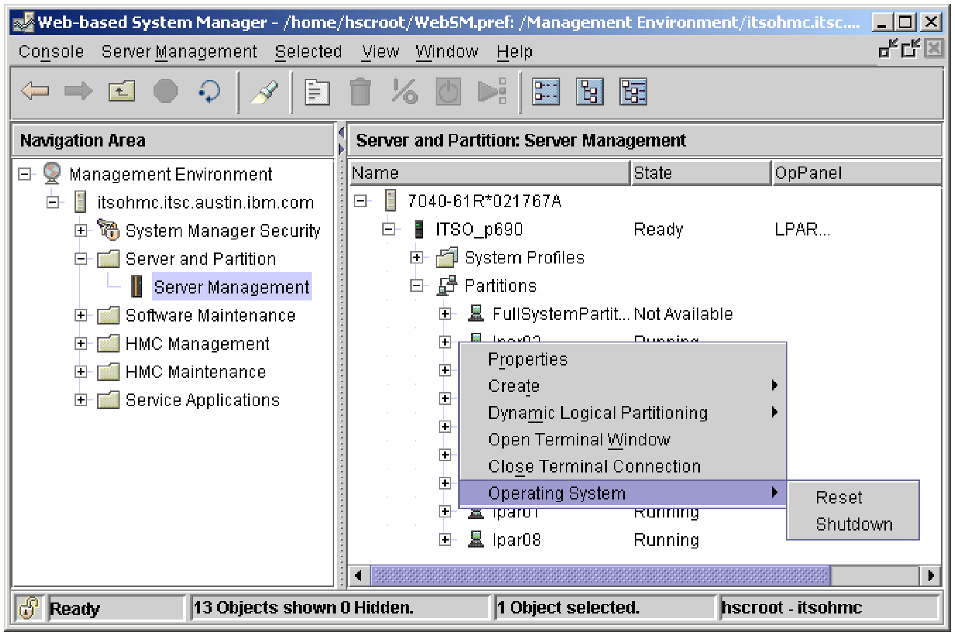
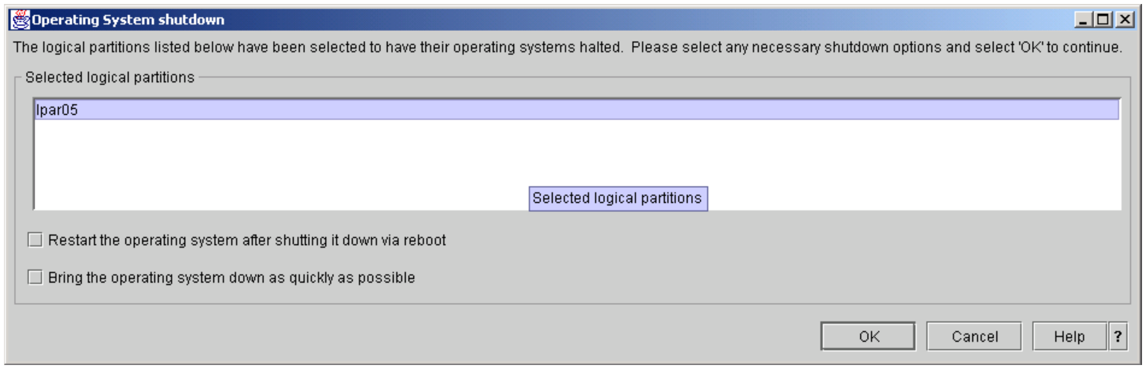


Figure 6-23 Operating System shutdown or reset

A dialog box shown in Figure 6-24 on page 198 will appear. Select the partition name and click **OK** to shut down the operating system in the selected partition. You may select the following options in the dialog box before clicking **OK**:

- Restart the operating system after shutting it down via reboot.
- Bring the operating system down as quickly as possible.

[4] If the partition is not installed with AIX 5L Version 5.2 plus 5200-01 Recommended Maintenance Level and later, the **Shutdown** menu selection is grayed out and unselectable.


The logical partitions listed below have been selected to have their operating systems halted. Please select any necessary shutdown options and select 'OK' to continue.

Selected logical partitions

lpar05

Selected logical partitions

☐ Restart the operating system after shutting it down via reboot

☐ Bring the operating system down as quickly as possible

Figure 6-24 Operating System shutdown

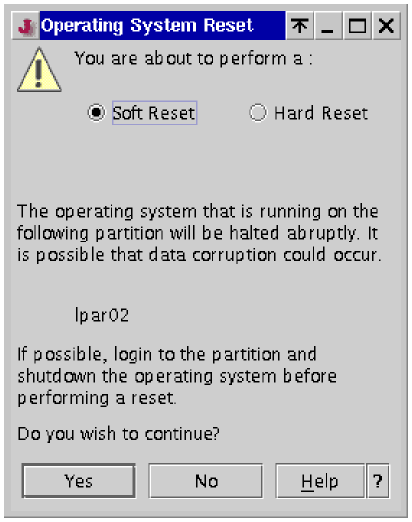
6.5 Reset the operating system in a partition

When an operating system in a partition stalls, you can use the HMC to restart the operating system.

Important: This operation may corrupt data on the resetting partition. Perform this procedure only after you have attempted to restart the operating system manually.

In the Contents area, select the partition you want to reset. From the menu bar choose **Selected** → **Operating System** → **Reset**. You can also perform this operation by selecting the partition name and right-clicking to display the window shown in Figure 6-23 on page 197.

A dialog box opens that offers two reset options, as shown in Figure 6-25 on page 199.


Operating System Reset

You are about to perform a :

[✓ Soft Reset] Hard Reset

The operating system that is running on the following partition will be halted abruptly. It is possible that data corruption could occur.

lpar02

If possible, login to the partition and shutdown the operating system before performing a reset.

Do you wish to continue?

Figure 6-25 Operating system reset options

The following operating system reset options are available:

Soft Reset	The actions of the operating system after a soft reset are determined by its policy settings. Depending on how you have configured these settings, the operating system may perform a dump of system information or will restart automatically. For more information about configuring your operating system's policy settings, refer to its supporting documentation.
Hard Reset	A hard reset acts as a virtual powering off of the partition, not the managed system. Issuing a hard reset forces termination and can corrupt information. Use this option only if the operating system is disrupted and cannot send or receive commands.

If the "Power off the system after all the logical partitions are powered off" policy is set shown in Figure 5-3 on page 155, and if you deactivate the last partition in the system (no partition is activated in the system), the managed system is powered off. The status of the managed system is changed from `LPAR…` to `No Power`.

7

Installing and migrating AIX in a partitioned environment

This chapter explains several AIX installation and migration methods in a partitioned environment in the following sections:

- Installing AIX on partioning-capable pSeries servers
- Installing AIX using removable media devices
- Installing AIX using Network Installation Manager (NIM)
- Installing AIX using alternate disk install
- Migrating AIX using alternate disk migration

For further information about AIX installation, please refer to the following publications, available at:

`http://techsupport.services.ibm.com/server/library`

- *AIX 5L Version 5.2 Installation Guide and Reference*
- *AIX 5L Version 5.2 Installation in a Partitioned Environment*.

Note: Throughout this redbook, we use the term *CD/DVD devices* to refer to CD-ROM, DVD-RAM, and DVD-ROM devices.

7.1 Installing AIX on partioning-capable pSeries servers

Installing AIX on partioning-capable pSeries servers is not a difficult task although you must understand the following considerations when installing AIX in partitions:

- Activating partitions

  In order to activate (virtually power on) partitions, you must use the HMC (see 6.3, “Activate partitions” on page 192).

- Console of partition upon installation

  In most cases, you have to use the virtual terminal window on the HMC for the console of the partition upon operating system install (see 4.5, “Virtual terminal window” on page 146).

- Install source device

  Although AIX supports various media as install source devices, such as CD/DVD, tape, over the network (see 7.3, “Installing AIX using Network Installation Manager (NIM)” on page 218), and disk file on file systems (see 7.4, “Installing AIX using alternate disk install” on page 231), you need to assign at least one removable media device to the partition for the first operating system installation to the system. Because a partioning-capable pSeries server is usually equipped with one or two removable media devices, you need to carefully plan the assignment of these devices to partitions.

  Note: The dynamic logical partitioning function in conjunction with AIX 5L Version 5.2 gives you greater flexibility in the assignment of these devices to partitions. We explain how to exploit the dynamic logical partitioning function for the AIX installation in 9.3, “Dynamic logical partitioning operations using chhwres” on page 285.

In fact, if you are going to newly install AIX on partitions on a partioning-capable pSeries server from scratch (this means that you do not have to worry about the existing data on the install disks), you can simply complete the following steps:

1. Boot up the managed system with the Full System Partition (see 5.2, “Power on the managed system” on page 157).
2. Install AIX from CD/DVD or tape (if you have an already customized install image bootable tape for the installation purpose) onto the Full System Partition (see 7.1.1, “Install AIX in the Full System Partition” on page 204).
3. Perform common customization on AIX, for example, installing common software components.

4. Clone the AIX image using the alternate disk install facility (see 7.4, “Installing AIX using alternate disk install” on page 231).
5. Shut down AIX and power off the managed system (see 5.3, “Power off the managed system” on page 161).
6. Boot up the managed system in the Partition Standby mode (see 5.2, “Power on the managed system” on page 157).
7. Configure the partitions (see 6.1, “Partition and system profile tasks” on page 166).
8. Activate (boot up) partitions (see 6.3, “Activate partitions” on page 192).
9. Perform any necessary additional configurations on partitions, such as host name, and TCP/IP address.

Note: These steps are the fastest way to newly install AIX onto all the partitions as the initial operating system loading task, regardless of the version of AIX.

Comparison of AIX installation methods

Each AIX installation method discussed in this chapter has its own merits and limitations:

- Installing AIX using removable media devices

  This is probably the easiest method, but may not be so convenient if you have to install more than two or three partitions.

- Installing AIX using Network Installation Manager (NIM)

  This is the most powerful and flexible method. It used to be difficult to configure and maintain NIM, but now it is quite easy to configure NIM in the recent AIX releases. Even though you need a machine or a partition to act as a NIM master, it is worthwhile to do if you have to administer several partitions.

- Installing AIX using alternate disk install

  This is probably the most suitable method if you want to *clone* the partitions, having partitions with similar configurations and filesets installed, for example, creating partitions for Web server clusters.

  The strength of this method is that it can be fully automated. There is almost no manual steps to do as long as the partition is loaded with AIX 5L Version 5.2 or later.

It would not be uncommon to find that in certain situations, you may need to use more than one method. For example, using CD-ROM to install the first partition, and then using alternate disk install to clone it to the other partitions that compose Web server clusters.

7.1.1 Install AIX in the Full System Partition

The installation of AIX in the Full System Partition is quite straightforward. It is performed as if on an existing pSeries system, except for the powering on and console explained previously.

After booting up the managed system with the Full System Partition (see 5.2, “Power on the managed system” on page 157), you can install AIX using the methods explained in the following sections:

- 7.2.1, “Boot AIX from removable media devices” on page 207
- 7.3.3, “Boot partitions over the network” on page 224

There is no difference between installing AIX in the Full System Partition and partitions except for the fact that the Full System Partition accesses all the hardware resources on the managed system.

7.1.2 Create a partition-ready AIX mksysb image

AIX can create the operating system install image (*mksysb image*) on the following media:

- CD/DVD
- Tape (4 mm or 8 mm)
- Disk file on file systems

AIX 5L Version 5.2 supports the creation of a bootable mksysb image on the DVD-RAM media formatted by Universal Disk Format (UDF), in addition to the ISO 9660 format supported by AIX 5L Version 5.1. For further information about how to create the bootable UDF DVD-RAM media, please refer to *AIX 5L Differences Guide*, SG24-5765. For further information about how to create the bootable ISO 9660 CD-R or DVD-RAM media, please refer to *Managing AIX Server Farms*, SG24-6606.

Note: Depending on the partioning-capable pSeries server models, you may need to apply additional APARs before creating a partition-ready mksysb (see 1.2, “Partitioning support on pSeries servers” on page 5).

If you have to use a partitioning-*incapable* pSeries server to create a valid mksysb image (called a *partition-ready* mksysb image) that can be used for installing to a partition, you must do the following:

Note: If you create a mksysb on a partition, it is always partition-ready.

1. If you are using AIX Version 4.3.3 or earlier, you have to migrate it to AIX 5L Version 5.1 first on a separate pSeries server, because the minimum supported software level in a partitioned environment is AIX 5L Version 5.1 with 5100-01 Recommended Maintenance Level plus APAR IY39794. We strongly recommend you test your applications on that environment.
2. If you are using AIX 5L Version 5.2, the mksysb will be always partition-ready unless you have explicitly uninstalled the following filesets from the system:
   - devices.chrp.base.rte
   - devices.chrp.base.ServiceRM
   - devices.chrp_lpar.base.ras
   - devices.chrp_lpar.base.rte

   Skip to step 4.

   Note: On AIX 5L Version 5.2, all device filesets are installed by default. Therefore, all AIX 5L Version 5.2 partitions should have the necessary device support for any adapters that can be added by DLPAR operations.

3. If you are using AIX 5L Version 5.1 *without* the 5100-01 Recommended Maintenance Level plus APAR IY39794, you have to apply this package using the following steps:

   Note: We strongly recommend you install the latest Recommended Maintenance Level (RML). At the time of writing of this redbook, the latest RML for AIX 5L Version 5.1is 5100-03.

   a. Insert the CD-ROM media that contains the 5100-01 Recommended Maintenance Level plus APAR IY39794 into the CD/DVD device.
   b. Type the following command:

   ```
   # installp -acX -d /dev/cd0 bos.rte.install
   ```

   It brings the bos.rte.install fileset level to 5.1.0.15:

   ```
   # lslpp -L bos.rte.install
     Fileset                 Level  State  Type  Description (Uninstaller)

   ----------------------------------------------------------------------
     bos.rte.install         5.1.0.15   C  F     LPP Install Commands
   ```

   c. Confirm that the APAR IY39794 was applied correctly:

   ```
   # instfix -ik IY22854
   ```

It should print the following output:

```
All filesets for IY22854 were found.
```

d. Issue the **`partition_ready`** command.

This command looks for update images in the first available CD/DVD device (usually /dev/cd0). If it does not find an update CD-ROM medium in the device, it prompts the user to insert one. After the **`partition_ready`** command verifies that the update images are valid, it updates all filesets if needed. This command also installs any new device drivers on the update media at the latest level. After the update, the **`partition_ready`** command verifies that all filesets were installed at the appropriate levels. If any filesets are still down a level, it specifies the names of these filesets, the levels at which they are actually installed, and the required levels. All **`partition_ready`** command activities are logged in the /var/adm/ras/partition_ready.log log.

For detailed information about this command, please refer to /usr/lpp/bos/README.PARTITION_INSTALL, which is installed with AIX 5L Version 5.1 with 5100-01 Recommended Maintenance Level plus APAR IY39794.

4. Create a partition-ready mksysb image.

– If you create a mksysb file, you can use the SMIT shortcut **`smitty mksysb`**, specifying a file name as a target file. This mksysb file can be used for NIM, alternate disk install, or creating a bootable CD/DVD media.

– If you use a tape device to create a bootable mksysb image tape media, you can use the SMIT shortcut **`smitty mksysb`** with a tape drive as a target device.

– If you use a CD/DVD device to create a bootable mksysb image CD/DVD media, select the following SMIT panels:

```
# smitty
    System Storage Management (Physical & Logical Storage)
        System Backup Manager
            Back Up the System
```

Then select either of the following, depending on the device type:

CD-R/CD-RW	Back up this system to CD.
DVD-RAM	Back up this system to DVD[1].

Note: The DVD-ROM cannot be used for creating the mksysb media.

[1] You can select the UDF option on AIX 5L Version 5.2.

7.2 Installing AIX using removable media devices

In a partitioned environment, you can use a removable media device to install AIX in partitions, but the removable media device can be used by only one partition at a time, because it is controlled by one SCSI adapter, which is assignable to only one partition at a time.

If the partition that currently holds the removable media device supports dynamic logical partitioning, then whenever you need a removable media device, you can remove the device by using either the graphical user interface (see 8.1, “Dynamic logical partitioning” on page 256) or command line interface (see 9.3, “Dynamic logical partitioning operations using chhwres” on page 285) on the HMC.

If the partition that currently holds the removable media device does not support dynamic logical partitioning, then whenever you need a removable media device, shut down and deactivate the partition that has the removable media device currently. Then, activate and start another partition that requires the removable media device for software installation purposes. Assigning the I/O slots that contain the SCSI adapter connected to the removable media device to a partition as a desired resource is helpful for software installation purposes.

We use the DVD-RAM drive equipped in the media drawer of pSeries 690 as an example configuration of an installation scenario from removable media devices (see Appendix A, “Test environment” on page 389). However, the method explained here should be applicable for all removable media device types on any partioning-capable pSeries servers as long as the removable media device is controlled by a SCSI adapter. The IDE DVD-ROM drive (FC 2634) for the pSeries 670 and pSeries 690 is also applicable for this method, because it is connected to a SCSI adapter using the IDE media to LVD SCSI interface bridge card (FC 4253) in the media drawer.

7.2.1 Boot AIX from removable media devices

To install AIX in a partition using the removable media device, do the following:

1. Verify if another partition uses the removable media device.

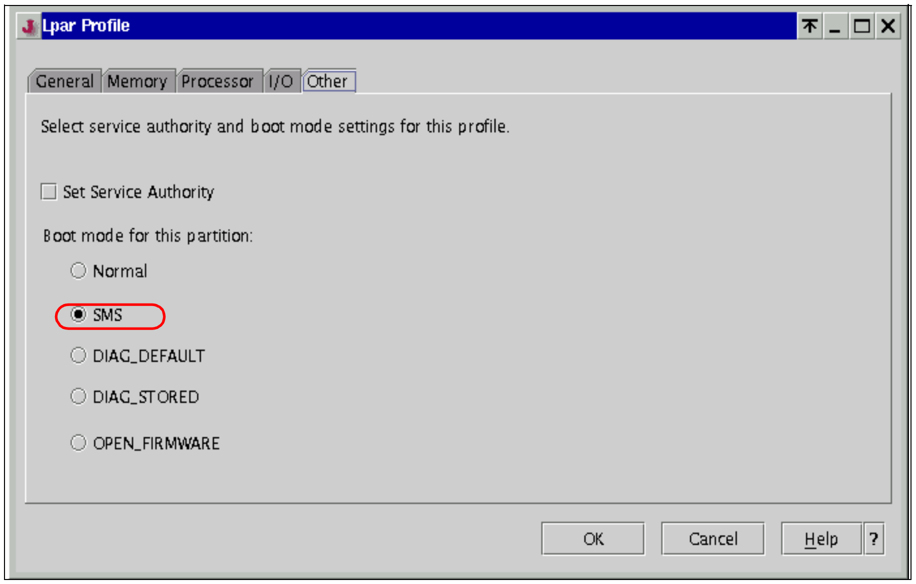
   If the device is used by another partition, do either of the following depending the partition’s capability:

   – If this partition does not support dynamic logical partitioning, shut down the partition that uses the removable media device when possible. You might have to wait before the partition can be shut down.

   – If this partition supports dynamic logical partitioning, remove the removable media device by performing a DLPAR operation (see 8.1.3, “Dynamically removing resources from a partition” on page 269).

2. Create a partition and partition profile on the HMC. Assign the SCSI adapter attached to the removable media device.
3. Set the boot mode for this partition to Server Management Services (SMS).

   To set the boot mode to SMS, select the partition profile, right-click, select **Properties**. Select the Others tab in the properties panel of the partition profile, and then select **SMS**, as shown in Figure 7-1.



Select service authority and boot mode settings for this profile.

☐ Set Service Authority

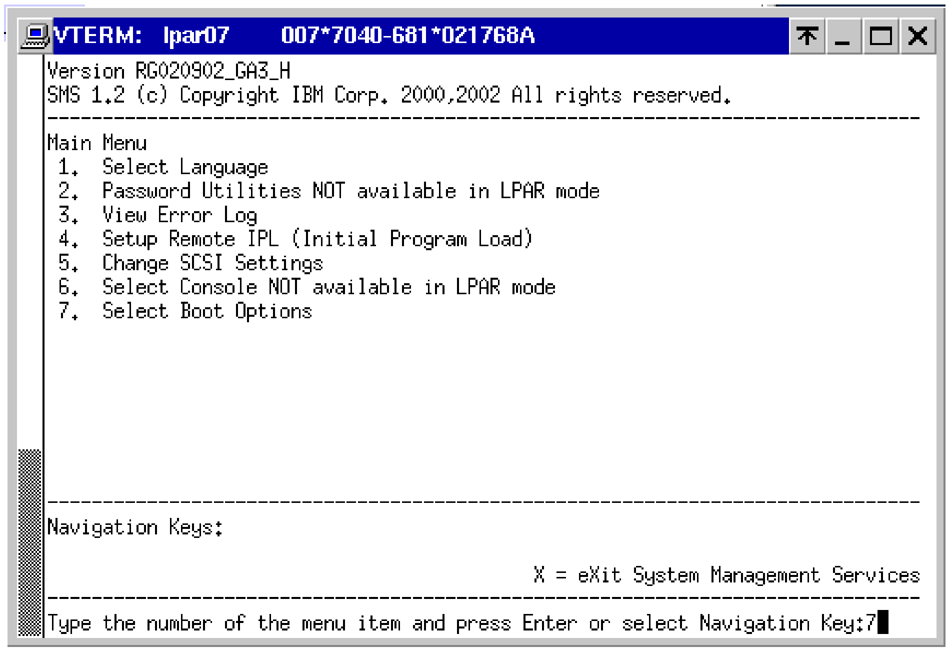
Boot mode for this partition:

○ Normal
● SMS
○ DIAG_DEFAULT
○ DIAG_STORED
○ OPEN_FIRMWARE

Figure 7-1 Set the boot mode to SMS

4. After you have successfully created the partition or changed the partition profile, leave the partition in the Ready state.
5. Place the bootable AIX installation media[2] in the removable media device in the media drawer.
6. Right-click the partition on which you are going to install AIX.
7. Select **Activate**. The Activate Partition menu opens with a selection of partition profiles. Be sure the correct profile is highlighted.
8. Select the **Open Terminal** check box at the bottom of the Activate Partition panel to open a virtual terminal window (see Figure 6-20 on page 193), and then click **OK**. A virtual terminal window panel opens for the partition. After several seconds, the SMS main menu opens in the virtual terminal window.
9. In the SMS menu, press the 7 key to select **7. Select Boot Options**, as shown in Figure 7-2.

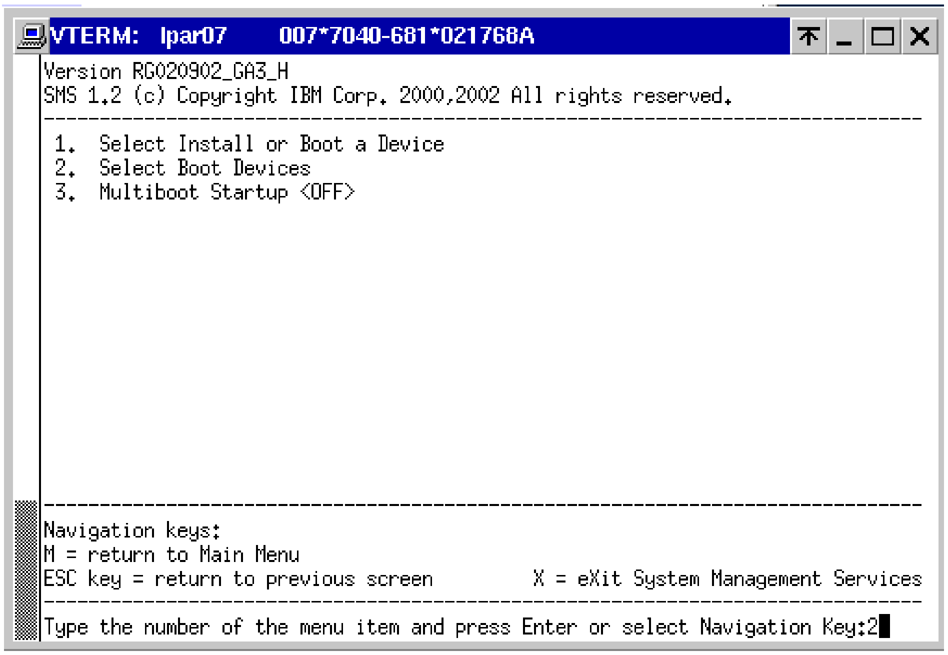
[2] If you are using the AIX product media, insert the CD-ROM media volume 1.



```
Version RG020902_GA3_H
SMS 1.2 (c) Copyright IBM Corp. 2000,2002 All rights reserved.
-------------------------------------------------------------------------------
Main Menu
 1.   Select Language
 2.   Password Utilities NOT available in LPAR mode
 3.   View Error Log
 4.   Setup Remote IPL (Initial Program Load)
 5.   Change SCSI Settings
 6.   Select Console NOT available in LPAR mode
 7.   Select Boot Options








-------------------------------------------------------------------------------
Navigation Keys:

                                            X = eXit System Management Services
-------------------------------------------------------------------------------
Type the number of the menu item and press Enter or select Navigation Key:7
```

Figure 7-2   System Management Services: Main menu

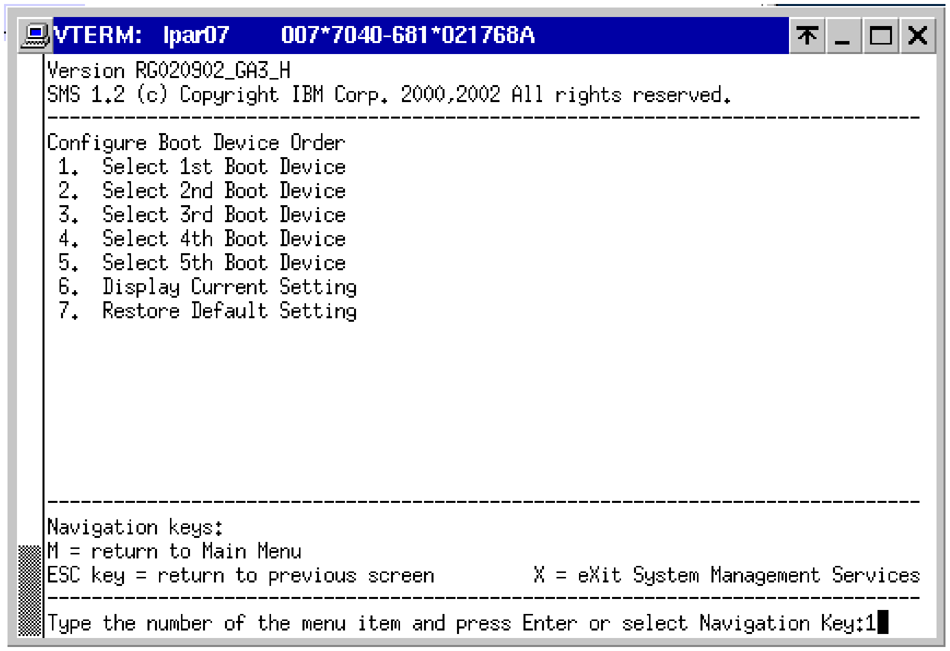
10. Press 2 to select **2. Select Boot Device**, as shown in Figure 7-3.



```
Version RG020902_GA3_H
SMS 1.2 (c) Copyright IBM Corp. 2000,2002 All rights reserved.
-------------------------------------------------------------------------------
 1.  Select Install or Boot a Device
 2.  Select Boot Devices
 3.  Multiboot Startup <OFF>









-------------------------------------------------------------------------------
Navigation keys:
M = return to Main Menu
ESC key = return to previous screen          X = eXit System Management Services
-------------------------------------------------------------------------------
Type the number of the menu item and press Enter or select Navigation Key:2
```

Figure 7-3 System Management Services: Select boot options

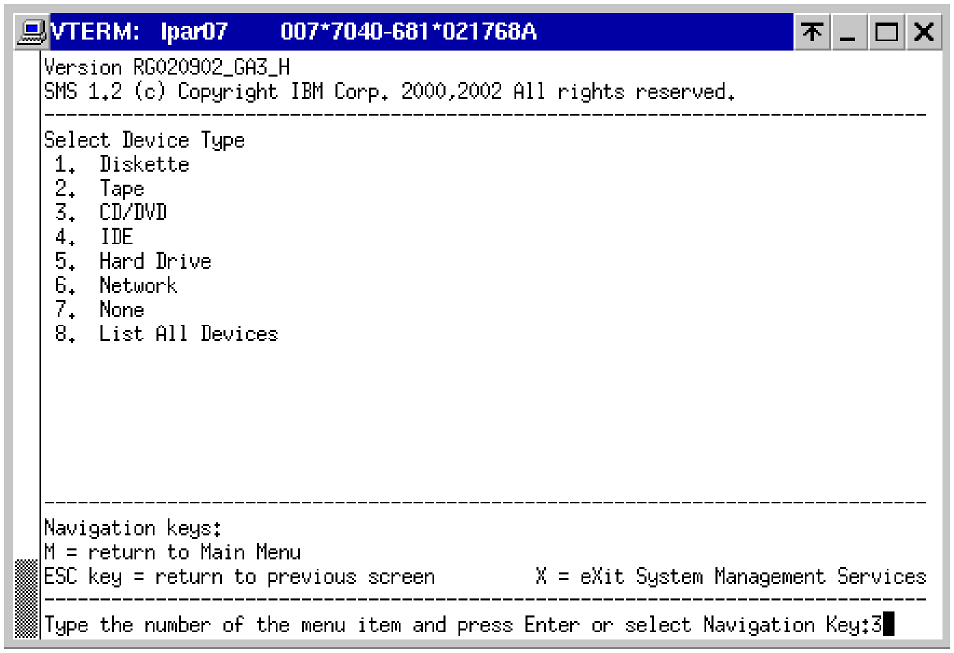
11. Press 1 to select **1. Select 1st Boot Devices**, as shown in Figure 7-4.



```
Version RG020902_GA3_H
SMS 1.2 (c) Copyright IBM Corp. 2000,2002 All rights reserved.
-------------------------------------------------------------------------------
Configure Boot Device Order
 1.  Select 1st Boot Device
 2.  Select 2nd Boot Device
 3.  Select 3rd Boot Device
 4.  Select 4th Boot Device
 5.  Select 5th Boot Device
 6.  Display Current Setting
 7.  Restore Default Setting









-------------------------------------------------------------------------------
Navigation keys:
M = return to Main Menu
ESC key = return to previous screen         X = eXit System Management Services
-------------------------------------------------------------------------------
Type the number of the menu item and press Enter or select Navigation Key:1
```

Figure 7-4 System Management Services: Configure boot device

12. Press 3 to select **3. CD/DVD**, as shown in Figure 7-5.

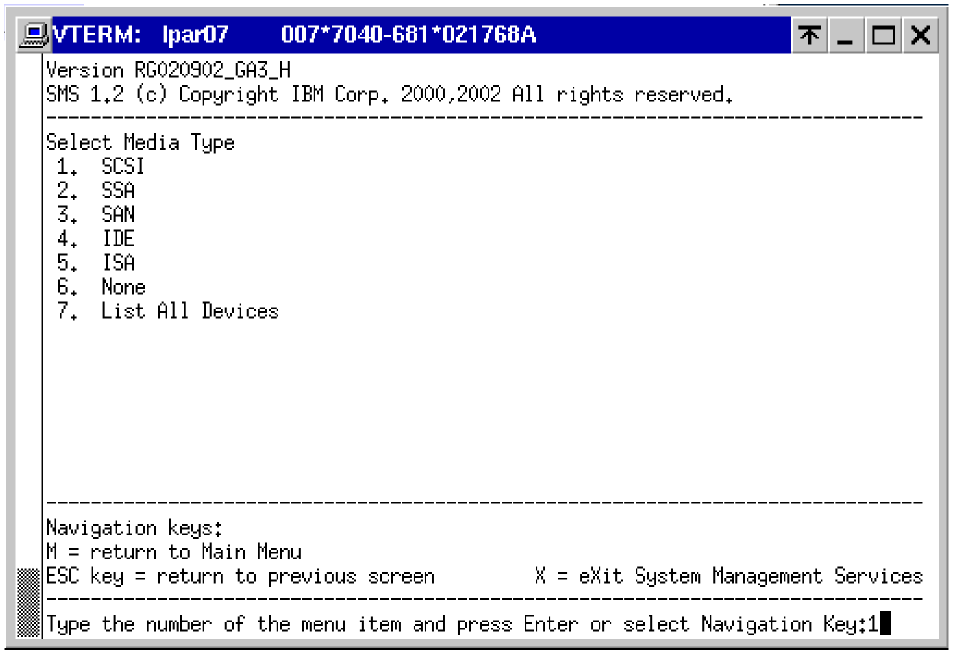


VTERM: lpar07 007*7040-681*021768A

```
Version RG020902_GA3_H
SMS 1.2 (c) Copyright IBM Corp. 2000,2002 All rights reserved.
-------------------------------------------------------------------------------
Select Device Type
 1.   Diskette
 2.   Tape
 3.   CD/DVD
 4.   IDE
 5.   Hard Drive
 6.   Network
 7.   None
 8.   List All Devices







-------------------------------------------------------------------------------
Navigation keys:
M = return to Main Menu
ESC key = return to previous screen         X = eXit System Management Services
-------------------------------------------------------------------------------
Type the number of the menu item and press Enter or select Navigation Key:3
```

Figure 7-5   System Management Services: Select device type

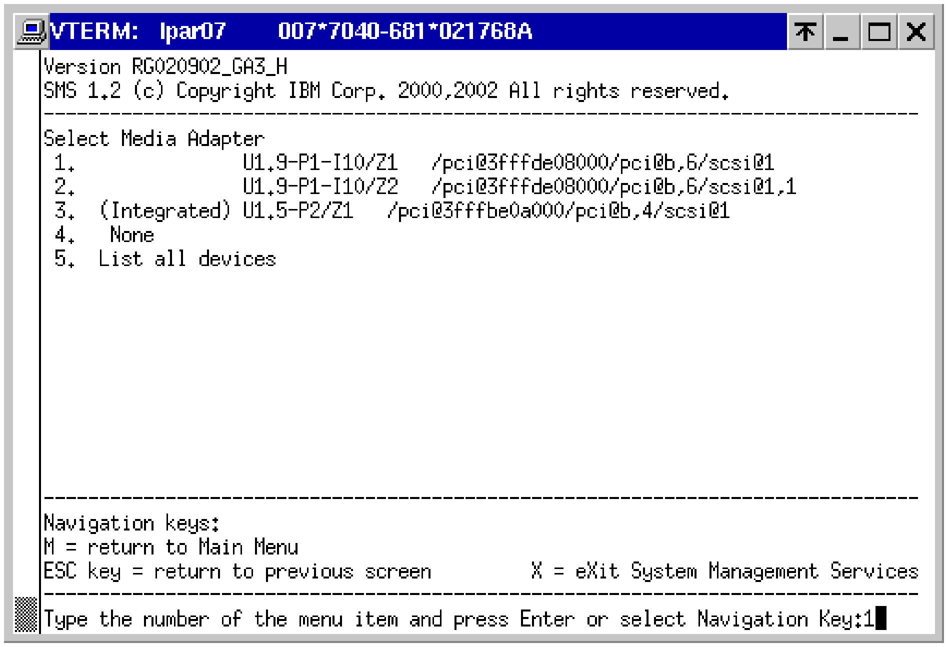
13. Type 1 to select **1. SCSI**, as shown in Figure 7-6.



```
VTERM:  lpar07      007*7040-681*021768A

Version RG020902_GA3_H
SMS 1.2 (c) Copyright IBM Corp. 2000,2002 All rights reserved.
-------------------------------------------------------------------------------
Select Media Type
 1.   SCSI
 2.   SSA
 3.   SAN
 4.   IDE
 5.   ISA
 6.   None
 7.   List All Devices








-------------------------------------------------------------------------------
Navigation keys:
M = return to Main Menu
ESC key = return to previous screen         X = eXit System Management Services
-------------------------------------------------------------------------------
Type the number of the menu item and press Enter or select Navigation Key:1
```

Figure 7-6   System Management Services: Select media type

14. Press the appropriate key to select the SCSI adapter to which the CD/DVD device is connected. In this example, we selected 1, as shown in Figure 7-7.

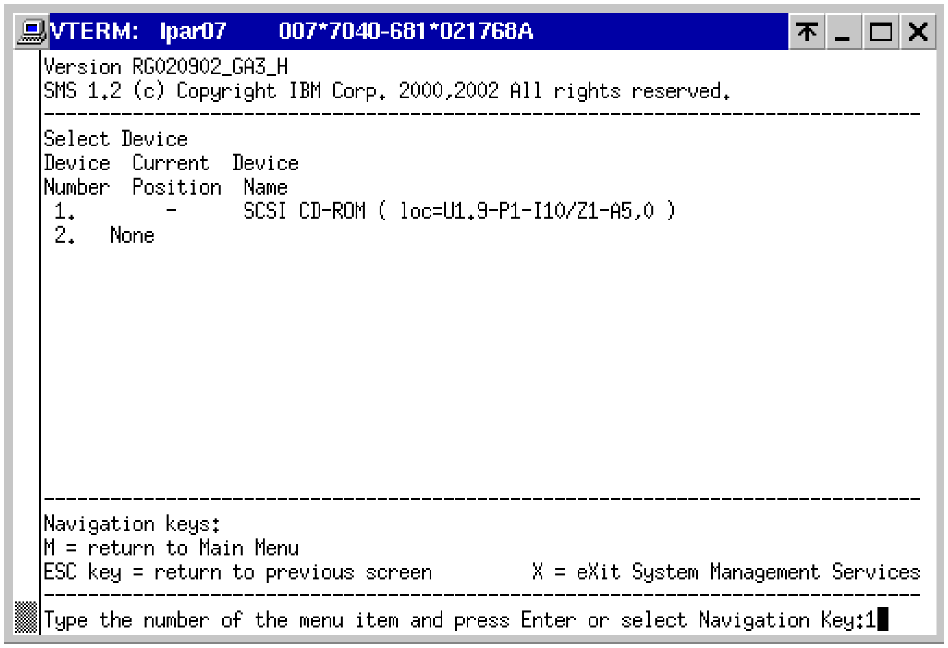


```
Version RG020902_GA3_H
SMS 1.2 (c) Copyright IBM Corp. 2000,2002 All rights reserved.
-------------------------------------------------------------------------------
Select Media Adapter
 1.              U1.9-P1-I10/Z1   /pci@3fffde08000/pci@b,6/scsi@1
 2.              U1.9-P1-I10/Z2   /pci@3fffde08000/pci@b,6/scsi@1,1
 3.  (Integrated) U1.5-P2/Z1   /pci@3fffbe0a000/pci@b,4/scsi@1
 4.   None
 5.  List all devices










-------------------------------------------------------------------------------
Navigation keys:
M = return to Main Menu
ESC key = return to previous screen         X = eXit System Management Services
-------------------------------------------------------------------------------
Type the number of the menu item and press Enter or select Navigation Key:1
```

Figure 7-7 System Management Services: Select media adapter

Note: As explained in Appendix A, “Test environment” on page 389, the DVD-RAM drive in the media drawer is connected to the SCSI adapter that is inserted into the tenth I/O slot in the left-half (planar 1) of the first I/O drawer. Because FC 6204, which is used to connect to the media drawer, has two separate SCSI ports, the first port (on lower one in the I/O drawer) is represented as /pci@3fffde0800/pci@b,6/scsi@1, and the second one is represented as /pci@3fffde0800/pci@b.6/scsi@1,1. You must understand which SCSI port is used to connect to the media drawer before selecting the option in Figure 7-7.

15. Press the appropriate key to select the CD/DVD device. In this example, we selected 1, as shown in Figure 7-8.

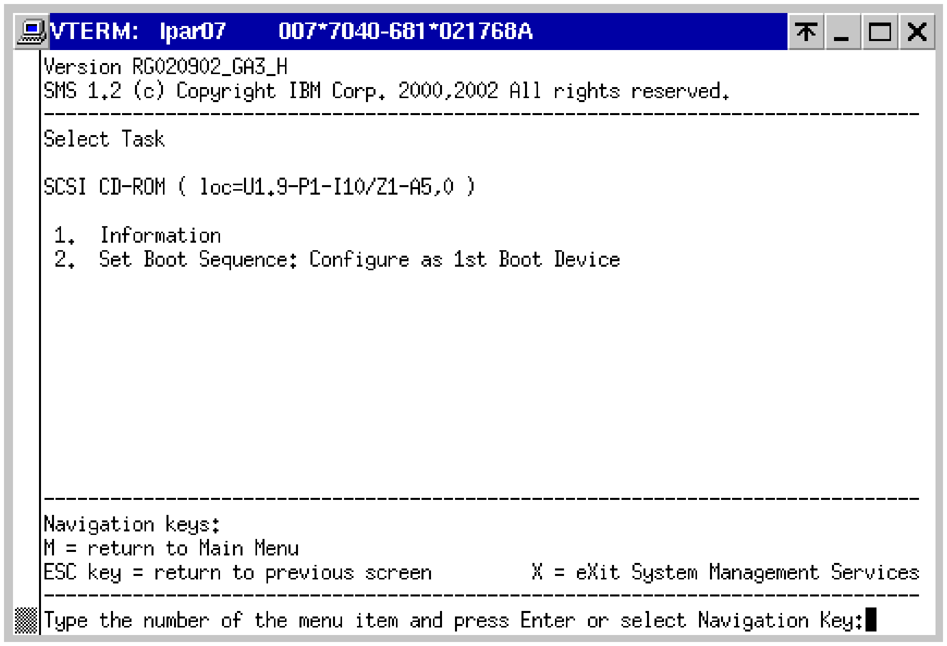


```
Version RG020902_GA3_H
SMS 1.2 (c) Copyright IBM Corp. 2000,2002 All rights reserved.
-------------------------------------------------------------------------------
Select Device
Device  Current  Device
Number  Position  Name
 1.        -      SCSI CD-ROM ( loc=U1.9-P1-I10/Z1-A5,0 )
 2.    None








-------------------------------------------------------------------------------
Navigation keys:
M = return to Main Menu
ESC key = return to previous screen         X = eXit System Management Services
-------------------------------------------------------------------------------
Type the number of the menu item and press Enter or select Navigation Key:1
```

Figure 7-8 System Management Services: Select device

Note: The CD/DVD device is shown as CD-ROM regardless of the actual media type (CD-ROM, DVD-RAM, or DVD-ROM) in this menu.

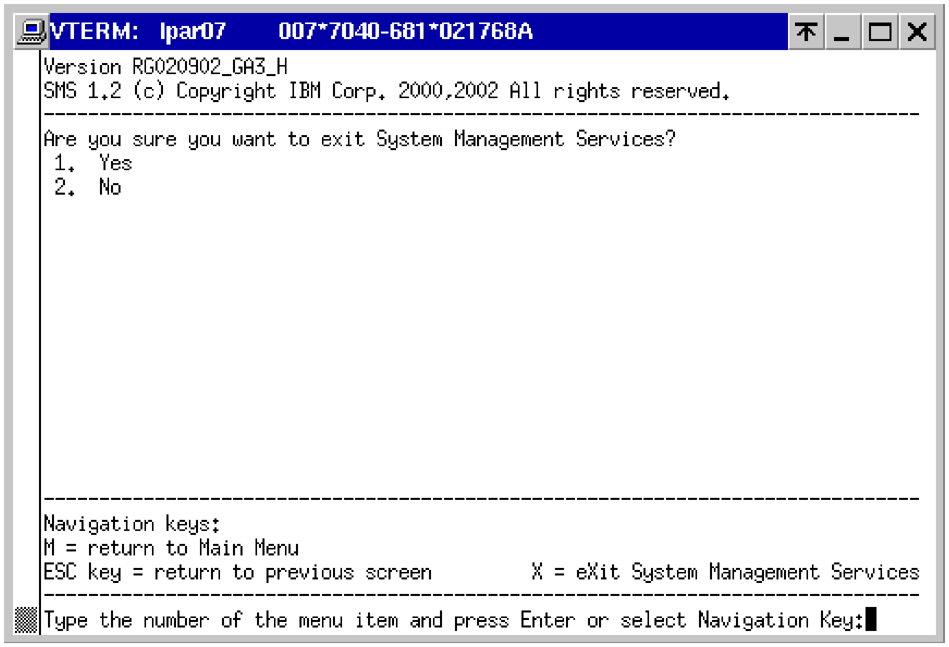
16. Type X to exit the SMS menu, as shown in Figure 7-9.


VTERM: lpar07 007*7040-681*021768A

```
Version RG020902_GA3_H
SMS 1.2 (c) Copyright IBM Corp. 2000,2002 All rights reserved.
-------------------------------------------------------------------------------
Select Task

SCSI CD-ROM ( loc=U1.9-P1-I10/Z1-A5,0 )

 1.   Information
 2.   Set Boot Sequence: Configure as 1st Boot Device








-------------------------------------------------------------------------------
Navigation keys:
M = return to Main Menu
ESC key = return to previous screen         X = eXit System Management Services
-------------------------------------------------------------------------------
Type the number of the menu item and press Enter or select Navigation Key:
```

Figure 7-9 System Management Services: Select task

17. Press 2 to confirm the exit from the SMS menu, as shown in Figure 7-10. After you exit from the SMS menu, the partition boots off from the CD/DVD device, and the standard AIX installation process starts.



```
Version RG020902_GA3_H
SMS 1.2 (c) Copyright IBM Corp. 2000,2002 All rights reserved.
-------------------------------------------------------------------------------
Are you sure you want to exit System Management Services?
 1.  Yes
 2.  No










-------------------------------------------------------------------------------
Navigation keys:
M = return to Main Menu
ESC key = return to previous screen         X = eXit System Management Services
-------------------------------------------------------------------------------
Type the number of the menu item and press Enter or select Navigation Key:
```

Figure 7-10 SMS: Exit confirmation

18. After the installation finishes, the partition automatically reboots. Log in to the partition to verify that the system is working properly.

19. Now, you should change the boot mode to Normal for the partition in the partition profile (see Figure 7-1 on page 208). Otherwise, it will show the SMS panel for every reboot.

20. Log in to the partition as the root user and perform initial customization, such as setting the date and time, host name, IP address, subnet mask, and default route.

21. Reboot the partition once in order for the RMC configuration change to take effect. Wait at least five minutes before attempting any DLPAR operations.

22. If the installed partition supports the dynamic logical partitioning function, you can remove the removable media device from this partition now. For further information about how to remove the device, see 8.1.3, “Dynamically removing resources from a partition” on page 269 and 9.3, “Dynamic logical partitioning operations using chhwres” on page 285.

7.3 Installing AIX using Network Installation Manager (NIM)

This section introduces the concept of the AIX Network Installation Manager (NIM) and its usage to install AIX in a partitioned environment.

7.3.1 NIM overview

NIM is a facility that enables you to install AIX over the network. NIM provides the ability to install AIX on a pSeries sever, called the *NIM client*, from a server, called the *NIM master*, over the network. NIM is able to not only install and maintain the AIX operating system, but also any additional software and fixes. NIM also enables you to customize the configuration of machines both during and after installation, such as a host name and TCP/IP addresses. NIM eliminates the need for access to physical media, such as tapes and CD-ROMs, because the installation images are stored on the NIM server and provided to NIM clients through the Network File System (NFS).

You can also create system backups using the **mksysb** command, with the appropriate customization for installation purposes. The mksysb image can be stored on any server in the NIM environment, including the NIM master. You can use NIM to restore a system backup to the same partition or to another partition. If you use a mksysb image to install AIX to some partitions, you can duplicate a customized[3] operating system image.

7.3.2 Configure NIM resources

Before installing AIX using NIM in a logical partitioned environment, you have to set up a NIM master either on another pSeries server or in one of the partitions. In this example, we use the lpar04 partition as the NIM master, which is installed with AIX 5L Version 5.2.

NIM master requirements

The NIM master must meet the following requirements:

- The NIM master must always be at the highest level of the AIX release that you are going to install. Therefore, if you are going to install AIX 5L Version 5.2, the NIM master also must be AIX 5L Version 5.2.
- You need a network connection between the NIM master and clients. This network can be either Ethernet or token ring.
- A minimum of 8 MB free space in the /tmp directory.

[3] To customize the system upon NIM installation, you can create and customize a nim_script resource on a NIM server.

- We recommend you create a separate volume group to be used for NIM operations on the NIM master. The initial free space needed for this setup is about 1.2 GB (usually one release of AIX requires approximately 4.5 GB).

The NIM master and NIM client host names must be consistently resolvable regardless on the NIM master or clients.

Configure a NIM master quickly

To configure a NIM master, you can use the **nim_master_setup** command, which is provided by AIX starting with the 5100-02 Recommended Maintenance Level. It is actually a script that greatly eases the task of setting up a NIM master. The following tasks, which used to be done separately, are all now automated in this command:

- Check and automatically install all necessary filesets required for the NIM master, for example, bos.sysmgt.sysbr and bos.sysmgt.nim.master.
- Define the NIM master.
- Create a generic mksysb resource.
- Create a resolv_conf resource.
- Create a bosinst_data resource.
- Copy filesets from the specified device and create a lpp_source resource.
- Create a Shared Product Object Tree (SPOT) resource.

The following example demonstrates how to invoke the **nim_master_setup** command. In this example, we have defined a volume group, nimvg, for storage of the NIM resources.

```
root@lpar04: # nim_master_setup -a volume_group=nimvg
############################## NIM master setup ##############################
#                                                                            #
#  During script execution, lpp_source and spot resource creation times      #
#  may vary. To view the install log at any time during nim_master_setup,    #
#  run the command: tail -f /var/adm/ras/nim.setup in a separate screen.     #
#                                                                            #
##############################################################################
Creating image.data file...done
Device location is /dev/cd0.
Resources will be defined on volume group nimvg.
Resources will exist in filesystem /export/nim.
Checking for backup software...already installed.
Checking /tmp space requirement...done
Installing NIM master fileset....
+-----------------------------------------------------------------------------+
                    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.sysmgt.nim.master 5.2.0.0               # Network Install Manager - Ma...
<< End of Success Section >>

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.sysmgt.nim.master 5.2.0.0
. . . . . << Copyright notice for bos.sysmgt >> . . . . . . .
 Licensed Materials - Property of IBM
5765E6200
   (C) Copyright International Business Machines Corp. 1993, 2002.
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.sysmgt >>. . . .

Finished processing all filesets.  (Total time:  15 secs).
+-----------------------------------------------------------------------------+
                                Summaries:
+-----------------------------------------------------------------------------+
Installation Summary
--------------------
Name                        Level           Part        Event       Result
-------------------------------------------------------------------------------
bos.sysmgt.nim.master       5.2.0.0         USR         APPLY       SUCCESS
Defining NIM master...0513-071 The nimesis Subsystem has been added.
0513-071 The nimd Subsystem has been added.
0513-059 The nimesis Subsystem has been started. Subsystem PID is 327768.
Located volume group nimvg.
Creating /export/nim filesystem...done
Creating /tftpboot filesystem...done
```

```
Checking /export/nim space requirement...done

Creating list of files to back up.
Backing up 27406 files.....................
27406 of 27406 files (100%)
0512-038 mksysb: Backup Completed Successfully.
Creating mksysb resource master_sysb...done
Creating resolv_conf resource master_net_conf...done
Creating bosinst_data resource (tty) bid_tty_ow...done
Creating bosinst_data resource (lft) bid_lft_ow...done
Creating 520lpp_source resource lpp_res...done
Creating 520spot resource spot_res...done
Creating resource group basic_res_grp...done
The following resources now exist:
boot                resources       boot
nim_script          resources       nim_script
master_sysb         resources       mksysb
master_net_conf     resources       resolv_conf
bid_tty_ow          resources       bosinst_data
bid_lft_ow          resources       bosinst_data
520lpp_res          resources       lpp_source
520spot_res         resources       spot
NIM master setup is complete - enjoy!
```

The log file, /var/adm/ras/nim.setup, provides more detailed information that can be used for debugging purposes. Because this file is replaced every time the command runs, make sure you save a copy before rerunning the **nim_master_setup** command next time.

The **nim_master_setup** command can be run several times with no ill-effect, because if the file system or resource already exists and is in proper condition, the command will not recreate it.

Note: We recommend you verify the state of NIM resources before continuing with the next step. Make sure the resource state is Ready for NIM operation by issuing **lsnim -l resource_name**.

To configure a NIM master, do the following:

1. Verify that the NIM client host name can be resolved on the NIM master and vice versa.
2. Define the NIM clients on the NIM master by selecting the following SMIT panels[4]:

   ```
   # smitty
       Software Installation and Maintenance
           EZ NIM (Easy NIM Tool)
               Configure as a NIM Master
                   Add client to the NIM environment
   ```

   Specify the NIM client host name in the Host Name of Machine field, and then press Enter (in this example, we specified lpar06). The Define a Machine SMIT panel opens, as shown in Example 7-1. Verify values in the panel, and then press Enter.

Example 7-1 Define a Machine screen

```
                          Define a Machine

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

[TOP]                                                    [Entry Fie]
* NIM Machine Name                                    [lpar06]
* Machine Type                                        [standalone]
* Hardware Platform Type                              [chrp]
  Kernel to use for Network Boot                      [mp]
  Primary Network Install Interface
*   Cable Type                                        tp
    Network Speed Setting                             []
    Network Duplex Setting                            []
*   NIM Network                                        master_net
*   Host Name                                          lpar06
    Network Adapter hardware address                   0
    Network Adapter Logical Device Name               []
  IPL ROM Emulation Device                            []
  CPU Id                                              []
  Machine Group                                       []
  Comments                                            []
```

[4] You can access the EZ NIM SMIT panel using the shortcut **smitty eznim**, which is available on AIX 5L Version 5.1 with 5100-03 Recommended Maintenance Level and AIX 5L Version 5.2 or later.

Verify that lpar06 has been defined correctly to the NIM master, as shown in the following example:

```
root@lpar04:/  # lsnim -l lpar06
lpar06:
   class          = machines
   type           = standalone
   platform       = chrp
   netboot_kernel = mp
   if1            = master_net lpar06 0
   cable_type1    = tp
   Cstate         = ready for a NIM operation
   prev_state     = ready for a NIM operation
   Mstate         = not running
```

3. Allocate resources for installation to the NIM clients:

```
root@dpar04:/ # nim_clients_setup
NSORDER=local,bind

Generating list of client objects in NIM environment...
Locating lpar06...done

Checking for resource group basic_res_grp...done
```

This command allocates the necessary resources that are used in the NIM installation, as shown in the following example:

```
root@lpar04:/ # lsnim -l lpar06
lpar06:
   class          = machines
   type           = standalone
   default_res    = basic_res_grp
   platform       = chrp
   netboot_kernel = mp
   if1            = master_net lpar06 0
   cable_type1    = tp
   Cstate         = BOS installation has been enabled
   prev_state     = ready for a NIM operation
   Mstate         = not running
   boot           = boot
   bosinst_data   = bid_ow
   lpp_source     = 520lpp_res
   mksysb         = master_sysb
   nim_script     = nim_script
   resolv_conf    = master_net_conf
   spot           = 520spot_res
   control        = master
```

Note: You can use the **`-m`** flag of the **`nim_client_setup`** command to specify the mksysb file that you want to install. If not specified, the default mksysb file is specified by the master_sysb object created by the **`nim_master_setup`** command.

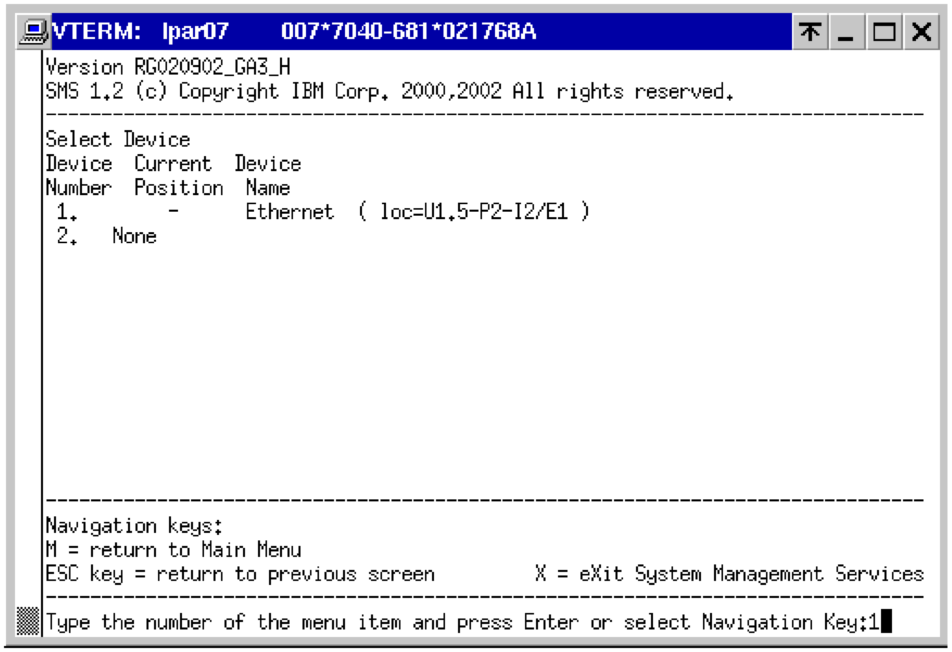
7.3.3 Boot partitions over the network

Select the partition profile that boots to SMS mode and activate the NIM client partition to be installed.

When SMS main menu opens, complete the following steps:

1. Press 7 to select **7. Select Boot Options**, as shown in Figure 7-2 on page 209.
2. Press 2 to select **2. Select Boot Devices**, as shown in Figure 7-3 on page 210.
3. Press 1 to select **1. Select 1st Boot Device**, as shown in Figure 7-4 on page 211).
4. Press 6 to select **6. Network**, as shown in Figure 7-5 on page 212.

5. Press the appropriate key to select the network adapter that you are going to use as the boot network adapter. In this example, we pressed 1 to select 1. Ethernet (loc=U1.5-P2/I2/E1), as shown in Figure 7-11.



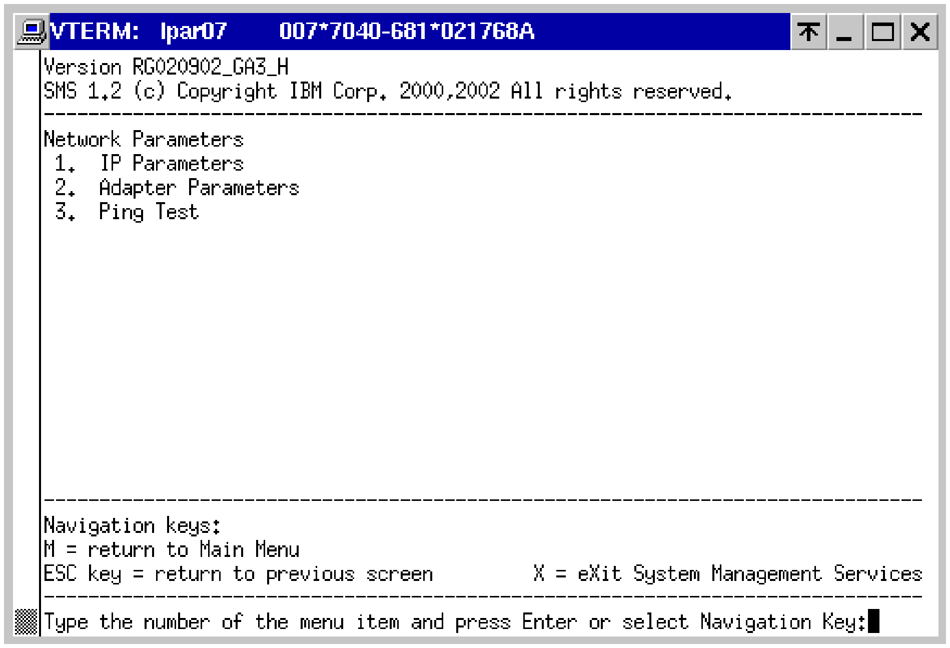
```
Version RG020902_GA3_H
SMS 1.2 (c) Copyright IBM Corp. 2000,2002 All rights reserved.
-------------------------------------------------------------------------------
Select Device
Device  Current  Device
Number  Position  Name
 1.        -      Ethernet  ( loc=U1.5-P2-I2/E1 )
 2.    None







-------------------------------------------------------------------------------
Navigation keys:
M = return to Main Menu
ESC key = return to previous screen         X = eXit System Management Services
-------------------------------------------------------------------------------
Type the number of the menu item and press Enter or select Navigation Key:1
```

Figure 7-11 System Management Services: Select Device screen (Network)

Note: In this example, the lpar07 partition is assigned only one Ethernet adapter. However, it is most likely that a partition is assigned several network adapters. Make sure to select the correct network adapter that is connected to the NIM master server.

6. Press 2. Set Boot Sequence: Configure as 1st Boot Device.
7. Press M to return to SMS main menu.
8. Press 4 to select **4. Setup Remote IPL (Initial Program Load)**.
9. Select an appropriate adapter to be used for the network installation for the partition.

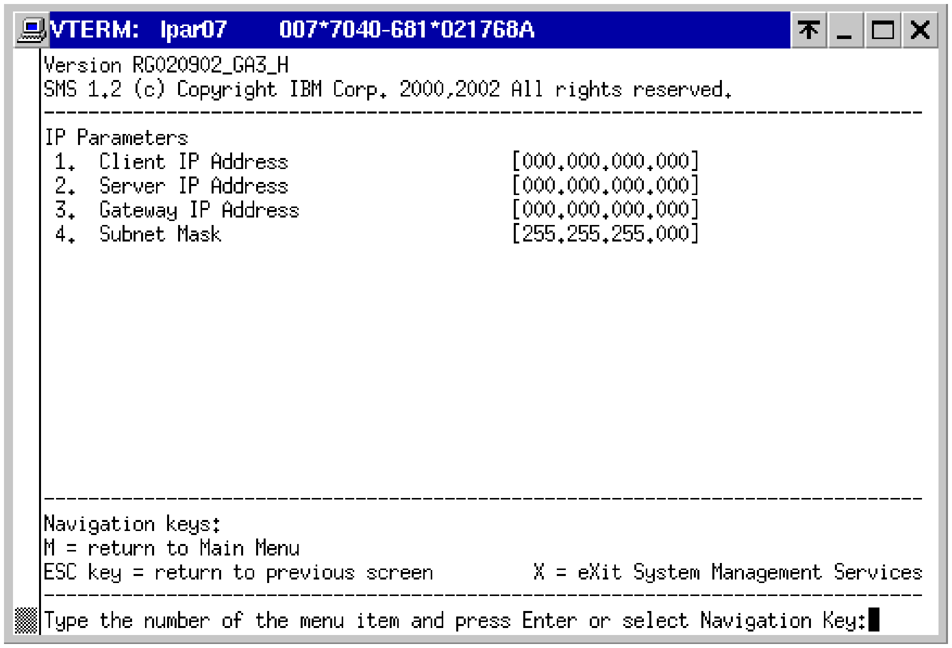
10.The Network Parameters screen opens, as shown in Figure 7-12.


VTERM: lpar07 007*7040-681*021768A

```
Version RG020902_GA3_H
SMS 1.2 (c) Copyright IBM Corp. 2000,2002 All rights reserved.
-------------------------------------------------------------------------------
Network Parameters
 1.   IP Parameters
 2.   Adapter Parameters
 3.   Ping Test









-------------------------------------------------------------------------------
Navigation keys:
M = return to Main Menu
ESC key = return to previous screen         X = eXit System Management Services
-------------------------------------------------------------------------------
Type the number of the menu item and press Enter or select Navigation Key:
```

Figure 7-12 System Management Services: Network Parameters screen

11. Press 1 to select **1. IP Parameters**. Specify the appropriate IP address of the partition to be installed, NIM master, default gateway, and subnet mask in the screen, as shown in Figure 7-13.

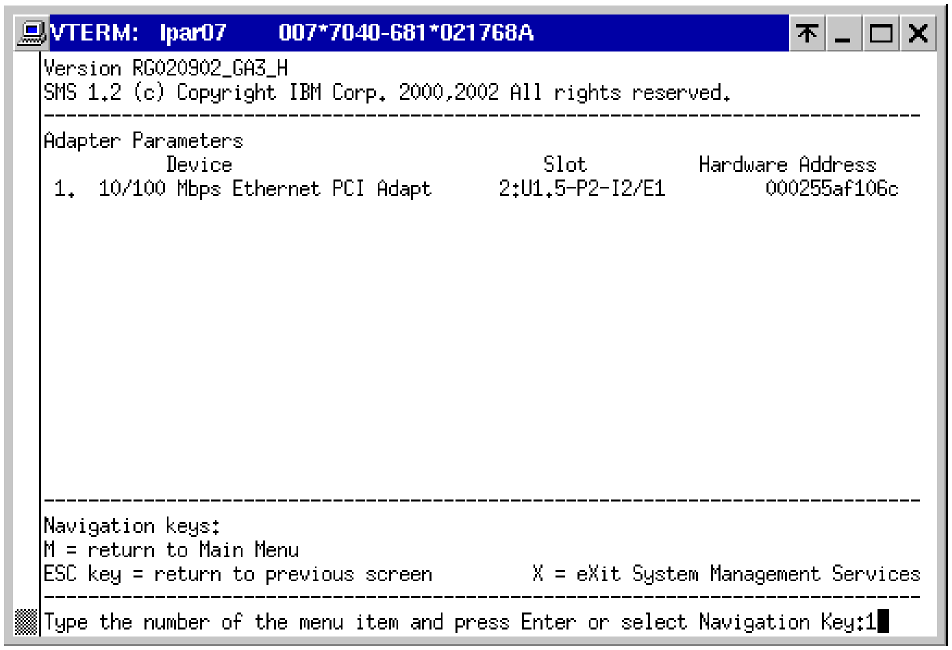


```
VTERM:  lpar07     007*7040-681*021768A
Version RG020902_GA3_H
SMS 1.2 (c) Copyright IBM Corp. 2000,2002 All rights reserved.
-------------------------------------------------------------------------------
IP Parameters
 1.   Client IP Address                     [000.000.000.000]
 2.   Server IP Address                     [000.000.000.000]
 3.   Gateway IP Address                    [000.000.000.000]
 4.   Subnet Mask                           [255.255.255.000]








-------------------------------------------------------------------------------
Navigation keys:
M = return to Main Menu
ESC key = return to previous screen         X = eXit System Management Services
-------------------------------------------------------------------------------
Type the number of the menu item and press Enter or select Navigation Key:
```

Figure 7-13 System Management Services: IP Parameters screen

Press the Esc key.

12. Select **2. Adapter Parameters** as shown in Figure 7-12.

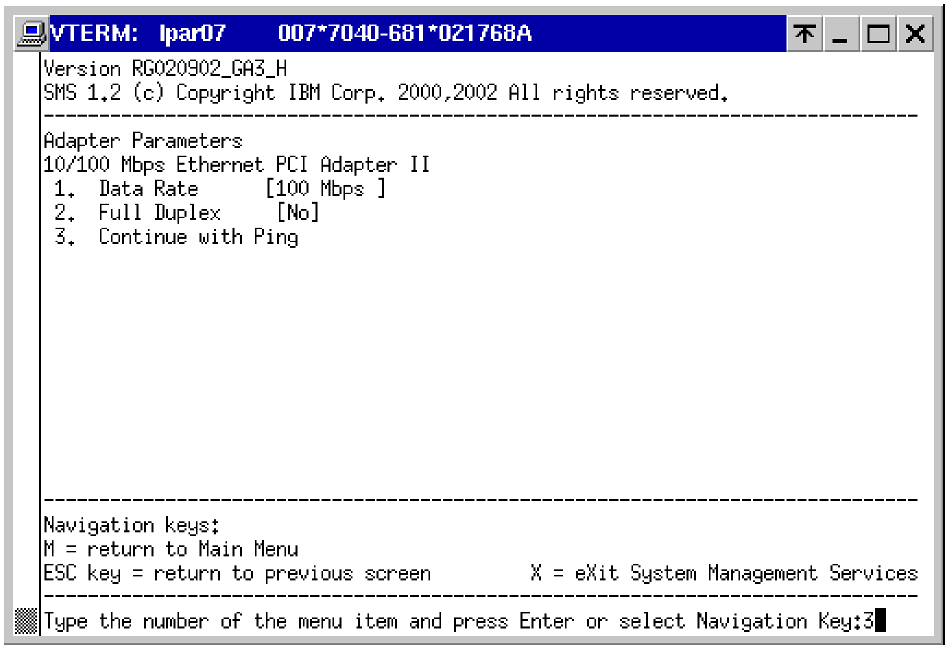


```
Version RG020902_GA3_H
SMS 1.2 (c) Copyright IBM Corp. 2000,2002 All rights reserved.
-------------------------------------------------------------------------------
Adapter Parameters
           Device                                  Slot          Hardware Address
 1.  10/100 Mbps Ethernet PCI Adapt      2:U1.5-P2-I2/E1         000255af106c









-------------------------------------------------------------------------------
Navigation keys:
M = return to Main Menu
ESC key = return to previous screen         X = eXit System Management Services
-------------------------------------------------------------------------------
Type the number of the menu item and press Enter or select Navigation Key:1
```

Figure 7-14 System Management Services: Adapter Parameters screen

Depending on the network media type between the partition and the NIM master server, specify the appropriate parameters for this adapter. Then, press the Esc key to go back to the Network Parameter screen, as shown in Figure 7-12 on page 226.

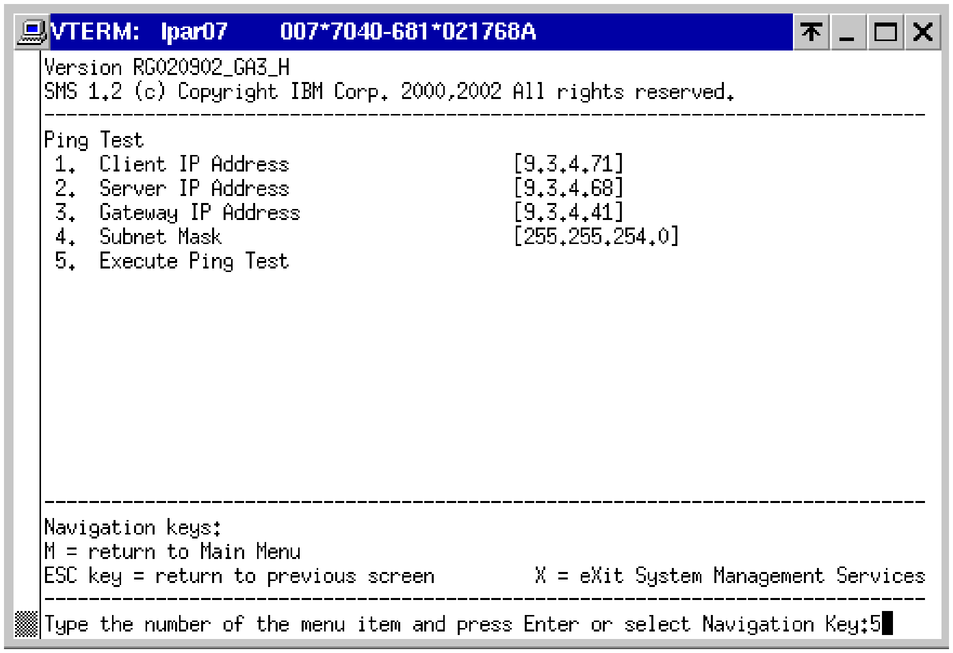
13.Press 3 to select **3. Ping Test**. Press 1. Execute Ping Test



```
Version RG020902_GA3_H
SMS 1.2 (c) Copyright IBM Corp. 2000,2002 All rights reserved.
-------------------------------------------------------------------------------
Adapter Parameters
10/100 Mbps Ethernet PCI Adapter II
 1.  Data Rate      [100 Mbps ]
 2.  Full Duplex     [No]
 3.  Continue with Ping









-------------------------------------------------------------------------------
Navigation keys:
M = return to Main Menu
ESC key = return to previous screen         X = eXit System Management Services
-------------------------------------------------------------------------------
Type the number of the menu item and press Enter or select Navigation Key:3
```

Figure 7-15 System Management Services: Adapter Parameters with ping test

14. After verifying the IP addresses, press 5 to select **5. Execute Ping Test**.



```
Version RG020902_GA3_H
SMS 1.2 (c) Copyright IBM Corp. 2000,2002 All rights reserved.
-------------------------------------------------------------------------------
Ping Test
 1.  Client IP Address                        [9.3.4.71]
 2.  Server IP Address                        [9.3.4.68]
 3.  Gateway IP Address                       [9.3.4.41]
 4.  Subnet Mask                              [255.255.254.0]
 5.  Execute Ping Test










-------------------------------------------------------------------------------
Navigation keys:
M = return to Main Menu
ESC key = return to previous screen        X = eXit System Management Services
-------------------------------------------------------------------------------
Type the number of the menu item and press Enter or select Navigation Key:5
```

Figure 7-16 System Management Services: Ping Test screen

If the ping test succeeds, you will see the message `Ping Success`. If you see this message, proceed to the next step. Otherwise, verify the following information:

– Whether or not you are using the correct network adapter. If you have several adapters with the same network media type, you might have selected the wrong one.

– The physical network cable connection status between the adapter and the network media, such as an Ethernet switch.

– Whether you have specified correct IP addresses in Figure 7-13 on page 227.

Note: If you use a 10/100 Mbps Ethernet adapter for NIM installation purposes, we do not recommend leaving the adapter parameters at *auto detect*. You can confirm the actual Ethernet adapter configuration using the **`entstat`** command (for example, **`entstat -d ent0`**) on the NIM server. You should set the same parameter on the NIM client and NIM server. If you still fail ping to the NIM server from the NIM client, you should use the setting 10 Mbps and half-duplex on NIM client, even if the NIM server is 100 Mbps and full-duplex.

15. Type X to exit the SMS menu.
16. Type 2 to confirm to exit from the SMS menu, as shown in Figure 7-10 on page 217. After exiting from the SMS menu, the partition boots off from the network as you defined. The NIM master then responds to the boot request from the client, and the installation process begins.
17. After the installation finishes, the client reboots. Log in to the client to verify that the system is working properly.

7.4 Installing AIX using alternate disk install

This section introduces the concept of alternate disk install and its usage to install AIX in a partitioned environment.

7.4.1 Alternate disk install overview

The AIX alternate disk installation is a system management function used in the environment that requires severe control of system availability and software level tracking. By using this function, the system administrator, who has to manage the system under limited system maintenance time, can reduce the software maintenance time using an additional disk drive for system maintenance.

In alternate disk installation, you have basically two ways to select the installation image source (see Figure 7-17 on page 232). The first way is called *cloning*. This clones the rootvg that is currently running. The second way is called *mksysb image install*, where you specify the mksysb image file location and instal it to the target disk. In both cases, the new volume group name will be altinst_rootvg. When a system or a partition is booted from that disk, its name is modified to rootvg. If you want to create more alternate install disks, you can use the **`-v`** option of the **`alt_disk_install`** command to avoid conflicting volume group names.

In both cases, the operating system that is currently running does not have to be shut down while installing the image. You can also customize the target volume group before you reboot from the target disk. After the alternate disk installation is finished, you can switch the boot disk. As a result, you can reduce the system maintenance time for system customization and software level updates.

For further information about alternate disk install, please refer to *AIX 5L Version 5.2 Reference Documentation: Commands Reference*, available at:

http://techsupport.services.ibm.com/server/library

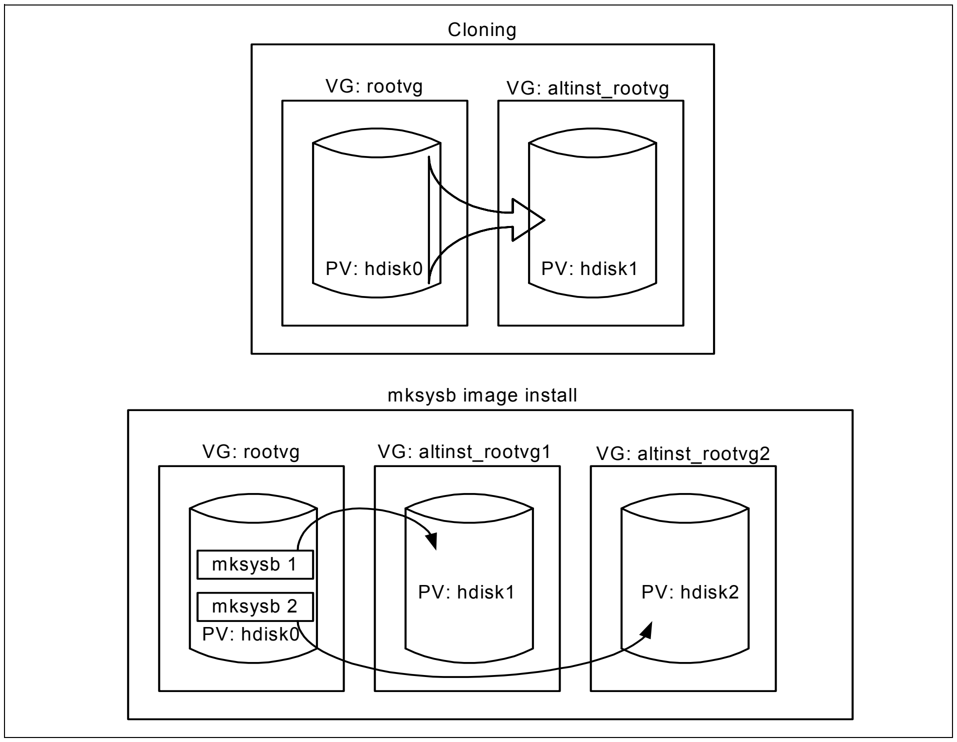


Figure 7-17 The alternate disk install concept

You can select multiple[5] mksysb images to install and customize the target disk before the operation finishes. For example, you can change the host name and IP address using the customization script.

If the disk drives used in this method are assigned to SCSI controllers other than the ones for the rootvg, you can reassign that SCSI controllers to the target partition. However, in this case, you have to shut down the source partition if the partition does not support dynamic logical partitioning.

5 Only one **`alt_disk_install`** command can be invoked at a time.

7.4.2 Install AIX 5L Version 5.1 using alternate disk install

The following example shows how to install AIX 5L Version 5.1 to a partition using alternate disk install (cloning method) in detail. We assume the following points in this example:

- You have a partition (*source partition*) AIX 5L Version 5.1 installed.
- The source partition has two disks. The rootvg contains only one disk drive (hdisk0), and it is not mirrored yet. Run the **lspv** command to check these disks:

```
# lspv
hdisk0          000030029158b447                    rootvg
hdisk1          00003002490a45b7                    None
```

  The SCSI controller, which is connected with hdisk0, is assigned to the source partition as Required, and the other SCSI controller, which is connected with hdisk1, is assigned as Desired, as shown in Figure 7-18.

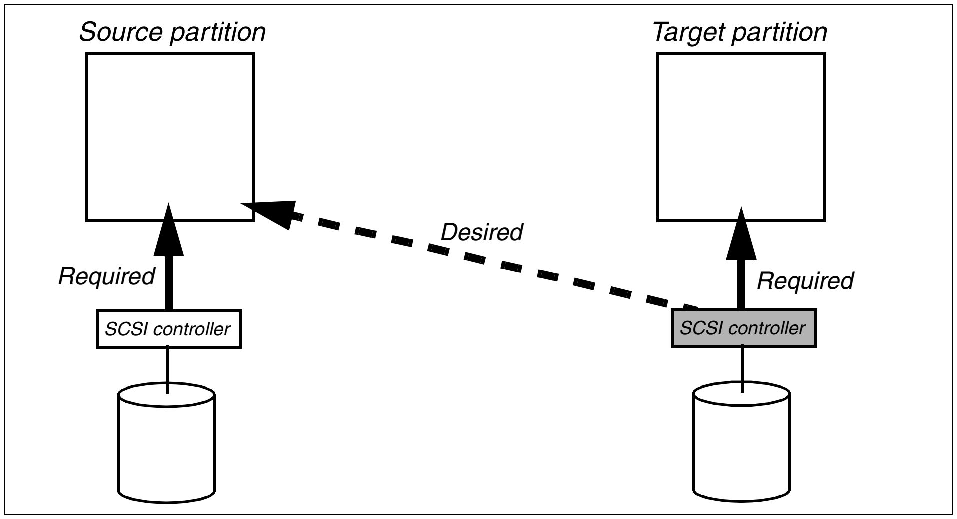


Figure 7-18 Installing AIX using alternate disk install

- The target partition has not been activated (started). The SCSI controller, which is assigned to the source partition as Desired, is assigned to the target partition as Required, as shown in Figure 7-18.

- The following filesets are installed on the source partition:
  - bos.alt_disk_install.boot_images[6]
  - bos.alt_disk_install.rte
  - bos.msg.en_US.alt_disk_install.rte

If these points are all met, do the following:

1. Run the **alt_disk_install** command on the source partition. The **-C** option causes the current rootvg to be cloned to the target disk hdisk1.

   This command creates the alternate rootvg volume group on hdisk1 as a cloned image from the rootvg on hdisk0. The produced alternate rootvg has the name altinst_rootvg. The file system and logical volume names of the altinst_rootvg have the prefix alt_.

   Note: The **-O** option wipes out the device definition from the ODM in altinst_rootvg. Using this option, the phantom devices will not appear on the target partition. The phantom devices are those that currently do not exist in the system and appear as Defined in the **lsdev -C** command output.

   Here is an example output of the **alt_disk_install** command:

```
# alt_disk_install -C -O hdisk1
Calling mkszfile to create new /image.data file.
Checking disk sizes.
Creating cloned rootvg volume group and associated logical volumes.
Creating logical volume alt_hd5.
Creating logical volume alt_hd6.
Creating logical volume alt_hd8.
Creating logical volume alt_hd4.
Creating logical volume alt_hd2.
Creating logical volume alt_hd9var.
Creating logical volume alt_hd3.
Creating logical volume alt_hd10opt.
Creating /alt_inst/ file system.
Creating /alt_inst/opt file system.
Creating /alt_inst/tmp file system.
Creating /alt_inst/usr file system.
Creating /alt_inst/var file system.
Generating a list of files
for backup and restore into the alternate file system...
Backing-up the rootvg files and restoring them to the alternate file
system...
```

[6] The fileset, bos.alt_disk_install.boot_images, is not required for cloning, but required for the mksysb install.

```
Modifying ODM on cloned disk.
Building boot image on cloned disk.
Resetting all device attributes.
NOTE: The first boot from altinst_rootvg will prompt to define the new
system console.
forced unmount of /alt_inst/var
forced unmount of /alt_inst/usr
forced unmount of /alt_inst/tmp
forced unmount of /alt_inst/opt
forced unmount of /alt_inst
forced unmount of /alt_inst
Changing logical volume names in volume group descriptor area.
Fixing LV control blocks...
Fixing file system superblocks...
Bootlist is set to the boot disk: hdisk1
```

After this command completes, the volume group altinst_rootvg is automatically created as a clone of rootvg:

```
# lpsv
hdisk0          000030029158b447                    rootvg
hdisk1          00003002490a45b7                    altinst_rootvg
```

At this point, the volume group altinst_rootvg is varied off. If you want to customize them, for example, to modify /etc/resolv.conf, then you can *wake up* altinst_rootvg using the **`-W`** option of the **`alt_disk_install`** command:

```
# alt_disk_install -W hdisk1
```

The volume group altinst_rootvg is varied on, and all the file systems contained in this volume group are mounted under the /alt_inst directory. You can access /etc/resolv.conf in this volume group using the full path name /alt_inst/etc/resolv.conf.

By issuing the **`alt_disk_install -S`** command, the volume group altinst_rootvg is suspended; therefore, all the file systems under /alt_inst (including itself) are unmounted, and the volume group is varied off:

```
# alt_disk_install -S
```

2. Remove the information created by the **`alt_disk_install`** command from the ODM on the source partition. This command also puts the boot device back to hdisk0:

```
# alt_disk_install -X
Bootlist is set to the boot disk: hdisk0
```

Note: Never issue **`exportvg altinst_rootvg`**.

3. Delete the hdisk1 definition from the ODM on the source partition:

```
# rmdev -l hdisk1 -d
```

4. Shut down the source partition and deactivate it:

   ```
   # shutdown -F now
   ```

5. Activate (start) the target partition by setting the boot mode to SMS (see Figure 7-1 on page 208). Then, set the boot device to the newly assigned disk. After the boot process, the alternate rootvg becomes rootvg, and all file system and logical volume names automatically lose the alt_ prefix.

   Note: You have to define the system console upon first reboot, because you wiped out the device definition from the ODM using the **`-O`** option of the **`alt_disk_install`** command.

6. Configure the appropriate TCP/IP settings, such as new host name, IP address, and subnet mask, and then issue the following command to reconfigure RMC configuration on the target partition[7]:

   ```
   /usr/sbin/rsct/install/bin/recfgct
   ```

   If the source partition is installed with 5100-03 Recommended Maintenance Level, you can create a shell script and specify it with the **`-x`** option of the **`alt_disk_install`** command to automate this step, as explained in 7.4.3, "Install AIX 5L Version 5.2 using alternate disk install" on page 236.

7. Now, you can restart the source partition.

If you are not familiar with the **`alt_disk_install`** command options, please refer to *AIX 5L Version 5.2 Reference Documentation: Commands Reference*, available at:

`http://techsupport.services.ibm.com/server/library`

7.4.3 Install AIX 5L Version 5.2 using alternate disk install

The following example shows how to install AIX 5L Version 5.2 to a partition using alternate disk install (cloning method) in detail. In addition to the points assumed in the previous section, we assume the following in this example:

- ► All device filesets have been installed on the source partition.

  You can ensure this by selecting **YES** for the Enable System Backups to install any system option when you first install AIX 5L Version 5.2 on the source partition.

[7] If APAR IY35312 is applied on the source partition, this command is not necessary.

- To ensure a more automated operation, verify that the install_assist entry is not in /etc/inittab or it is commented out, as shown in the following example:

  ```
  # grep install_assist /etc/inittab
  :install_assist:2:wait:/usr/sbin/install_assist </dev/console >/dev/console
  2>&1
  ```

  If it is not commented out when the partition is activated, install_assist runs, and you need to do some manual interaction with it, such as set the terminal type and configure the date and time.

- The SCSI controller, which is highlighted in Figure 7-18 on page 233, is identified with the physical location code U1.9-P2/Z2.

If these points are all met, do the following:

1. Perform a DLPAR operation to dynamically add the SCSI controller, which is identified with the physical location code U1.9-P2/Z2, to the source partition by using either the graphical user interface or command line interface on the HMC. For further information about this DLPAR operation, see 8.1.1, “Dynamically adding resources to a partition” on page 256 or 9.3, “Dynamic logical partitioning operations using chhwres” on page 285.

   Note: It is possible to automate this step by issuing the **`chhwres`** command from the remote **`ssh`** session. However, we recommend you test this operation extensively.

2. On the source partition, run **`cfgmgr`** to configure the devices that have just been added. In the following example, the scsi1, hdisk2, hdisk3, and hdisk4 devices are newly configured after the **`cfgmgr`** invocation:

   ```
   # lsdev -Cc adapter
   ent0  Available 3F-08 10/100 Mbps Ethernet PCI Adapter II (1410ff01)
   ent1  Available 3J-08 10/100 Mbps Ethernet PCI Adapter II (1410ff01)
   scsi0 Available 3b-08 Wide/Ultra-3 SCSI I/O Controller
   sa0   Available       LPAR Virtual Serial Adapter
   root@lpar03:/ # lspv
   hdisk0          0021768a20eb2b9d                    None
   hdisk1          0021768aa445b99d                    rootvg          active
   # cfgmgr
   # lsdev -Cc adapter
   ent0  Available 3F-08 10/100 Mbps Ethernet PCI Adapter II (1410ff01)
   ent1  Available 3J-08 10/100 Mbps Ethernet PCI Adapter II (1410ff01)
   scsi0 Available 3b-08 Wide/Ultra-3 SCSI I/O Controller
   sa0   Available       LPAR Virtual Serial Adapter
   scsi1 Available 60-08 Wide/Ultra-3 SCSI I/O Controller
   # lspv
   hdisk0          0021768a20eb2b9d                    None
   hdisk1          0021768aa445b99d                    rootvg          active
   ```

```
hdisk2          0021768a217bcf86                    None
hdisk3          0021768a9c8a895e                    None
hdisk4          none                                None
```

3. Create a shell script in order to customize the host name, IP address, default gateway, and DNS information on the target partition. This script is specified with the **-x** option of the **alt_disk_install** command in the next step, and then it executes on the target partition after alternate disk install:

   ```
   # vi /var/customize.lpar05
   /usr/sbin/mktcpip -h'lpar05.itsc.austin.ibm.com' \
           -a'9.3.4.69' \
           -m'255.255.254.0' \
           -i'en0' \
           -g'9.3.4.41' \
           -n'9.3.4.2' \
           -d'itsc.austin.ibm.com'
   /usr/sbin/rsct/install/bin/recfgct[8]
   # chmod a+rx /var/customize.lpar05
   ```

   If you are not familiar with the **mktcpip** command syntax, please refer to *AIX 5L Version 5.2 Reference Documentation: Commands Reference*, available at:

   http://techsupport.services.ibm.com/server/library

   Note: The **uname -L** command returns the partition ID and node name, as shown in the following example:

   ```
   # uname -L
   3 lpar03
   ```

   You may find it is useful to write your own customization script.

4. Execute the **alt_disk_install** command to clone rootvg to hdisk4 on the source partition. We summarize the options we have specified in this example in Table 7-1 on page 239.

[8] If APAR IY35456 is applied on the source partition, this line is not necessary.

Table 7-1 Command line interface options of the alt_disk_install command

Specified command line options	Explanation
`-C`	Specifies the cloning of rootvg
`-B`	Specifies not to set the bootlist to the target disk
`-O`	Specifies not to retain the device configurations
`-x /var/customize.lpar05`	Specifies the customization script (/var/customize.lpar05) that runs once when the target partition boots up
`-c /dev/tty0`	Specifies the console (/dev/tty0)
`-g`	Specifies not to check whether or not the target disk (hdisk4) is bootable

Note: The following new options are added to the **`alt_disk_install`** command to facilitate the task of cloning rootvg in a partitioned environment:

- The **`-x`** option specifies the customization script to be run once when the node boots up. This can be used to set up each node's unique attributes, such as host name and IP address. The specified script is called from /etc/firstboot upon the first boot. Therefore, it runs only once.
- The **`-c`** option specifies the console for the system. This helps eliminate the manual task of having to reply 1 to the prompt to define the console on the virtual terminal when the partition is activated the first time.
- The **`alt_disk_install`** command always checks first whether or not the disk is bootable by using the **`bootinfo -B`** command. However, the **`bootinfo -B`** command returns that the disk is not bootable if it was added after the partition boot, for example, the disk configured by the **`cfgmgr`** command after the DLPAR operation. The **`-g`** option instructs the **`alt_disk_install`** command to bypass this check. You should only specify this when the target disk has been verified as bootable.

If you do not specify the **`-g`** flag, you will get the following warning message, and the command will abort:

```
# alt_disk_install -C -B -O -x /var/customize.lpar05 -c /dev/tty0 hdisk4
0505-212 alt_disk_install: System checks indicate that hdisk4 may not be
bootable. This may be because the disk's parent adapter does not support
boot, or it does support boot but was dynamically added (in which case the
disk should still be bootable). If you believe this disk is bootable,
re-execute this command with the "-g" (ignore boot check) flag.
```

The following example shows an example output of the **`alt_disk_install`** command with the **`-g`** option:

```
# alt_disk_install -C -B -O -x /var/customize.lpar05 -c /dev/tty0 -g hdisk4
Calling mkszfile to create new /image.data file.
Checking disk sizes.
```

```
Creating cloned rootvg volume group and associated logical volumes.
Creating logical volume alt_hd5
Creating logical volume alt_hd6
Creating logical volume alt_hd8
Creating logical volume alt_hd4
Creating logical volume alt_hd2
Creating logical volume alt_hd9var
Creating logical volume alt_hd3
Creating logical volume alt_hd1
Creating logical volume alt_hd10opt
Creating logical volume alt_lg_dumplv
Creating /alt_inst/ file system.
Creating /alt_inst/home file system.
Creating /alt_inst/opt file system.
Creating /alt_inst/tmp file system.
Creating /alt_inst/usr file system.
Creating /alt_inst/var file system.
Generating a list of files
for backup and restore into the alternate file system...
Backing-up the rootvg files and restoring them to the alternate file
system...
Modifying ODM on cloned disk.
Building boot image on cloned disk.
Resetting all device attributes.
forced unmount of /alt_inst/var
forced unmount of /alt_inst/usr
forced unmount of /alt_inst/tmp
forced unmount of /alt_inst/opt
forced unmount of /alt_inst/home
forced unmount of /alt_inst
forced unmount of /alt_inst
Changing logical volume names in volume group descriptor area.
Fixing LV control blocks...
Fixing file system superblocks...
```

After the successful completion of the **alt_disk_install** command, hdisk4 belongs the altinst_rootvg volume group, which is copied from the rootvg, as shown in the following example:

```
# lspv
hdisk0          0021768a20eb2b9d                    None
hdisk1          0021768aa445b99d                    rootvg          active
hdisk2          0021768a217bcf86                    None
hdisk3          0021768a9c8a895e                    None
hdisk4          0021768a39eec2ff                    altinst_rootvg
```

Note: The **alt_disk_install** command generates the log file, /var/adm/ras/alt_disk_inst.log, that can be useful for debugging purposes.

5. Remove all definitions of altinst_rootvg from the ODM on the source partition by issuing the **alt_disk_install -X** command:

```
# alt_disk_install -X
Bootlist is set to the boot disk: hdisk1
```

   This cleans up unnecessary definitions in the ODM in order to ensure that there are no conflicts when executing another alternate disk install task later.

6. Remove the definition of the target disk and its parent devices before dynamically removing them from the source partition:

```
# lsdev -Cc disk
hdisk0 Available 3b-08-00-8,0  16 Bit LVD SCSI Disk Drive
hdisk1 Available 3b-08-00-10,0 16 Bit LVD SCSI Disk Drive
hdisk2 Available 60-08-00-8,0  16 Bit LVD SCSI Disk Drive
hdisk3 Available 60-08-00-9,0  16 Bit LVD SCSI Disk Drive
hdisk4 Available 60-08-00-10,0 16 Bit LVD SCSI Disk Drive
# lsdev -Cl hdisk4 -F parent
scsi1
# lsslot -c slot
# Slot      Description  Device(s)
U1.9-P2-I1  DLPAR slot   pci13 ent0
U1.9-P2-I2  DLPAR slot   pci14 ent1
U1.9-P2-I3  DLPAR slot   pci15
U1.9-P2-I4  DLPAR slot   pci16
U1.9-P2-I5  DLPAR slot   pci17
U1.9-P2/Z1  DLPAR slot   pci18 scsi0
U1.9-P2/Z2  DLPAR slot   pci19 scsi1
# rmdev -d -l pci19 -R[9]
hdisk2 deleted
hdisk3 deleted
hdisk4 deleted
ses1 deleted
scsi1 deleted
pci19 deleted
```

7. Perform a DLPAR operation to dynamically remove the SCSI controller, which is identified with the physical location code U1.9-P2/Z2, from the source partition by using either the graphical user interface or command line interface on the HMC. For further information about this DLPAR operation, see 8.1.1, “Dynamically adding resources to a partition” on page 256 or 9.3, “Dynamic logical partitioning operations using chhwres” on page 285.

   Note: It is possible to automate this step by issuing the **chhwres** command from the remote **ssh** session. However, we recommend you test this operation extensively.

[9] The **-R** option instructs the **rmdev** command to delete all children devices recursively.

8. Activate (start) the target partition by setting the boot mode to SMS (see Figure 7-1 on page 208). Then, set the boot device to the newly assigned disk. After the boot process, the alternate rootvg becomes rootvg, and all file system and logical volume names automatically lose the alt_ prefix.
9. Verify the following on the target partition:
   - The boot disk device
   - TCP/IP configuration (host name, IP address, and subnet mask)

Note: You should also confirm that you can perform DLPAR operations on the target partition.

7.5 Migrating AIX using alternate disk migration

This section introduces the concept of alternate disk migration and its usage to migrate AIX in a partition.

7.5.1 Alternate disk migration overview

In the past, when migrating AIX, some scheduled downtime was required, because of the need to shut down the machine and boot it from CD-ROM and then select the migration option to start the AIX migration. The downtime required depends on the number of filesets installed on the machine.

However, more and more machines are now being used in mission-critical applications, and thus, it is getting more and more difficult to find a long period of downtime.

The alternate disk migration is a facility that enables the system administrator to quickly migrate a live production system. This is achieved with the close co-operation between alternate disk install and NIM. Figure 7-19 on page 243 illustrates the concept of alternate disk migration, which is summarized in the following:

1. Use alternate disk install to clone rootvg of the system or partition to the target disk.
2. Export the /alt_inst/* file systems created in the alternate rootvg, and then mount them on the NIM master[10] through NFS.
3. Migrate the AIX release in the alternate rootvg using NIM resources on the NIM master.

[10] The NIM master can be configured on another partition or a stand-alone pSeries server.

4. After the migration successfully completes, unmount the NFS mounted file systems, and reboot the system or partition.

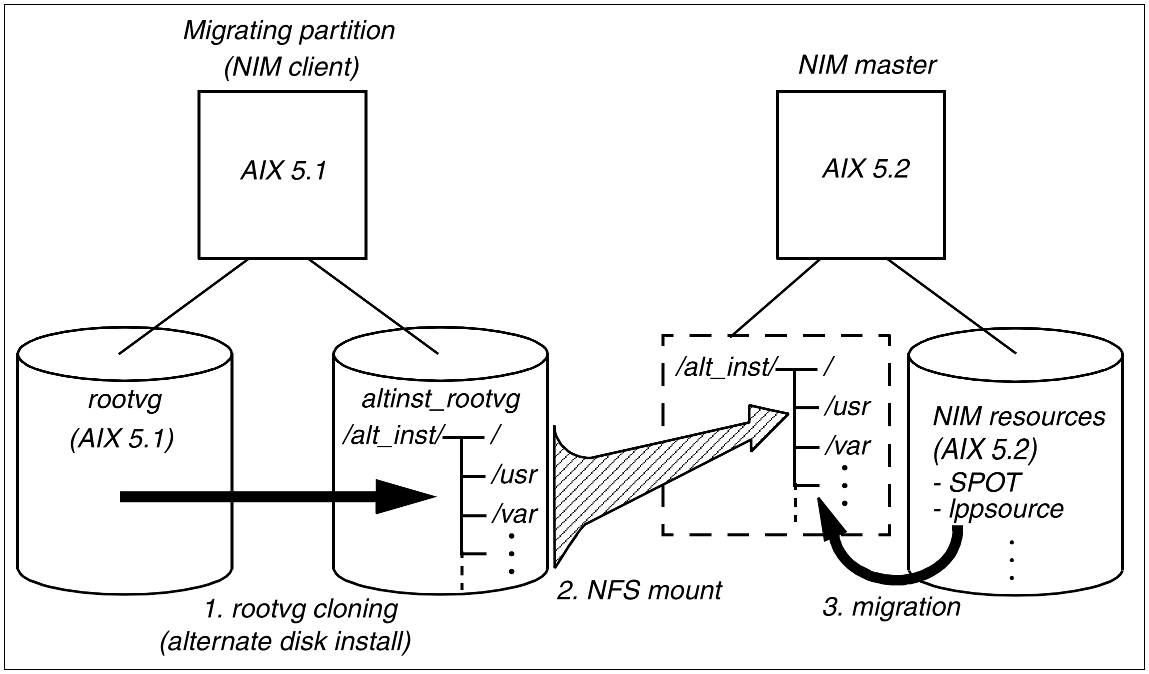


Figure 7-19 The alternate disk migration concept

The **nimadm** command, which stands for Network Installation Manager Alternate Disk Migration, is provided to perform alternate disk migration. This was introduced in AIX 5L Version 5.1 with 5100-03 Recommended Maintenance Level and AIX 5L Version 5.2.

For further information about alternate disk migration, please refer to *AIX 5L Version 5.2 Reference Documentation: Commands Reference*, available at:

http://techsupport.services.ibm.com/server/library

7.5.2 Requirements

The following requirements must be met in order to perform an alternate disk migration:

1. Configure a NIM master running AIX 5L Version 5.1 or later with at least 5100-03 Recommended Maintenance Level.
2. The NIM master must have the same level of bos.alt_disk_install.rte installed in the rootvg and SPOT.

3. The selected lpp_source and SPOT must match the AIX level that you are migrating to.
4. The NIM master should be at the same or later AIX level as the level being migrated to.
5. The client (the system to be migrated) must be at AIX Version 4.3.3 or later.
6. The client must have a disk (or disks) large enough to clone its rootvg and an additional 500 MB of free space for the migration.

   This free space is used by the **nimadm** command to automatically expand the size of any /alt_inst file system that needs to be increased so that it can satisfy the free space requirement needed by the migration.
7. The client must be registered with the master as a stand-alone NIM client.
8. The NIM master must be able to execute remote commands on the clients using the rshd protocol.
9. The NIM master and client must have a minimum of 128 MB memory.

   Because there are a lot of NFS read/write activities going on, if the machine does not have enough memory, NFS performance may be degraded to the point that a time-out begins.

   This requirement makes it less likely that an NFS time-out will happen.
10. A reliable network, which can facilitate large amounts of NFS traffic between the NIM master and the clients. The NIM master and the clients must be able to perform NFS mounts and read/write operations.

Note: Please be aware that the migration is not possible if the client hardware and software do not support the level of AIX being migrated to.

7.5.3 Limitations

The following are limitations of alternate disk migration:

1. All NIM resources used by alternate disk migration must be local to the NIM master.

   Using NIM resources that are not served by the master can introduce complications to the environment and is, therefore, not supported.
2. If the client uses trusted computing base (TCB), you will either need to disable it permanently or perform a conventional migration.

   This limitation exists because TCB needs to access file metadata that is not available over NFS.

Note:

- Although no changes are done to the client's rootvg, the client might experience some performance degradation during the migration due to the increased disk and network I/O and some CPU usage.
- NFS performance tuning may be required to increase **nimadm** performance.

7.5.4 Operation examples

The **nimadm** command can be used to perform a migration, cleanup, wakeup, and sleep.

Migration

The **nimadm** command performs migration in the following 12 phases. Each phase can be executed individually with the **-P** flag.

1. The NIM master issues a remote command to the client to perform rootvg cloning to the target disk.
2. The NIM master issues a remote command to the client to export all of the /alt_inst/* file systems to the master. They are exported in read/write mode with root access.

   Note: In phase 2, the **nimadm** command creates entries for the /alt_inst/* file systems in the /etc/exports file on the client. These entries allow only the root user from the NIM master a read/write access to the /alt_inst/* file systems. Therefore, for security reasons, phase 2 fails if there are already any entries in /etc/exports for the /alt_inst/* file systems.

3. The NIM master mounts the /alt_inst/* file systems.
4. The NIM master runs a premigration script if specified.
5. The NIM master prepares for the migration by saving the configuration files, calculating the free space needed, and expanding them. The bos fileset is restored, and the device configuration is merged.
6. The NIM master performs the migration of all system filesets. Any required RPM images are also installed.
7. The NIM master runs a post-migration script if specified.
8. The NIM master runs bosboot to create a client boot image.
9. The NIM master unmounts the /alt_inst/* file systems.

10. The NIM master issues a remote command to the client to unexport all of the /alt_inst/* file systems.
11. The NIM master issues a remote command to the client for some final adjustments and puts the alternate rootvg to sleep. The bootlist is set to the target disk unless the **-B** flag is specified.
12. The NIM master runs cleanup to end the migration. The client is rebooted if the **-r** flag is specified.

Note: The premigration and post-migration scripts have to be defined as NIM resources of type *script*. These scripts must be executable. By default, these scripts are executed on the NIM master in the client environment (using the **chroot** command).

Cleanup

This is used to clean up after a failed migration that somehow did not perform a cleanup itself. It can also be used to clear a previous migration in order to perform a new one.

When the **nimadm** command fails, it tries to clean up everything that it has done. Suppose that the migration fails at phase 2, exporting /alt_inst/* file systems from the client, the clean up process will undo things that have been done in all the previous phases. This means that if you want to run it again, you have to begin from phase 1, cloning the rootvg.

If you want the cleanup process not to undo everything it did, set the environment variable ALT_PARTIAL_CLEANUP to `yes` before issuing the **nimadm** command.

If you set this variable to yes in the previous example, the cleanup process will not undo phase 1. So, you can resume your operation from phase 2 by specifying the phases you would like to run with the **-P** flag, for example:

```
# nimadm -c lpar02 -l 520lpp_res -s 520spot_res -d hdisk2 -Y -r -P 2 3 4 5 6
```

Wakeup

This performs an alternate disk install wakeup. NFS exports of the /alt_inst/* file systems and mounts them on the NIM master.

The **alt_disk_install** command also provides a wakeup option. However, with alternate disk install, if your rootvg is AIX Version 4.3.3 and alternate rootvg is AIX 5L Version 5.1, you cannot use the wakeup option because you can only wakeup a lower or equal level of AIX. In this case, you need to reboot the system to AIX 5L Version 5.1, and then wakeup the AIX Version 4.3.3 volume group.

There is no such limitation in the wakeup option of the **nimadm** command due to the fact that the client uses the command in the SPOT provided from the NIM master that is always at the same level as the alternate rootvg.

Sleep

This performs a reverse of the alternate disk install wakeup by unmounting the /alt_inst/* file systems on the NIM master, unexporting /alt_inst/* file systems, and running alternate disk install sleep on the client.

7.5.5 Use alternate disk migration to migrate AIX

In this example, we migrate the lpar02 partition to AIX 5L Version 5.2 from AIX 5L Version 5.1 with 5100-02 Recommended Maintenance Level. The lpar04 partition is used as the NIM master server installed with AIX 5L Version 5.2.

To use alternate disk migration:

1. Set up the NIM master (see 7.3.2, “Configure NIM resources” on page 218).
2. Prepare the SPOT to support alternate disk migration by installing the fileset bos.alt_disk_install.rte to SPOT. Because the **nim_master_setup** command does not copy the bos.alt_disk_install package to the lppsource and install it to the SPOT, you need to explicitly do it.
   a. Find out the directory of the lppsource resource, 520lpp_res, created by the **nim_master_setup** command:

```
root@lpar04:/ # lsnim -l 520lpp_res
520lpp_res:
   class       = resources
   type        = lpp_source
   arch        = power
   Rstate      = ready for use
   prev_state  = unavailable for use
   location    = /export/nim/lpp_source/520lpp_res
   simages     = yes
   alloc_count = 1
   server      = master
```

   In this example, the 520lpp_res lppsource resource is located in the /export/nim/lpp_source/520lpp_res directory.

b. Copy the bos.alt_disk_install package to the lppsource 520lpp_res:

```
root@lpar04:/ # smitty
    Software Installation and Maintenance
        Software Maintenance and Utilities
            Copy Software to Hard Disk for Future Installation
```

- Select `/dev/cd0` for Input device.
- Select `bos.alt_disk_install` for Software package to copy.
- Specify `/export/nim/lpp_source/520lpp_res` for Directory for storing software package.

Press Enter.

c. Install the bos.alt_disk_install package to the SPOT 520spot_res:

```
root@lpar04:/ # smitty
    Software Installation and Maintenance
        Network Installation Management
            Perform NIM Software Installation and Maintenance Tasks
                Install and Update Software
                    Install Software
```

- Select `520spot_res` as target for the operation.
- Select `520lpp_res` as lppsource.
- Press F4 to list Software to Install and select bos.`alt_disk_install`.

Press Enter twice.

Note: To install software packages in the SPOT, it must not be allocated to any NIM client. Use the following command to deallocate SPOT from the clients:

```
nim -o deallocate -a spot=520spot_res <client_name>
```

After the installation, boot images (for example, 520spot_res.chrp.mp.ent) are recreated in the /tftpboot directory.

3. On the migration target partition (lpar02), define the partition as a NIM client:

```
root@lpar02:/ # smitty
    Software Installation and Maintenance
        Network Installation Management
            Configure Network Installation Management Client Fileset
```

The Configure Network Installation Management Client Fileset panel opens. Select and specify fields, as shown in Example 7-2 on page 249, and then press Enter. This panel defines lpar02 as a NIM client for the NIM master lpar04.

Example 7-2 Configure Network Installation Management Client Fileset panel

```
           Configure Network Installation Management Client Fileset

Type or select values in entry fields.
Press Enter AFTER making all desired changes.
                                                     [Entry Fields]
* Machine Name                                       [lpar02]
* Primary Network Install Interface                  [en0]
* Host Name of Network Install Master                [lpar04]

  Hardware Platform Type                             [chrp]
  Kernel to use for Network Boot                     [mp]
  IPL ROM Emulation Device                           []
  Comments                                           []

  Alternate Port Numbers for Network Communications
       (reserved values will be used if left blank)
    Client Registration                              []
    Client Communications                            []
```

It also creates the following on the NIM client:

- A NIM configuration file, /etc/niminfo
- The /.rhosts file, which allows the NIM master remote command execution on the client

If the client is installed with AIX 5L Version 5.1 with 5100-03 Recommended Maintenance Level or later, you can select the following SMIT panels in order to do the same task: **`smitty eznim`** → **Configure as a NIM Client** → **Add this system to a NIM environment**.

4. Verify that the following requirements are met:
   - The migration target disk is enough large to clone the rootvg and has an additional space of 500 MB.
   - The network between NIM master and NIM client is reliable and can facilitate a large amount of NFS traffic.
5. Verify that the limitations are not violated.

   See 7.5.2, “Requirements” on page 243 and 7.5.3, “Limitations” on page 244.
6. Execute the **`nimadm`** command on the NIM master. We summarize the options we have specified in this example in Table 7-2 on page 250.

Table 7-2 Command line interface options of the nimadm command

Specified command line options	**Explanation**
-c lpar02	Specifies the client to do the alternate disk migration. Only one client can be specified with one command at execution time, but you can have multiple **nimadm** commands running at the same time to migrate several clients.
-l 520_lppres	Specifies the lppsource resource to be used. We recommend copying all software packages in the AIX BOS installation CD-ROM media to lppsource in order to ensure that the NIM master can find all necessary filesets during the migration process. Otherwise, you need to manually migrate the filesets that were missing in the lppsource resource when the **nimadm** command was executed.
-s 520spot_res	Specifies the SPOT resource to be used. This is mandatory.
-d hdisk2	Specifies the target disk on the client to be used for the migration.
-Y	Specifies that the required software license agreements for the software to be installed are accepted. The **nimadm** command will not run unless you specify the **-Y** flag. You can also set the shell environment variable `ADM_ACCEPT_LICENSES` to `yes` to achieve the same result.
-r	Specifies that the client is rebooted after the migration completes.

The following example shows an example output of the **nimadm** command:

```
root@lpar04:/ # nimadm -c lpar02 -l 520lpp_res -s 520spot_res -d hdisk2 -Y
-r
Initializing the NIM master.
Initializing NIM client lpar02.itsc.austin.ibm.com.
Verifying alt_disk_migration eligibility.
Initializing log: /var/adm/ras/alt_mig/lpar02_alt_mig.log
Starting Alternate Disk Migration.

+-----------------------------------------------------------------------------+
Executing nimadm phase 1.
+-----------------------------------------------------------------------------+
Cloning altinst_rootvg on client, Phase 1.
Client alt_disk_install command: alt_disk_install -M 5.2 -C -P1 hdisk2
Calling mkszfile to create new /image.data file.
Checking disk sizes.
Creating cloned rootvg volume group and associated logical volumes.
Creating logical volume alt_hd5
Creating logical volume alt_hd6
Creating logical volume alt_hd8
Creating logical volume alt_hd4
Creating logical volume alt_hd2
Creating logical volume alt_hd9var
Creating logical volume alt_hd3
Creating logical volume alt_hd1
Creating logical volume alt_hd10opt
Creating /alt_inst/ file system.
Creating /alt_inst/home file system.
Creating /alt_inst/opt file system.
Creating /alt_inst/tmp file system.
Creating /alt_inst/usr file system.
Creating /alt_inst/var file system.
Generating a list of files
for backup and restore into the alternate file system...
Backing-up the rootvg files and restoring them to the
alternate file system...
Phase 1 complete.

+-----------------------------------------------------------------------------+
Executing nimadm phase 2.
+-----------------------------------------------------------------------------+
Exporting alt_inst filesystems from client lpar02.itsc.austin.ibm.com
to NIM master lpar04.itsc.austin.ibm.com:
Exporting /alt_inst from client.
Exporting /alt_inst/home from client.
Exporting /alt_inst/opt from client.
Exporting /alt_inst/tmp from client.
Exporting /alt_inst/usr from client.
Exporting /alt_inst/var from client.

+-----------------------------------------------------------------------------+
Executing nimadm phase 3.
+-----------------------------------------------------------------------------+
NFS mounting client's alt_inst filesystems on the NIM master:
Mounting lpar02.itsc.austin.ibm.com:/alt_inst.
Mounting lpar02.itsc.austin.ibm.com:/alt_inst/home.
Mounting lpar02.itsc.austin.ibm.com:/alt_inst/opt.
Mounting lpar02.itsc.austin.ibm.com:/alt_inst/tmp.
Mounting lpar02.itsc.austin.ibm.com:/alt_inst/usr.
Mounting lpar02.itsc.austin.ibm.com:/alt_inst/var.

+-----------------------------------------------------------------------------+
Executing nimadm phase 4.
+-----------------------------------------------------------------------------+
nimadm: There is no user customization script specified for this phase.

+-----------------------------------------------------------------------------+
Executing nimadm phase 5.
+-----------------------------------------------------------------------------+
Saving system configuration files.
```

```
Checking for initial required migration space.
Expanding /alt_inst/ client filesystem.
Filesystem size changed to 131072
Setting up for base operating system restore.
Restoring base operating system.
Restoring device ODM database.
Merging system configuration files.
Running migration merge method: ODM_merge SWservAt.
Running migration merge method: convert_errnotify.
Running migration merge method: passwd_mig.
Running migration merge method: login_mrg.
Running migration merge method: user_mrg.
Running migration merge method: secur_mig.
Running migration merge method: mkusr_mig.
Running migration merge method: group_mig.
Running migration merge method: ldapcfg_mig.
Running migration merge method: convert_errlog.
Running migration merge method: merge_smit_db.
Running migration merge method: ODM_merge fix.
Running migration merge method: merge_swvpds.

+-----------------------------------------------------------------------------+
Executing nimadm phase 6.
+-----------------------------------------------------------------------------+
Installing and migrating software.
Checking space requirements for installp install.
Expanding /alt_inst/usr client filesystem.
Filesystem size changed to 2097152
Installing software with the installp installer.
+-----------------------------------------------------------------------------+
                    Pre-installation Verification...
+-----------------------------------------------------------------------------+
Verifying selections...done
Verifying requisites...done
Results...
...
... omitted lines ...
...
csm.msg.en_US.dsh          1.3.0.0        USR        APPLY      SUCCESS
csm.msg.en_US.client       1.3.0.0        USR        APPLY      SUCCESS

Checking space requirements for rpm install.
Installing software with the rpm installer.
package cdrecord-1.9-4 is already installed
package mkisofs-1.13-4 is already installed

+-----------------------------------------------------------------------------+
Executing nimadm phase 7.
+-----------------------------------------------------------------------------+
nimadm: There is no user customization script specified for this phase.

+-----------------------------------------------------------------------------+
Executing nimadm phase 8.
+-----------------------------------------------------------------------------+
Creating client boot image.
bosboot: Boot image is 16758 512 byte blocks.
Writing boot image to client's alternate boot disk hdisk2.

+-----------------------------------------------------------------------------+
Executing nimadm phase 9.
+-----------------------------------------------------------------------------+
Unmounting client NFS mounts on the NIM master:
forced unmount of /lpar02_alt/alt_inst/var
forced unmount of /lpar02_alt/alt_inst/usr
forced unmount of /lpar02_alt/alt_inst/tmp
forced unmount of /lpar02_alt/alt_inst/opt
forced unmount of /lpar02_alt/alt_inst/home
forced unmount of /lpar02_alt/alt_inst

+-----------------------------------------------------------------------------+

+-----------------------------------------------------------------------------+
Executing nimadm phase 10.
+-----------------------------------------------------------------------------+
Unexporting alt_inst filesystems on client lpar02.itsc.austin.ibm.com:
exportfs: 1831-184 unexported /alt_inst
exportfs: 1831-184 unexported /alt_inst/home
exportfs: 1831-184 unexported /alt_inst/opt
exportfs: 1831-184 unexported /alt_inst/tmp
exportfs: 1831-184 unexported /alt_inst/usr
exportfs: 1831-184 unexported /alt_inst/var

+-----------------------------------------------------------------------------+
Executing nimadm phase 11.
+-----------------------------------------------------------------------------+
Cloning altinst_rootvg on client, Phase 3.
Client alt_disk_install command: alt_disk_install -M 5.2 -C -P3 hdisk2
## Phase 3 ###################
Verifying altinst_rootvg...
Modifying ODM on cloned disk.
forced unmount of /alt_inst/var
forced unmount of /alt_inst/usr
forced unmount of /alt_inst/tmp
forced unmount of /alt_inst/opt
forced unmount of /alt_inst/home
forced unmount of /alt_inst
forced unmount of /alt_inst
Changing logical volume names in volume group descriptor area.
```

```
Fixing LV control blocks...
Fixing file system superblocks...
Bootlist is set to the boot disk: hdisk2

+-----------------------------------------------------------------------------+
Executing nimadm phase 12.
+-----------------------------------------------------------------------------+
Cleaning up alt_disk_migration on the NIM master.
Cleaning up alt_disk_migration on client lpar02.
Rebooting NIM client.
```

7. After lpar02 reboots, verify that it has been successfully migrated to AIX 5L Version 5.2:

```
root@lpar02:/ [301] # oslevel
5.2.0.0
```

Even though the **oslevel** command might show that we are already at level 5.2.0.0, we recommend you look at the **nimadm** log file, /var/adm/ras/alt_mig/<host_name>_alt_mig.log, on the NIM master to find out whether or not there are any filesets that failed the migration. For example, fileset migration can fail if the newer version of the fileset itself, or the prerequisite filesets, cannot be found in the lppsource.

Therefore, the safest way is to make sure that all filesets in the AIX BOS installation CD-ROM media are in the lppsource resource.

8

DLPAR operation using graphical user interface

This chapter describes how to use the IBM Hardware Management Console for pSeries (HMC) to perform DLPAR operations against to AIX partitions on partioning-capable pSeries servers.

For further information about the HMC and its usage, please refer to *IBM Hardware Management Console for pSeries Installation and Operations Guide*, SA38-0590.

8.1 Dynamic logical partitioning

Dynamic logical partition provides the ability to logically attach and detach a managed system's resources to and from a partition's operating system without rebooting.

Note: You can dynamically reassign I/O resources between affinity logical partitions, but not processor or memory resources.

8.1.1 Dynamically adding resources to a partition

This task enables you to add resources, such as processors, memory, and I/O slots, to a partition without rebooting the partition's operating system.

Processors

You can add up to the amount of *free* system processors (processors that are not assigned to a running partition) to a partition. You cannot exceed the *maximum* number specified in the partition's active profile. To view profile properties, see Figure 6-4 on page 169.

To add available processor resources without rebooting the partition, do the following:

1. Log in to the HMC using either the System Administrator or Advanced Operator role.
2. In the Navigation area, click the console icon to expand the tree.
3. In the Navigation area, click the Server and Partition folder.
4. In the Contents area, click the Server Management icon.
5. In the Contents area, click the managed system icon to expand the tree.
6. Select the partition to which you want to add the processors.

7. From the menu bar, select **Selected** → **Dynamic Logical Partitioning** → **Processors**, as shown Figure 8-1.

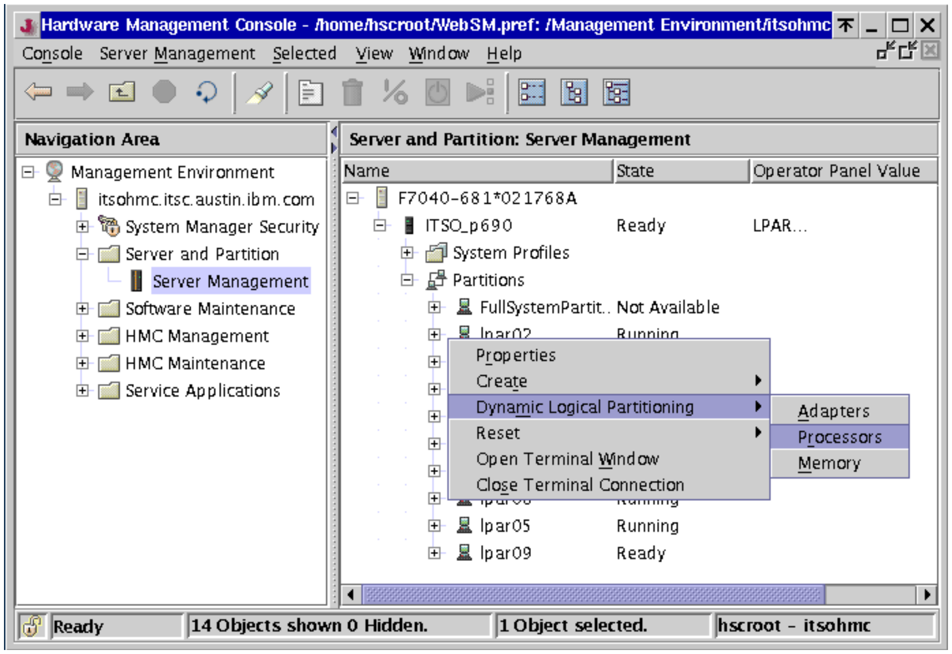
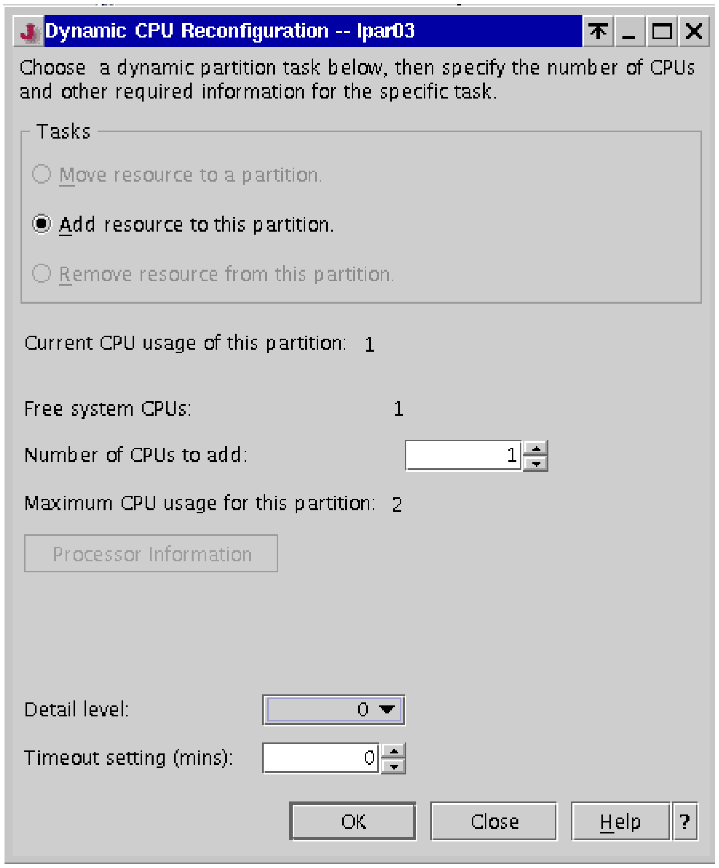


Figure 8-1   Dynamic logical partitioning

8. The Dynamic CPU Reconfiguration window opens. Select **Add resource to this partition**, as shown in Figure 8-2.


Dynamic CPU Reconfiguration -- lpar03

Choose a dynamic partition task below, then specify the number of CPUs and other required information for the specific task.

Tasks
- Move resource to a partition.
- [✓ Add resource to this partition.]
- Remove resource from this partition.

Current CPU usage of this partition: 1

Free system CPUs: 1

Number of CPUs to add: 1

Maximum CPU usage for this partition: 2

Detail level: 0

Timeout setting (mins): 0

Figure 8-2 Dynamic CPU addition to a partition

9. Select the number of processors you want to add to this partition in the Number of CPUs to add field.

   In this example, we have been able to select only 1 in this field, because the Maximum CPU usage of this partition field is set to 2.

Note: If the Processor Information button appears underneath the Number of CPUs to add field, the HMC has discovered disabled processors that you might be able to deconfigure and free for system use. For more information about restoring these processors, see Appendix C, “Error Messages and Recovery Information,” in *IBM Hardware Management Console for pSeries Installation and Operations Guide*, SA38-0590.

10. In the Timeout setting field, select the number of minutes you want the system to wait before it stops the processor dynamic logical partitioning task.
11. In the Detail level field, select the level of feedback you would like to see while the HMC performs the task. Details shown include the operating system's standard output and standard error information.
12. When you are finished, click **OK**.

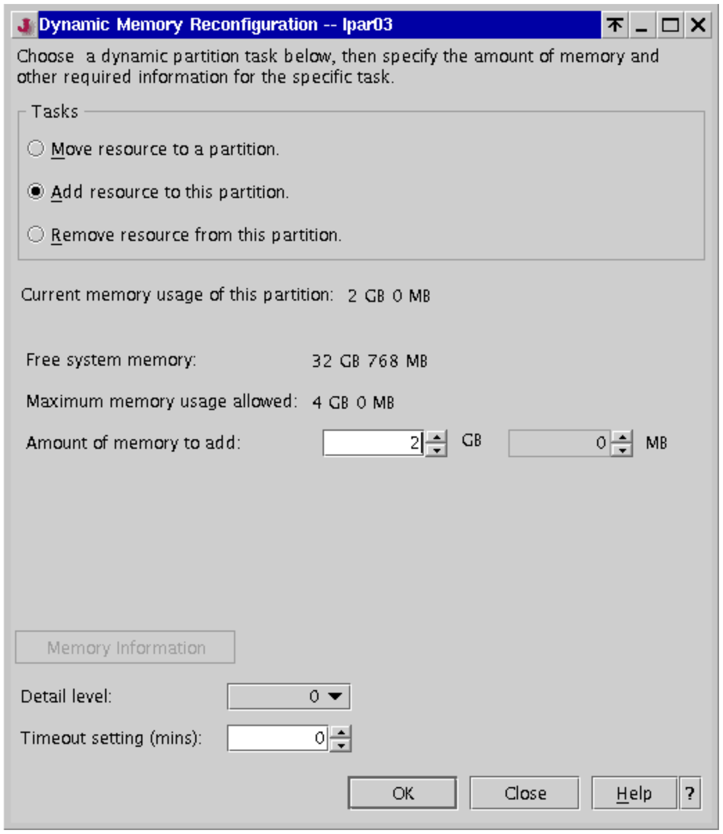
Memory

This task enables you to add memory to a partition without rebooting the partition's operating system. You can only add up to the amount of *free* memory, or memory that is not assigned to a running partition. You also cannot exceed the *maximum* number specified in the partition's active profile. To view profile properties, see Figure 6-5 on page 170.

To add available memory resources without rebooting the partition, do the following:

1. Log in to the HMC using either the System Administrator or Advanced Operator role.
2. In the Navigation area, click the console icon to expand the tree.
3. In the Navigation area, click the Server and Partition folder.
4. In the Contents area, click the Server Management icon.
5. In the Contents area, click the managed system icon to expand the tree.
6. Select the partition to which you want to add the memory.
7. From the menu bar, select **Selected** → **Dynamic Logical Partitioning** → **Memory** (see Figure 8-1 on page 257).

8. The Dynamic Memory Reconfiguration window opens. Select **Add resource to this partition**, as shown in Figure 8-3.


Dynamic Memory Reconfiguration -- lpar03

Choose a dynamic partition task below, then specify the amount of memory and other required information for the specific task.

Tasks

○ Move resource to a partition.

● Add resource to this partition.

○ Remove resource from this partition.

Current memory usage of this partition: 2 GB 0 MB

Free system memory: 32 GB 768 MB

Maximum memory usage allowed: 4 GB 0 MB

Amount of memory to add: 2 GB 0 MB

Memory Information

Detail level: 0

Timeout setting (mins): 0

Figure 8-3 Dynamic memory addition to a partition

9. Select the amount of memory you want to add to this partition. The window shows you how much free memory the system has for this partition's use.

   In this example, we are able to select up to 2 GB in this field, because the Maximum memory usage allowed field is set to 4 GB.

10. In the Timeout setting field, select the number of minutes you want the system to wait before it stops the memory dynamic logical partitioning task.

11. In the Detail level field, select the level of feedback you would like to see while the HMC performs the task. Details shown include the operating system's standard output and standard error information.

12. When you are finished, click **OK**.

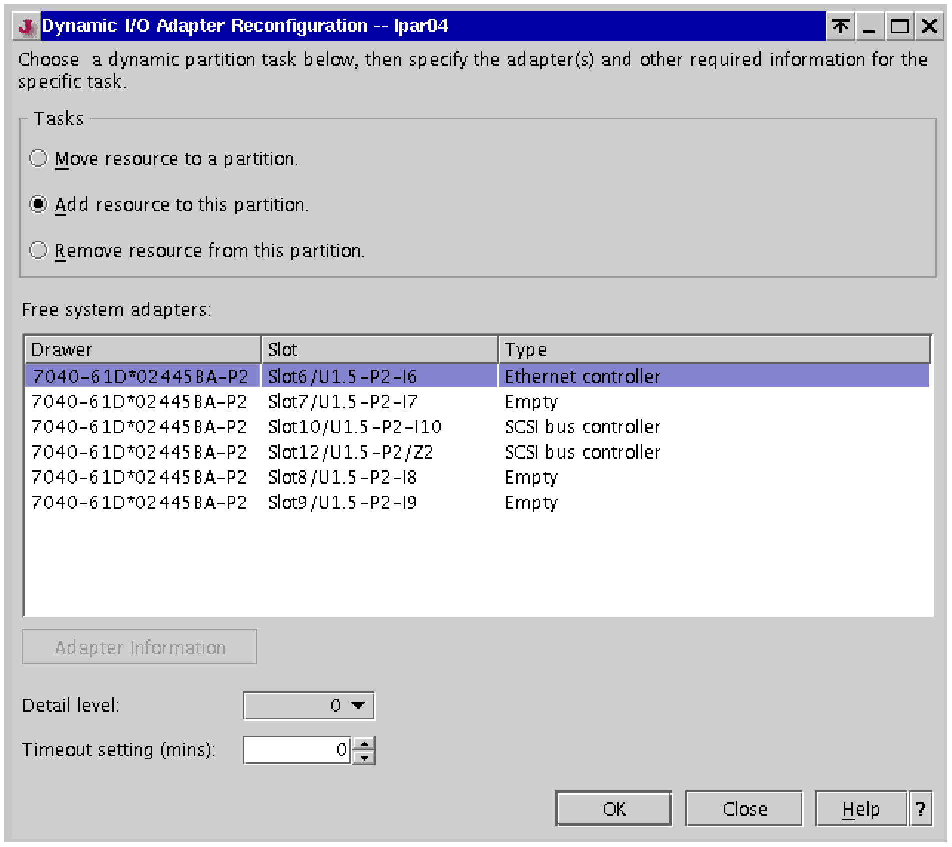
Adapters

This task enables you to add I/O slots to a partition without rebooting the partition's operating system. The following is a task description of adding an I/O slot containing a PCI adapter to a partition dynamically without rebooting the partition. You can dynamically add any *free* adapters to the partition.

To add I/O slots to a partition without rebooting the partition, do the following:

1. Log in to the HMC using either the System Administrator or Advanced Operator role.
2. In the Navigation area, click the console icon to expand the tree.
3. In the Navigation area, click the Server and Partition folder.
4. In the Contents area, click the Server Management icon.
5. In the Contents area, click the managed system icon to expand the tree.
6. Select the partition to which you want to add the adapters.
7. From the menu bar, select **Selected** → **Dynamic Logical Partitioning** → **Adapters**, as shown in Figure 8-1 on page 257.

8. The Dynamic I/O Adapter Reconfiguration window opens. Select **Add resource to this partition**, as shown in Figure 8-4.


Dynamic I/O Adapter Reconfiguration -- lpar04

Choose a dynamic partition task below, then specify the adapter(s) and other required information for the specific task.

Tasks

( ) Move resource to a partition.
[✓ Add resource to this partition.]
( ) Remove resource from this partition.

Free system adapters:

Drawer	Slot	Type
7040-61D*02445BA-P2	Slot6/U1.5-P2-I6	Ethernet controller
7040-61D*02445BA-P2	Slot7/U1.5-P2-I7	Empty
7040-61D*02445BA-P2	Slot10/U1.5-P2-I10	SCSI bus controller
7040-61D*02445BA-P2	Slot12/U1.5-P2/Z2	SCSI bus controller
7040-61D*02445BA-P2	Slot8/U1.5-P2-I8	Empty
7040-61D*02445BA-P2	Slot9/U1.5-P2-I9	Empty

Detail level: 0

Timeout setting (mins): 0

Figure 8-4 Dynamic I/O slot addition to a partition

9. Select the free system adapters that you want to add to this partition.

   The adapters shown in the Free system adapters field are the ones not assigned to the other running partitions.

   Note: If the Adapter Information button appears underneath the Free system adapters field, the HMC has discovered disabled adapters that you might be able to deconfigure and free for system use. For more information about restoring these adapters, see Appendix C, “Error Messages and Recovery Information,” in *IBM Hardware Management Console for pSeries Installation and Operations Guide*, SA38-0590.

10. In the Timeout setting field, select the number of minutes you want the system to wait before it stops the adapter dynamic logical partitioning task.
11. In the Detail level field, select the level of feedback you would like to see while the HMC performs the task. Details shown include the operating system's standard output and standard error information.
12. When you are finished, click **OK**.

Note: After adding the I/O slot, you need manual interaction on the partition. See 3.3.2, "Internal activity for I/O slots in a DLPAR event" on page 63.

8.1.2 Dynamically moving resources between partitions

This task enables you to move resources, such as processors, memory, and I/O slots, from one partition to another without rebooting either partition's operating system.

Processors

You can move processors from one partition to another partition. You have to select appropriate values so that these two partitions satisfy the following conditions:

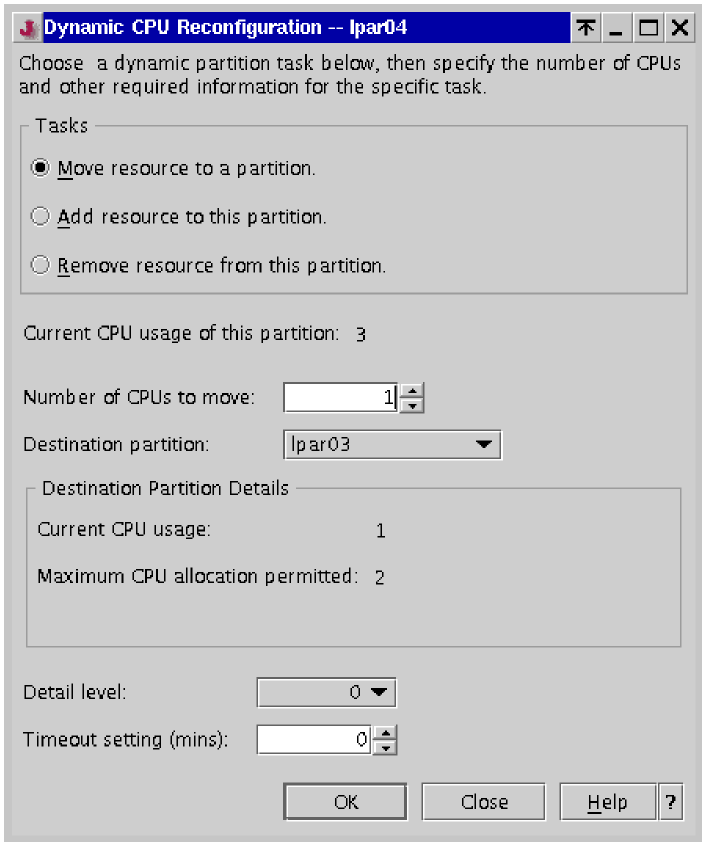
- The partition whose processors will be removed must be assigned more than or equal to the number of processors that is defined in the partition profile as the minimum value after the operation.
- The partition whose processors will be added must be assigned less than or equal to the number of processors that is defined in the partition profile as the maximum value after the operation.

To view profile properties, see Figure 6-4 on page 169.

To move processors from one active partition to another without rebooting either partition, do the following:

1. Log in to the HMC using either the System Administrator or Advanced Operator role.
2. In the Navigation area, click the console icon to expand the tree.
3. In the Navigation area, click the Server and Partition folder.
4. In the Contents area, click the Server Management icon.
5. In the Contents area, click the managed system icon to expand the tree.
6. Select the partition from which you want to move the processors.

7. From the menu bar, select **Selected** → **Dynamic Logical Partitioning** → **Processors**, as shown in Figure 8-1 on page 257.
8. The Dynamic CPU Reconfiguration window opens. Select **Move resource to a partition**, as shown in Figure 8-5.



Choose a dynamic partition task below, then specify the number of CPUs and other required information for the specific task.

Tasks

- [✓ Move resource to a partition.]
- Add resource to this partition.
- Remove resource from this partition.

Current CPU usage of this partition: 3

Number of CPUs to move: 1

Destination partition: lpar03

Destination Partition Details

Current CPU usage: 1

Maximum CPU allocation permitted: 2

Detail level: 0

Timeout setting (mins): 0

Figure 8-5 Dynamic CPU reconfiguration between partitions

9. Select the number of processors you want to move from this partition.

> **Note:** The number of processors you remove from the source partition cannot make the remaining number of processors be less than the minimum number specified in this partition's active profile. Likewise, the number you are adding to the target partition cannot exceed the destination partition's maximums.

10. Select name of the partition to which you want to move the processors.

11. In the Timeout setting field, select the number of minutes you want the system to wait before it stops the move processor dynamic logical partitioning task.
12. In the Detail level field, select the level of feedback you would like to see while the HMC performs the task. Details shown include the operating system's standard output and standard error information.
13. When you are finished, click **OK**. The processors are moved from this partition to the partition you selected.

Memory

You can move memory from one partition to another partition. You have to select appropriate values so that these two partitions satisfy the following conditions:

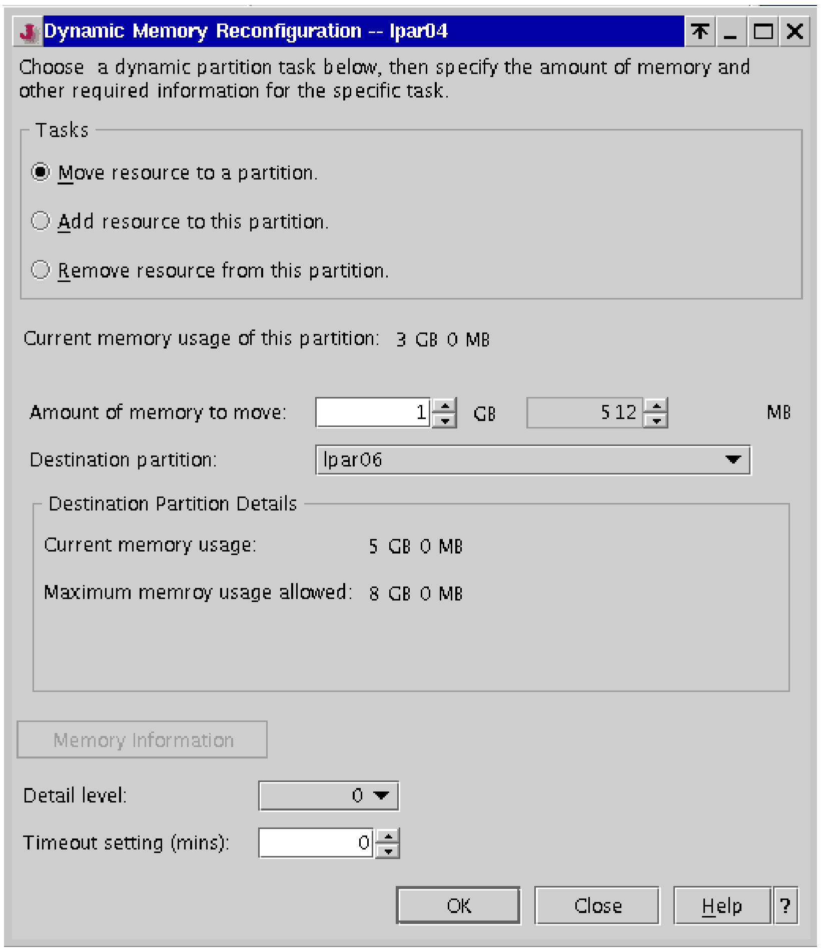
- The partition whose memory will be removed must be assigned more than or equal to the memory size that is defined in the partition profile as the minimum value after the operation.
- The partition whose memory will be added must be assigned less than or equal to the memory size that is defined in the partition profile as the maximum value after the operation.

To view profile properties, see Figure 6-5 on page 170.

To move memory from one active partition to another without rebooting either partition, do the following:

1. Log in to the HMC using either the System Administrator or Advanced Operator role.
2. In the Navigation area, click the console icon to expand the tree.
3. In the Navigation area, click the Server and Partition folder.
4. In the Contents area, click the Server Management icon.
5. In the Contents area, click the managed system icon to expand the tree.
6. Select the partition from which you want to move the memory.
7. From the menu bar, select **Selected** → **Dynamic Logical Partitioning** → **Memory**, as shown in Figure 8-1 on page 257.

8. The Dynamic Memory Reconfiguration window opens. Select **Move resource to a partition**, as shown in Figure 8-6.



Dynamic Memory Reconfiguration -- lpar04

Choose a dynamic partition task below, then specify the amount of memory and other required information for the specific task.

Tasks

(●) Move resource to a partition.

( ) Add resource to this partition.

( ) Remove resource from this partition.

Current memory usage of this partition: 3 GB 0 MB

Amount of memory to move: 1 GB 512 MB

Destination partition: lpar06

Destination Partition Details

Current memory usage: 5 GB 0 MB

Maximum memroy usage allowed: 8 GB 0 MB

Detail level: 0

Timeout setting (mins): 0

Figure 8-6 Dynamic memory reconfiguration between partitions

9. Select the memory size you want to move from this partition.

Note: The memory you are removing from the source partition cannot make the remaining memory amount be less than the minimum number specified in this partition’s active profile. Likewise, the memory you are adding to the target partition cannot exceed the destination partition’s maximums.

10. Select the name of the partition to which you want to move the memory.

Note: If the Memory Information button appears in this window, the HMC has discovered an inconsistency between a partition's Allocated and Requested memory amounts. Click this button to correct the requested memory value and free memory resources to the system.

11. In the Timeout setting field, select the number of minutes you want the system to wait before it stops the memory dynamic logical partitioning task.
12. In the Detail level field, select the level of feedback you would like to see while the HMC performs the task. Details shown include the operating system's standard output and standard error information.
13. When you are finished, click **OK**.

Adapters

This task enables you to move a PCI I/O slot containing a PCI adapter from one partition to another without rebooting the partition's operating system. Before moving an I/O slot, you must to log in to the OS on the source partition and unconfigure the adapter and the I/O slot (see 3.3.2, "Internal activity for I/O slots in a DLPAR event" on page 63). The adapter that you are going to move must not be defined as Required in the current active partition profile on the source partition.

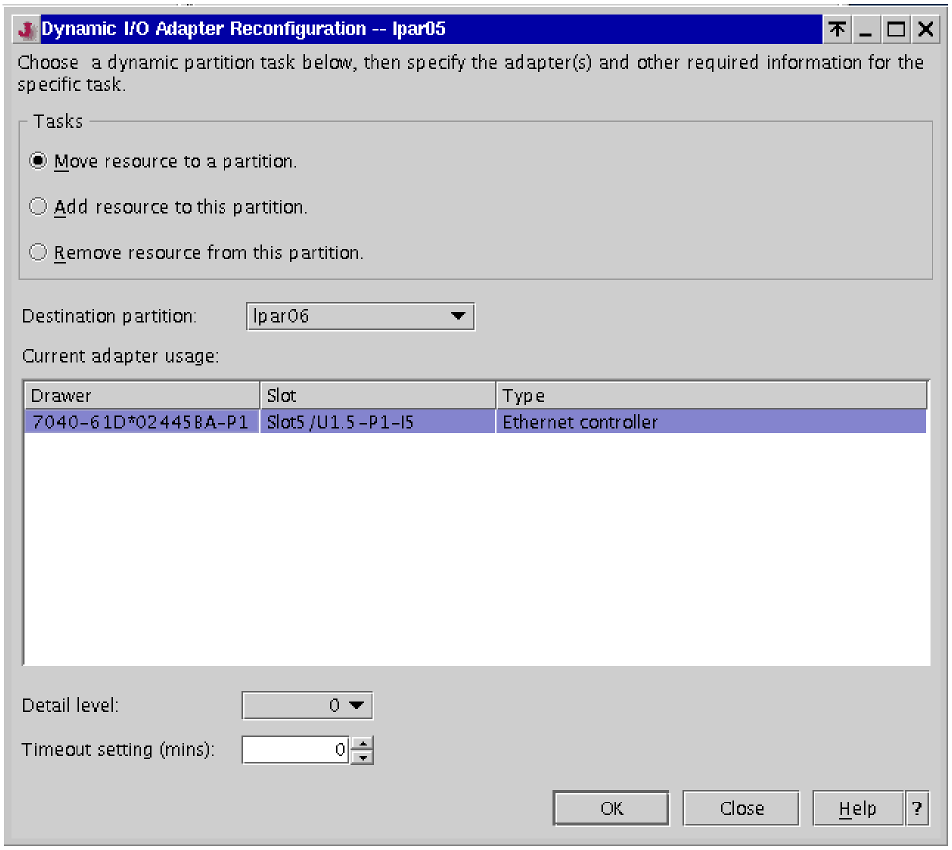
Note: To ensure that Service Focal Point and DLPAR operations continue to function correctly, do not dynamically move the Ethernet adapter, which is used to communicate with the HMC.

To move adapter resources from one active partition to another without rebooting either partition, do the following:

Note: Before moving the I/O slot, you need manual interaction on the source partition. See 3.3.2, "Internal activity for I/O slots in a DLPAR event" on page 63.

1. Log in to the HMC using either the System Administrator or Advanced Operator role.
2. In the Navigation area, click the console icon to expand the tree.
3. In the Navigation area, click the Server and Partition folder.
4. In the Contents area, click the Server Management icon.
5. In the Contents area, click the managed system icon to expand the tree.
6. Select the partition from which you want to move the adapters.

7. From the menu bar, select **Selected** → **Dynamic Logical Partitioning** → **Adapters**, as shown in Figure 8-1 on page 257.
8. The Dynamic I/O Reconfiguration window opens. Select **Move resource to a partition**, as shown in Figure 8-7.


Dynamic I/O Adapter Reconfiguration -- lpar05

Choose a dynamic partition task below, then specify the adapter(s) and other required information for the specific task.

Tasks

- [✓ Move resource to a partition.]
- Add resource to this partition.
- Remove resource from this partition.

Destination partition: lpar06

Current adapter usage:

Drawer	Slot	Type
7040-61D*024458A-P1	Slot5 /U1.5-P1-I5	Ethernet controller

Detail level: 0

Timeout setting (mins): 0

Figure 8-7 Dynamic adapter reconfiguration between partitions

9. Select the I/O adapters you want to move from the list. Adapters designated as *Required* in this partition's active profile are not included in this list and cannot be removed (see Figure 6-6 on page 172).
10. Select the partition to which you would like move the adapters.
11. In the Timeout setting field, select the number of minutes you want the system to wait before it stops the adapter dynamic logical partitioning task.
12. In the Detail level field, select the level of feedback you would like to see while the HMC performs the task. Details shown include the operating system's standard output and standard error information.

13. When you are finished, click **OK**. Adapters are then moved from this partition to the partition you selected. Now, you must log in to the other partition's operating system and configure the adapter.

Note: After moving the I/O slot, you need manual interaction on the target partition. See 3.3.2, "Internal activity for I/O slots in a DLPAR event" on page 63.

8.1.3 Dynamically removing resources from a partition

This task allows you to remove resources, such as processors, memory, and I/O slots, from a partition without rebooting the partition's operating system.

Processors

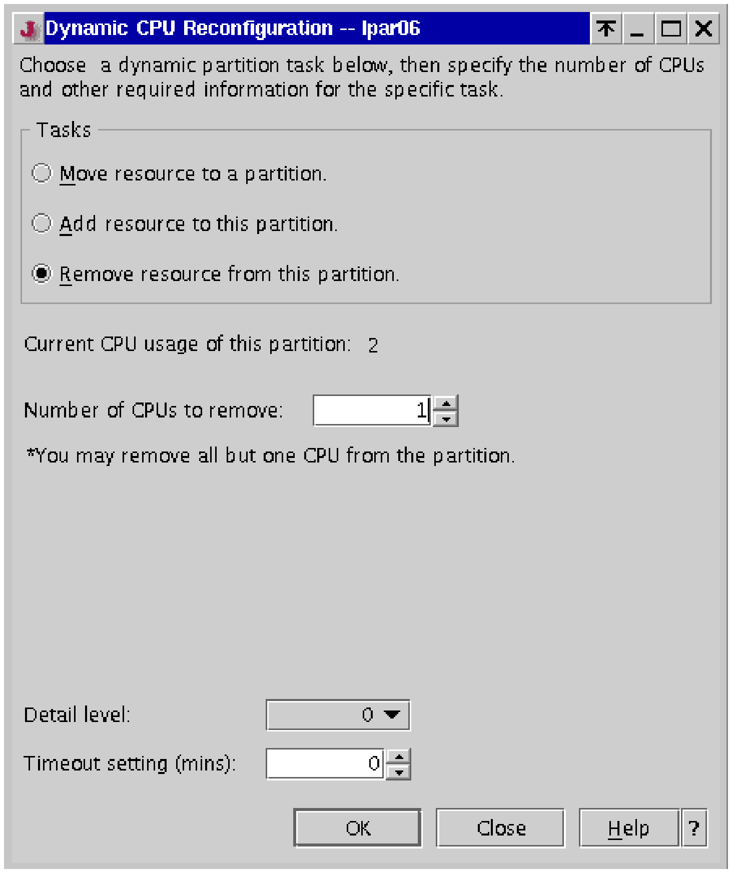
You can remove processors from a partition without rebooting the partition's operating system. When you remove a processor, it is freed by the partition and available for use by other partitions.

Note: The number of processors remaining after the removal operation cannot be less than the minimum value specified in this partition's active profile (see Figure 6-4 on page 169).

To remove processor from a partition without rebooting, do the following:

1. Log in to the HMC using either the System Administrator or Advanced Operator role.
2. In the Navigation area, click the console icon to expand the tree.
3. In the Navigation area, click the Server and Partition folder.
4. In the Contents area, click the Server Management icon.
5. In the Contents area, click the managed system icon to expand the tree.
6. Select the partition from which you want to remove the processors.
7. From the menu bar, select **Selected** → **Dynamic Logical Partitioning** → **Processors** (see Figure 8-1 on page 257).

8. The Dynamic CPU Reconfiguration window opens. Select **Remove resource from this partition**, as shown in Figure 8-8.



Dynamic CPU Reconfiguration -- lpar06

Choose a dynamic partition task below, then specify the number of CPUs and other required information for the specific task.

Tasks

- Move resource to a partition.
- Add resource to this partition.
- [✓ Remove resource from this partition.]

Current CPU usage of this partition: 2

Number of CPUs to remove: 1

*You may remove all but one CPU from the partition.

Detail level: 0

Timeout setting (mins): 0

Figure 8-8 Dynamic CPU removal from a partition

9. Select the number of processors you want to remove from this partition.

10. In the Timeout setting field, select the number of minutes you want the system to wait before it stops the processor dynamic logical partitioning task.

11. In the Detail level field, select the level of feedback you would like to see while the HMC performs the task. Details shown include the operating system's standard output and standard error information.

12. When you are finished, click **OK**.

Memory

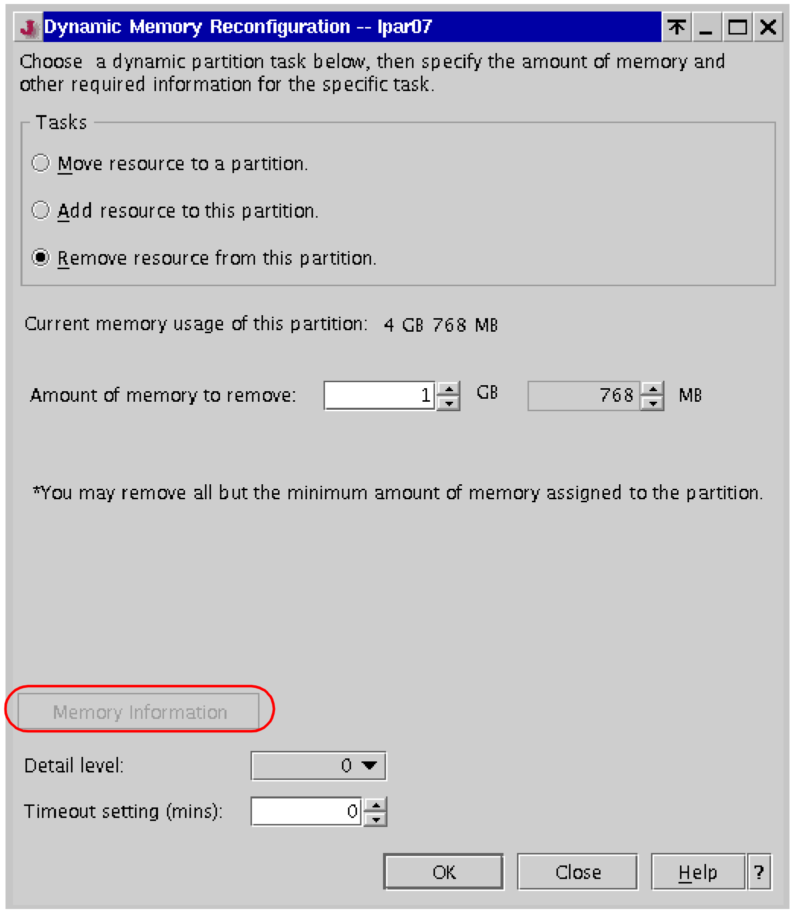
You can remove memory from a partition without rebooting the partition's operating system. When you remove memory, it is freed by the partition and available for use by other partitions.

Note: The size of memory remaining after the removal operation cannot be less than the minimum value specified in this partition's active profile (see Figure 6-5 on page 170).

To remove memory from a partition without rebooting, do the following:

1. Log in to the HMC using either the System Administrator or Advanced Operator role.
2. In the Navigation area, click the console icon to expand the tree.
3. In the Navigation area, click the Server and Partition folder.
4. In the Contents area, click the Server Management icon.
5. In the Contents area, click the managed system icon to expand the tree.
6. Select the partition from which you want to remove the memory.
7. From the menu bar, select **Selected** → **Dynamic Logical Partitioning** → **Memory** (see Figure 8-1 on page 257).

8. The Dynamic Memory Reconfiguration window opens. Select **Remove resource from this partition**, as shown in Figure 8-9.


Dynamic Memory Reconfiguration -- lpar07

Choose a dynamic partition task below, then specify the amount of memory and other required information for the specific task.

Tasks

Move resource to a partition. [ ]

Add resource to this partition. [ ]

[✓ Remove resource from this partition.]

Current memory usage of this partition: 4 GB 768 MB

Amount of memory to remove: 1 GB 768 MB

*You may remove all but the minimum amount of memory assigned to the partition.

Detail level: 0

Timeout setting (mins): 0

Figure 8-9 Dynamic memory removal from a partition

9. Select the amount of memory you want to remove from this partition.

   If the Memory Information button is available in this window, the HMC has discovered an inconsistency between a partition's Allocated and Requested memory amounts. Click this button to correct the requested memory value and free memory resources to the system. For more information about restoring memory, see Appendix C, "Error Messages and Recovery Information," in the *IBM Hardware Management Console for pSeries Installation and Operations Guide*, SA38-0590.

10. In the Timeout setting field, select the number of minutes you want the system to wait before it stops the memory dynamic logical partitioning task.

11. In the Detail level field, select the level of feedback you would like to see while the HMC performs the task. Details shown include the operating system's standard output and standard error information.
12. When you are finished, click **OK**.

Note: If many virtual memory pages are used by application on the target partition, the operating system might crash after the dynamic memory removal operation because of the low paging space. In order to avoid this situation, keep the at least following amount of paging space on the partition:

(paging space required in worst case) + (memory max) - (memory min)

Adapters

This task enables you to remove I/O slots, which can contain an adapter, from a partition without rebooting the partition's operating system. Before continuing with this task, you must use the partition's operating system to manually deconfigure each adapter that you want to remove. You cannot remove an adapter defined as Required in the current active partition profile.

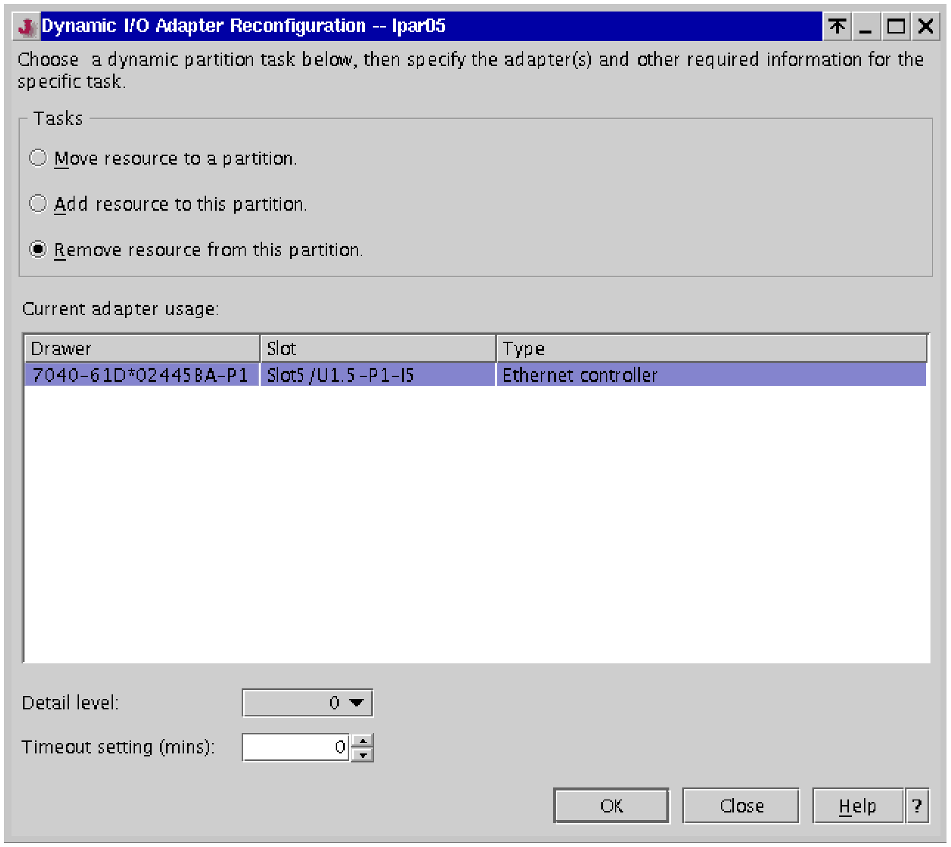
To learn more about this partition's active profile, view the activated profile's properties. To view profile properties, see Chapter 15, "User Management Tasks," in the *IBM Hardware Management Console for pSeries Installation and Operations Guide*, SA38-0590.

To remove adapter resources from one active partition to another without rebooting either partition, do the following:

Note: Before removing the I/O slot, you need manual interaction on the partition. See 3.3.2, "Internal activity for I/O slots in a DLPAR event" on page 63.

1. Log in to the HMC using either the System Administrator or Advanced Operator role.
2. In the Navigation area, click the console icon to expand the tree.
3. In the Navigation area, click the Server and Partition folder.
4. In the Contents area, click the Server Management icon.
5. In the Contents area, click the managed system icon to expand the tree.
6. Select the partition from which you want to remove the adapters.
7. From the menu bar, select **Selected** → **Dynamic Logical Partitioning** → **Adapters**, as shown in Figure 8-1 on page 257.

8. The Dynamic I/O Adapter Reconfiguration window opens. Select **Remove resource from a partition**, as shown in Figure 8-10.


Dynamic I/O Adapter Reconfiguration -- lpar05

Choose a dynamic partition task below, then specify the adapter(s) and other required information for the specific task.

Tasks

( ) Move resource to a partition.

( ) Add resource to this partition.

(●) Remove resource from this partition.

Current adapter usage:

Drawer	Slot	Type
7040-61D*024458A-P1	Slot5 /U1.5-P1-I5	Ethernet controller

Detail level: 0

Timeout setting (mins): 0

Figure 8-10 Dynamic removal of an I/O slot from a partition

9. Select the adapters that you want to remove from the partition.

Note: Adapters designated as *Required* in this partition's active profile are not included in this list and cannot be removed.

10. In the Timeout setting field, select the number of minutes you want the system to wait before it stops the adapter dynamic logical partitioning task.
11. In the Detail level field, select the level of feedback you would like to see while the HMC performs the task. Details shown include the operating system's standard output and standard error information.
12. When you are finished, click **OK**.

9

DLPAR operation using a command line interface

This chapter provides information on how to use the command line interface on the IBM Hardware Management Console for pSeries (HMC) to be used for DLPAR operations. The command line interface is sometimes quite useful, especially in the following two situations:

- Consistent results are required.

  If you have to administer several managed systems, you can achieve consistent results by using the command line interface. The command sequence can be stored in scripts and executed remotely.

- Automated operations are required.

  After you have developed a consistent way to manage your managed systems, you can automate the operations by invoking the scripts from batch processing applications, such as the cron daemon, from other systems.

For further information about the command line interface on the HMC, refer to *IBM Hardware Management Console for pSeries Installation and Operations Guide*, SA38-0590.

9.1 Secure remote connection to the HMC

The HMCs are typically placed inside the machine room where managed systems are located. Therefore, you might not be allowed to physically access the HMC. In this case, you have to remotely access the HMC to use the command line interface. The HMC supports two commands to be remotely accessed: **`ssh`** and **`rexec`**. Throughout this redbook, we assume that you use OpenSSH to securely connect between AIX systems (including partitions) and the HMC.

9.1.1 Connection to the HMC for command line operations

You have two ways to remotely execute the command line interface on the HMC using OpenSSH:

- Execute commands remotely.

  Example 9-1 shows that the **`df -k`** command is remotely executed using the **`ssh`** command. In this example, the remote user (user) on the remote host (host1) is executing the command as hscroot user on the HMC (itsohmc). You will be prompted to enter the password of the hscroot user, and then the command will be executed.

Example 9-1 Remote command execution

```
user@host1:/ # ssh -l hscroot itsohmc df -k
hscroot@itsohmc's password:
Filesystem           1k-blocks      Used Available Use% Mounted on
/dev/hda5             10130580    241632   9580260   3% /
/dev/hda9              2015440        24   1994936   1% /console
none                    256544         0    256544   0% /dev/shm
/dev/hda6              4031836    884464   3024476  23% /usr
/dev/hda7              4031836    113488   3795452   3% /var
```

- Execute commands after logging in to the HMC.

  Example 9-2 shows that the **`df -k`** command is executed on the HMC after logging to the HMC.

Example 9-2 Local command execution

```
user@host1:/ # ssh -l hscroot itsohmc
hscroot@itsohmc's password:
Last login: Thu Jul 24 19:47:37 2003 from host1
[hscroot@itsohmc hscroot]$ df -k
Filesystem           1k-blocks      Used Available Use% Mounted on
/dev/hda5             10130580    241632   9580260   3% /
/dev/hda9              2015440        24   1994936   1% /console
```

```
none                    256544         0    256544   0% /dev/shm
/dev/hda6              4031836    884464   3024476  23% /usr
/dev/hda7              4031836    113488   3795452   3% /var
```

Note: Copying any files onto the HMC not documented in any IBM publications should be avoided as it can cause complexity in problem determination by IBM support personnel. Therefore, all the supported commands should be executed via either the **rexec** or **ssh** facility from remote systems. If you need to have some administrative scripts, we suggest you store them on other systems besides the HMC.

9.1.2 Connection to the HMC for automated operations

In case we want to automate remote commands execution, by a scheduler for example, we need a remote connection to the HMC without being prompted for a password. **ssh** uses a pair of key files, private and public, to authenticate remote users.

After the public key file is copied over to the HMC, you will be prompted to enter a passphrase when you attempt to remotely log in to the HMC. Although it is desirable that a passphrase should be entered while creating the public key, it can be created with a null passphrase. If a null passphrase is used, you will not be prompted to remotely log in to the HMC after copying the public key file.

If a non-null passphrase is used, ssh-agent can be used in order to automate the connection. The agent is invoked with the passphrase and stores it in memory. For further information about how to use the ssh-agent in the automated connection environment on AIX, refer to *Managing AIX Server Farms*, SG24-6606.

9.2 Command line interface

In this section, we explain some useful commands to understand the DLPAR operations of section 9.3, “Dynamic logical partitioning operations using chhwres” on page 285.

Table 9-1 lists the commands to get information and modify a managed system.

Table 9-1 Commands list

Command	Function
lssyscfg	Get managed system information

Command	Function
lshwres	Get partition resources information
chsysstate	Change the state of the managed system
chhwres	Modify hardware resources of a partition

The commands listed in Table 9-1 are located in the /opt/hsc/bin directory and are issued by the hscroot user on the HMC.

For further information about commands available on HMC, refer to *Effective System Management using the IBM Hardware Management Console for pSeries*, SG24-7038.

9.2.1 Get system information using lssyscfg

This section explains how to use the command line interface to get information such as system status and partition name from the managed system.

Get information from the CEC

Example 9-3 shows how to use the **lssyscfg** command to get the managed system name, machine type and model, serial number, CEC version, state, and operator panel string.

Example 9-3 Get information from the CEC

```
[hscroot@itsohmc]$ lssyscfg -r sys --all
Name       CageNum  LMBSize  Mode  State  CSPVersion  Model     OpPanel S/N
ITSO_p690           256      255   Ready  V4.0        7040-681  LPAR... 021768A
```

Get defined partition list

Example 9-4 shows how to use the **lssyscfg** command to get the names of all defined partitions and the state of each partition on the managed system. Managed system ITSO_p690 has nine partitions defined: lpar06, lpar03, FullSystemPartition, lpar07, lpar04, lpar01, lpar08, lpar05, and lpar02. Partition lpar08 is running with profile SMS.

Example 9-4 Get defined partition list[1]

```
[hscroot@itsohmc]$ lssyscfg -r lpar -m ITSO_p690 --all
Name                 id   DLPAR  State          Profile               OpPanel
lpar06               006  15     Running        aix52_64
lpar03               003  15     Running        aix52_64
```

[1] The values shown in the DLPAR column are anticipated to be changed to YES (DLPAR-capable) and NO (DLPAR-incapable) in the future HMC software release.

```
FullSystemPartition  000  0      Not Available  PowerOnNormalProfile
lpar07               007  15     Running        aix52_32
lpar04               004  15     Running        aix52_64
lpar01               001  0      Ready          aix51_64
lpar08               008  15     Running        SMS
lpar05               005  15     Running        aix52_64
lpar02               002  0      Ready          aix51_64
```

Get informations from a partition

Example 9-5 shows how to use the **lssyscfg** command to get the profile list of a partition. It also gets information for each profile of the given partition. Profile aix52_64 requires 1 CPU and 512 MB of memory to start.

Example 9-5 Get partition informations

```
[hscroot@itsohmc]$ lssyscfg -r prof -m ITSO_p690 -p lpar05 --all
Name         BootMode  DesiredCPU  DesiredMEM  MaxCPU  MaxMEM  MinCPU  MinMEM
SMS          3         2           2048        6       8192    1       2048
aix52_64     1         2           1280        6       8192    1       512
OpenFirmware 4         2           2048        6       8192    1       2048
```

9.2.2 Get hardware resource information using lshwres

This section explains how to use the command line interface to get hardware resource information from a managed system.

The **lshwres** command lists a hardware resource configuration. The command takes the following options:

- **-m managed_system**

  Specify the managed system name.

- **-p partition_name**

  Specify the partition name.

- **-r [ALL | slot | cpu | mem | led]**

  Specify the hardware resource type to view:

ALL	All resources
slot	I/O slot
cpu	CPU
mem	Memory
led	LED

- **`-f format`**

  Specify the formatted fields to be listed.

To get a list of all processors, use **`lshwres`** with the **`-r cpu`** flag as shown in Example 9-6. Partition lpar05 has two processors, one with id 1 and the other with id 0. Processors with id 7, 19, 20, and 21 do not belong to any partition.

Example 9-6 lshwres for processors

```
[hscroot@itsohmc]$ lshwres -r cpu -m ITSO_p690
id  Status              partition             assigned_to
22  Configured by System  003*7040-681*021768A  lpar03
23  Configured by System  002*7040-681*021768A  lpar02
3   Configured by System  008*7040-681*021768A  lpar08
2   Configured by System  007*7040-681*021768A  lpar07
1   Configured by System  005*7040-681*021768A  lpar05
16  Configured by System  004*7040-681*021768A  lpar04
17  Configured by System  003*7040-681*021768A  lpar03
5   Configured by System  004*7040-681*021768A  lpar04
0   Configured by System  005*7040-681*021768A  lpar05
20  Configured by System
21  Configured by System
6   Configured by System  006*7040-681*021768A  lpar06
7   Configured by System
18  Configured by System  007*7040-681*021768A  lpar07
19  Configured by System
4   Configured by System  008*7040-681*021768A  lpar08
```

Example 9-7 shows memory information for each running partition. The allocated column shows all the memory allocated to a partition including the page table. The size of the page table is determined by the maximum memory given in the partition profile. For example, partition lpar05 has 2176 MB of memory allocated and a 128 MB page table, which means the available memory size for partition lpar05 is 2176 - 128 = 2048 MB (2 GB).

Example 9-7 lshwres for memory

```
[hscroot@itsohmc]$ lshwres -r mem -m ITSO_p690
allocated  page_table  partition             assigned_to
2112       64          006*7040-681*021768A  lpar06
3584       512         003*7040-681*021768A  lpar03
2112       64          007*7040-681*021768A  lpar07
1088       64          004*7040-681*021768A  lpar04
2176       128         008*7040-681*021768A  lpar08
2176       128         005*7040-681*021768A  lpar05
```

9.2.3 Change system state using chsysstate

This section explains how to use the command line interface to start and stop the managed system and its partitions.

Power on the system

Example 9-8 shows how to use the **chsysstate** command to power on the system in partition standby mode.

Example 9-8 System power on

```
[hscroot@itsohmc]$ chsysstate -o on -r sys -n ITSO_p690 -c lpar
```

Power off the system

Example 9-9 shows how to use the **chsysstate** command to power off the system.

Example 9-9 System power off

```
[hscroot@itsohmc]$ chsysstate -o off -r sys -n ITSO_p690
```

Note: The managed system is powered off even if partitions are still active.

Start a partition

Example 9-10 shows how to use the **chsysstate** command to start a partition using the default profile.

Example 9-10 Start a partition

```
[hscroot@itsohmc]$ chsysstate -m ITSO_p690 -o on -r lpar -n lpar05
```

Reset a partition

There are multiple types of reset available:

- Soft

  AIX will force a kernel dump and reboot. The option is: **`-o reset`**
- Hard

  Power off the partition. The option is: **`-o off`**
- Shut down

  Shut down and power off the partition. The option is: **`-o osshutdown`**
- Shut down and reboot

  Shut down and reboot the partition. The option is: **`-o osreset`**

Example 9-11 shows the options of the **chsysstate** command to reset a partition.

Example 9-11 Reset a partition

```
[hscroot@itsohmc]$ chsysstate -m ITSO_p690 -o reset -r lpar -n lpar05

[hscroot@itsohmc]$ chsysstate -m ITSO_p690 -o off -r lpar -n lpar05

[hscroot@itsohmc]$ chsysstate -m ITSO_p690 -o osshutdown -r lpar -n lpar05

[hscroot@itsohmc]$ chsysstate -m ITSO_p690 -o osreset -r lpar -n lpar05
```

9.2.4 Change hardware resources using chhwres

This section explains how to use the command line interface to modify the hardware resources of a managed system.

The **chhwres** command instructs the managed system to initiate the specified dynamic logical partitioning operation.

The command takes the following options:

- **-r [mem | cpu | slot | led]**

  Specify the resource type to change:

mem	Memory
cpu	CPU
slot	I/O slots
led	LED

- **-o [a | r | m]**

  Specify the operation to be performed:

a	Add resources
r	Remove resources
m	Move resources
s	set LED status

- **-m managed_system**

  The managed system name.

- **-p source_partition_name**

  The source partition name.

- **`-t target_partition_name`**

  The target partition name if the operation is to move resources between partitions.

- **`-i drawer_id`**

  The drawer ID. This option needs to be specified for the I/O slot resource operation only.

- **`-s slot_id`**

  The I/O slot ID. This option needs to be specified for the I/O slot resource operation only.

- **`-q quantity`**

  The quantity of hardware resources to change. For the processor, this value specifies the number of processors to add, remove, or move. For the memory, this value specifies the number of logical memory blocks (LMBs). This option is not used for the I/O slot operation.

- **`-w timeout`**

  The time-out value to be used by the **`drmgr`** command running on the partition. The default value is five minutes.

- **`-d detail_level`**

  The detail level to be used by the **`drmgr`** command running on the partition. Valid values are 0 through 5.

The next section is dedicated to several examples using the **`chhwres`** command, such as CPU and memory operations.

9.3 Dynamic logical partitioning operations using chhwres

Although dynamic logical partitioning (DLPAR) operations can easily be done using the HMC graphical user interface, you can also use the **`chhwres`** command on the HMC to achieve the following two objectives during a DLPAR operation:

- Consistent results
- Automated operations

9.3.1 DLPAR operation to add a processor

Example 9-12 on page 286 shows the output of a two-processor-addition DLPAR operation using the **`chhwres`** command on partition lpar05 in managed system

ITSO_p690. The **lshwres** command outputs show the difference before and after the **chhwres** command execution (the lpar05 partition was allocated two CPUs before the DLPAR operation and four after the operation).

Example 9-12 chhwres CPU addition

```
[hscroot@itsohmc]$ lshwres -m ITSO_p690 -r cpu -p lpar05
allocated  free  max  min   partition            partition_name  assigned_to  system
2          5     6    1     005*7040-681*021768A  lpar05          lpar05       ITSO_p690

[hscroot@itsohmc]$ chhwres -m ITSO_p690 -p lpar05 -r cpu -o a -q 2 -w 4

[hscroot@itsohmc]$ lshwres -m ITSO_p690 -r cpu -p lpar05
allocated  free  max  min   partition            partition_name  assigned_to  system
4          3     6    1     005*7040-681*021768A  lpar05          lpar05       ITSO_p690
```

In Example 9-13 we try to add seven processors. The operation fails because only five processors are available.

Example 9-13 CPU addition failure

```
[hscroot@itsohmc]$ lshwres -m ITSO_p690 -r cpu -p lpar05
allocated  free  max  min   partition            partition_name  assigned_to  system
2          5     6    1     005*7040-681*021768A  lpar05          lpar05       ITSO_p690

[hscroot@itsohmc]$ chhwres -m ITSO_p690 -p lpar05 -r cpu -o a -q 7 -w 4
The quantity to be added exceeds the available resources. There are 5 resources currently
available. Please retry command.
```

9.3.2 DLPAR operation to remove a processor

Example 9-14 shows the output of a processor-removal DLPAR operation using the **chhwres** command. Before removal, partition lpar05 has four processors; three after the operation.

Example 9-14 chhwres CPU removal

```
[hscroot@itsohmc]$ lshwres -m ITSO_p690 -r cpu -p lpar05
allocated  free  max  min   partition            partition_name  assigned_to  system
4          3     6    1     005*7040-681*021768A  lpar05          lpar05       ITSO_p690

[hscroot@itsohmc]$ chhwres -m ITSO_p690 -p lpar05 -r cpu -o r -q 1 -w 4

[hscroot@itsohmc]$ lshwres -m ITSO_p690 -r cpu -p lpar05
allocated  free  max  min   partition            partition_name  assigned_to  system
3          4     6    1     005*7040-681*021768A  lpar05          lpar05       ITSO_p690
```

In Example 9-15 we try to remove five processors from partition lpar05, which has only two. The profile policy prevents removing processors below the minimum number of processors specified in the partition profile.

Example 9-15 CPU removal failure

```
[hscroot@itsohmc]$ lshwres -m ITSO_p690 -r cpu -p lpar05
allocated  free  max  min  partition            partition_name  assigned_to  system
2          5     6    1    005*7040-681*021768A  lpar05          lpar05       ITSO_p690

[hscroot@itsohmc]$ chhwres -m ITSO_p690 -p lpar05 -r cpu -o r -q 5 -w 4
Your processor request goes below the profile's Required processor limit. You can remove or
move 1 or fewer processors. Retry the operation.
```

9.3.3 DLPAR operation to add memory

Example 9-16 shows the output of a 5 GB memory-addition DLPAR operation using the **chhwres** command. Before the operation, partition lpar05 has 8 LMBs allocated, that is to say 2 GB of memory (8 x 256 MB). The difference between before and after the **chhwres** command execution is also shown by the **lshwres** command outputs (the partition lpar05 was allocated 2 GB of memory before the DLPAR operation and 7 GB after).

Example 9-16 chhwres memory addition

```
[hscroot@itsohmc]$ lshwres -m ITSO_p690 -r mem -p lpar05
allocated  free  lmb_size  max  min  partition             system     partition_name
8          202   256       32   8    005*7040-681*021768A  ITSO_p690  lpar05

[hscroot@itsohmc]$ chhwres -m ITSO_p690 -p lpar05 -r mem -o a -q 20 -w 4

[hscroot@itsohmc]$ lshwres -m ITSO_p690 -r mem -p lpar05
allocated  free  lmb_size  max  min  partition             system     partition_name
28         181   256       32   8    005*7040-681*021768A  ITSO_p690  lpar05
```

9.3.4 DLPAR operation to remove memory

Example 9-17 shows the output of a 4 GB memory-removal DLPAR operation using the **chhwres** command. Before removal, partition lpar05 has 7 GB of memory, 3 GB after removal.

Example 9-17 chhwres memory removal

```
[hscroot@itsohmc]$ lshwres -m ITSO_p690 -r mem -p lpar05
allocated  free  lmb_size  max  min  partition             system     partition_name
28         181   256       32   8    005*7040-681*021768A  ITSO_p690  lpar05
```

```
[hscroot@itsohmc]$ chhwres -m ITSO_p690 -p lpar05 -r mem -o r -q 16 -w 4

[hscroot@itsohmc]$ lshwres -m ITSO_p690 -r mem -p lpar05
allocated  free  lmb_size  max  min  partition            system     partition_name
12         196   256       32   8    005*7040-681*021768A  ITSO_p690  lpar05
```

Unfortunately, memory cannot always be removed correctly, as shown in Example 9-19 on page 289. In this example, we try to remove one LMB from partition lpar06, which has only three LMBs allocated, one of which is reserved for the RMO region.

Example 9-20 on page 291 shows syslog output on the lpar06 partition. We can see that **drmgr** tries to remove LMB 0x2 and receives error 27. This means that removing this LMB would not leave enough free memory pages in the system. Then **drmgr** tries to remove LMB 0x1 and receives error 16, which means that this LMB contains non-removable memory. There are no other available LMBs to be freed, so the operation fails.

To understand what happens, we examine the partition, as in Example 9-18.

Example 9-18 Memory information

```
root@lpar06:/ # lsattr -El mem0
goodsize 768 Amount of usable physical memory in Mbytes False
size     768 Total amount of physical memory in Mbytes  False

root@lpar06:/ # svmon -G
               size      inuse       free        pin    virtual
memory       196608     151100      45508     120126     144364
pg space     131072       1405

               work       pers       clnt      lpage
pin           38206          0          0      81920
in use        59132          0      10048          0

             pgsize       size       free
lpage pool    16 MB         20         20

root@lpar06:/ # vmo -o maxpin%
maxpin% = 80
```

The partition has 768 MB of memory. We try to remove 1 LMB (that is, 256 MB), which would leave 512 MB (131072 frames of 4KB), if the removal were successful. The **svmon** command shows that the partition is using 120126 4KB page frames of pinned memory. If we do the ratio of pinned and available memory pages, the result is about 92%, which is greater than the permitted value shown by the **vmo** command (that is, 80%). This explains error 27.

The reason why there are so many pinned pages is that *large memory pages* has been enabled and configured on partition lpar06. The output of the **svmon** command in Example 9-18 on page 288 shows a large page memory pool of 320 MB (16 MB x 20). This explains error 16—an LMB with large pages in it cannot be removed.

Figure 9-1 illustrates what can happen to a memory removal operation.

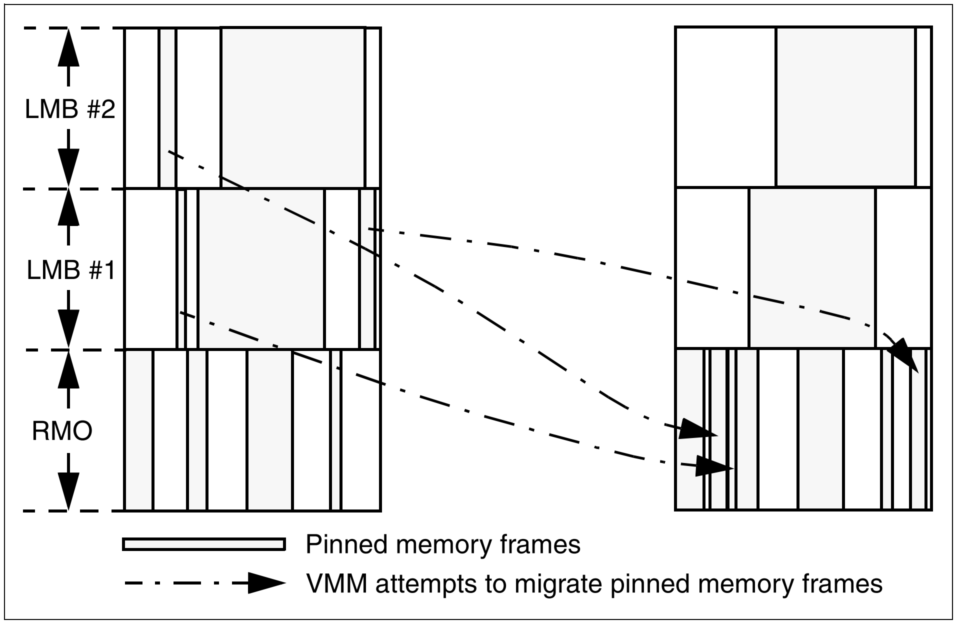


Figure 9-1 1 fixed sized RMO and 2 LMBs are allocated to an AIX 5L Version 5.2 partition

If a DLPAR memory-remove operation is requested, the VMM first finds an LMB, then tries to migrate pinned memory frames in this LMB to the other LMBs, wherever a 256 MB segment can accommodate those pages. However, the migration does not always succeed (for large pages, for example).

Example 9-19 Memory removal failure

```
[hscroot@itsohmc]$ /opt/hsc/bin/chhwres -m ITSO_p690 -p lpar06 -r mem -o r -q 1 -d 5
aixErr: true
HMCERRV3DLPAR020: r operation for mem has completed, but only 0 out of 1 were successful.
The AIX command is:
drmgr -r -c mem -q 1 -w 5 -d 5

The AIX standard output is:
 Checksum for the file is: 0x9552ec7b9269d2ac
Setting LED: 0x2003, MEMR:FW UPD
collect_drc_info:pid for '/' root path:4
```

```
collect_drc_info:id for lpar-capable prop:1
collect_drc_info: total_LMBs:11
collect_drc_info: s_ptr:LMB 1
collect_mem_nodes: collecting all memory nodes
collect_mem_nodes: last_unmarked_node:0x2000fb6820008600
collect_mem_nodes: total_m_nodes:2
remove_mem: total amount:0x10000000
remove_mem: starting script processing CHECK and PRE phases
Setting up scripts library interface for pre ops
Setting LED: 0x2003, MEMR:CHECK
To wait on applications/signals for 10 seconds
Issuing checkrelease command to the scripts
DR Timeout value:300, Current time: 0x3f214da2, DR start time: 0x3f214da2, Remaining Time:300
To wait on applications/signals for 10 seconds
DR Timeout value:300, Current time: 0x3f214da2, DR start time: 0x3f214da2, Remaining Time:300
Setting LED: 0x2003, MEMR:PRE
To wait on applications/signals for 10 seconds
DR Timeout value:300, Current time: 0x3f214da2, DR start time: 0x3f214da2, Remaining Time:300
Issuing prerelease command to the scripts
DR Timeout value:300, Current time: 0x3f214da2, DR start time: 0x3f214da2, Remaining Time:300
To wait on applications/signals for 10 seconds
DR Timeout value:300, Current time: 0x3f214da2, DR start time: 0x3f214da2, Remaining Time:300
remove_mem: last_mem_node=0x2000fb68, last_unmarked_node=0x2000fb38
DR Timeout value:300, Current time: 0x3f214da2, DR start time: 0x3f214da2, Remaining Time:300
Setting LED: 0x2003, MEMR: 1 of 1
remove_mem: kernel update addr:0x20000000       size:0x10000000
                timeout:300 sec         flag:0
remove_mem: dr_resource_identifier:2
remove_mem: kernel rm_mem: Error: notify kernel errno:27
mark_diff_LMB_for_rem: choosing a different LMB to remove...
mark_diff_LMB_for_rem: choosing index: 0x1
DR Timeout value:300, Current time: 0x3f214da2, DR start time: 0x3f214da2, Remaining Time:300
Setting LED: 0x2003, MEMR: 2 of 1
remove_mem: kernel update addr:0x10000000       size:0x10000000
                timeout:300 sec         flag:0
remove_mem: dr_resource_identifier:1
remove_mem: kernel rm_mem: Error: notify kernel errno:16
mark_diff_LMB_for_rem: choosing a different LMB to remove...
remove_mem: invoking convert_and_write to write to iplcb
Setting LED: 0x2003, MEMR:ODM UPD
remove_mem: Successfully configured 'sys0'..
remove_mem: Successfully configured 'mem0'..
update_post_mem_data: after kinfo.req_memsz_change=0x10000000
remove_mem: after s_mem_dets.free_frames=0xc7a1
remove_mem: after s_mem_dets.pinnable_frames=0xb006
remove_mem: after s_mem_dets.total_frames=0x30000
remove_mem: starting script processing POSTERROR phase
Setting LED: 0x2003, MEMR:POST
Setting up scripts library for post ops
```

```
To wait on applications/signals for 10 seconds
To wait on applications/signals for 10 seconds
Issuing undoprerelease command to the scripts
Setting LED: 0x2003, MEMR:FAIL
mem_error_exit: Exiting with msgid:14 msg=''
Setting LED: 0xffff,

The AIX standard error is:

0931-012 Unable to unallocate the resource from the partition.


The return code is 0. The AIX return code is 1.
```

Option **-d 5** is used in Example 9-19 to display the maximum debug information. Since we use a debug level greater than 0, the AIX standard output is redirected to the HMC. That leads to many error messages in the output:

- DRMGR returns two error messages (errno 16 and 27).
- AIX returns error `0931-012 Unable to unallocate the resource from the partition`.
- HMC has a return code 1.

In case of problems during a DLPAR operation, it is a good idea to have a look at the syslog daemon log shown in Example 9-20. It shows the error messages for the LMB removal.

Example 9-20 syslogd daemon log

```
Jul 25 11:56:00 lpar06 DRMGR:  ==== Start: MEM Removal operation ====
Jul 25 11:56:00 lpar06 DRMGR: Starting CHECK phase for mem Remove operation.
Jul 25 11:56:00 lpar06 DRMGR: Phase CHECK started for scripts,kernel extensions and
applications.
Jul 25 11:56:00 lpar06 DRMGR: Starting CHECK phase for Scripts.
Jul 25 11:56:00 lpar06 DRMGR: Completed the phase for Scripts.
Jul 25 11:56:00 lpar06 DRMGR: Starting the phase for kernel extensions.
Jul 25 11:56:00 lpar06 DRMGR: Completed the phase for kernel extensions.
Jul 25 11:56:00 lpar06 DRMGR: Starting the phase for application signal handlers.
Jul 25 11:56:00 lpar06 DRMGR: Completed the phase for kernel extensions.
Jul 25 11:56:00 lpar06 DRMGR: Starting PRE phase.
Jul 25 11:56:00 lpar06 DRMGR: Phase PRE started for scripts,kernel extensions and applications.
Jul 25 11:56:00 lpar06 DRMGR: Starting PRE phase for scripts.
Jul 25 11:56:00 lpar06 DRMGR: Completed the phase for Scripts.
Jul 25 11:56:00 lpar06 DRMGR: Starting the phase for application signal handlers.
Jul 25 11:56:00 lpar06 DRMGR: Completed the phase for kernel extensions.
Jul 25 11:56:00 lpar06 DRMGR: Error: dr_notify to remove LMB:0x2 failed with errno value=27
Jul 25 11:56:00 lpar06 DRMGR: Error: dr_notify to remove LMB:0x1 failed with errno value=16
Jul 25 11:56:00 lpar06 DRMGR: Firmware operations complete
Jul 25 11:56:00 lpar06 DRMGR: ODM operations complete
Jul 25 11:56:00 lpar06 DRMGR: Starting POST phase.
Jul 25 11:56:00 lpar06 DRMGR: Phase POST_ERROR started for scripts,kernel extensions and
applications.
Jul 25 11:56:00 lpar06 DRMGR: Starting the phase for application signal handlers.
Jul 25 11:56:00 lpar06 DRMGR: Completed the phase for kernel extensions.
Jul 25 11:56:00 lpar06 DRMGR: Starting UNDOPRE phase for scripts.
Jul 25 11:56:00 lpar06 DRMGR: Completed the phase for Scripts.
Jul 25 11:56:01 lpar06 DRMGR:  ~~~~ End: DR operation ~~~~
```

9.3.5 DLPAR operation to add an I/O slot

In this section we show how to add an I/O slot with an adapter in it. We first add the resource using the **chhwres** command on the HMC, then we configure the adapter on the partition.

Since many adapters can be added to an I/O drawer, the following section focuses on only two adapter types:

- Network adapter
- Storage adapter

DLPAR operation on a network adapter addition

This section describes how to add a network adapter. For example, we want to add the network adapter in drawer U1.5 slot 6 to partition lpar06.

Operation on HMC

We the **chhwres** command to add the slot needed; **lshwres** shows that the slot is correctly added.

Example 9-21 I/O slot addition

```
[hscroot@itsohmc]$ lshwres -m ITSO_p690 -r slot -p lpar06 -F
phys_loc:slot_id:slot_type
U1.5-P1-I10:10:SCSI bus controller
U1.5-P1-I5:5:Ethernet controller
U1.5-P1/Z2:12:SCSI bus controller

[hscroot@itsohmc]$ chhwres -m ITSO_p690 -p lpar06 -r slot -o a -l U1.5-P1-I6
DrawerId[0]: 7040-61D*02445BA-P1
SlotId[0]: 6
PCIBus[0]: 1

[hscroot@itsohmc]$ lshwres -m ITSO_p690 -r slot -p lpar06 -F
phys_loc:slot_id:slot_type
U1.5-P1-I10:10:SCSI bus controller
```

```
U1.5-P1-I5:5:Ethernet controller
U1.5-P1-I6:6:Ethernet controller
U1.5-P1/Z2:12:SCSI bus controller
```

Operation on partition

Run the **cfgmgr** command to instruct AIX to configure the newly added hardware resources and verify with the **lsdev** command. A new Ethernet adapter, ent1, is now available and can be configured using the **ifconfig** command. The **netstat** command shows the new en1 Ethernet interface with address 10.0.0.11.

Example 9-22 New network adapter

```
root@lpar06:/ # lsdev -Cc adapter
ent0  Available 4F-08 10/100 Mbps Ethernet PCI Adapter II (1410ff01)
sa0   Available       LPAR Virtual Serial Adapter
scsi0 Available 4b-08 Wide/Ultra-3 SCSI I/O Controller
scsi1 Available 4e-08 Wide/Ultra-3 SCSI I/O Controller
scsi2 Available 4e-09 Wide/Ultra-3 SCSI I/O Controller

root@lpar06:/ # cfgmgr

root@lpar06:/ # lsdev -Cc adapter
ent0  Available 4F-08 10/100 Mbps Ethernet PCI Adapter II (1410ff01)
ent1  Available 4J-08 10/100 Mbps Ethernet PCI Adapter II (1410ff01)
sa0   Available       LPAR Virtual Serial Adapter
scsi0 Available 4b-08 Wide/Ultra-3 SCSI I/O Controller
scsi1 Available 4e-08 Wide/Ultra-3 SCSI I/O Controller
scsi2 Available 4e-09 Wide/Ultra-3 SCSI I/O Controller

root@lpar06:/ # ifconfig en1 inet 10.0.0.11 netmask 255.255.255.192

root@lpar06:/ # netstat -in
Name  Mtu   Network     Address            Ipkts Ierrs    Opkts Oerrs  Coll
en0   1500  link#2      0.2.55.af.17.e8    2381877     0    69774     0     0
en0   1500  9.3.4       9.3.4.70           2381877     0    69774     0     0
en1   1500  link#3      0.2.55.af.19.5           3     0        0     0     0
en1   1500  10.0.0      10.0.0.11                3     0        0     0     0
lo0   16896 link#1                           31949     0    32017     0     0
lo0   16896 127         127.0.0.1            31949     0    32017     0     0
lo0   16896 ::1                              31949     0    32017     0     0
```

DLPAR operation on a storage adapter addition

This section describes how to add a storage adapter. For example, we want to add the SSA adapter in drawer U1.5 slot 7 to partition lpar06.

Operation on HMC

The only thing to do on the HMC is to use the **chhwres** command to add the slot needed; **lshwres** shows that the slot is correctly added.

Example 9-23 I/O slot addition

```
[hscroot@itsohmc hscroot]$ lshwres  -m ITSO_p690 -r slot -p lpar06 -F
phys_loc:slot_id:slot_type
U1.5-P1-I10:10:SCSI bus controller
U1.5-P1-I5:5:Ethernet controller
U1.5-P1-I6:6:Ethernet controller
U1.5-P1/Z2:12:SCSI bus controller

[hscroot@itsohmc hscroot]$ chhwres -m ITSO_p690 -p lpar06 -r slot -o a -l
U1.5-P1-I7 -d 5
DrawerId[0]: 7040-61D*02445BA-P1
SlotId[0]: 7
PCIBus[0]: 1

[hscroot@itsohmc hscroot]$ lshwres  -m ITSO_p690 -r slot -p lpar06 -F
phys_loc:slot_id:slot_type
U1.5-P1-I10:10:SCSI bus controller
U1.5-P1-I5:5:Ethernet controller
U1.5-P1-I6:6:Ethernet controller
U1.5-P1-I7:7:SSA Serial Bus
U1.5-P1/Z2:12:SCSI bus controller
```

Operation on partition

Run the **cfgmgr** command to instruct AIX to configure the newly added hardware resources. In Example 9-24, the SSA adapter we added to lpar06 is configured as ssa0, and its children disk drive devices (physical volumes) are configured as hdisk 2, 3, 4, and 5.

Example 9-24 New SSA adapter

```
root@lpar06:/ # lsdev -Cc adapter
ent0  Available 4F-08 10/100 Mbps Ethernet PCI Adapter II (1410ff01)
ent1  Available 4J-08 10/100 Mbps Ethernet PCI Adapter II (1410ff01)
sa0   Available       LPAR Virtual Serial Adapter
scsi0 Available 4b-08 Wide/Ultra-3 SCSI I/O Controller
scsi1 Available 4e-08 Wide/Ultra-3 SCSI I/O Controller
scsi2 Available 4e-09 Wide/Ultra-3 SCSI I/O Controller

root@lpar06:/ # lsdev -Ccdisk
hdisk0 Available 4b-08-00-8,0  16 Bit LVD SCSI Disk Drive
hdisk1 Available 4b-08-00-10,0 16 Bit LVD SCSI Disk Drive

root@lpar06:/ # cfgmgr
```

```
root@lpar06:/ # lsdev -Cc adapter
ent0  Available 4F-08 10/100 Mbps Ethernet PCI Adapter II (1410ff01)
ent1  Available 4J-08 10/100 Mbps Ethernet PCI Adapter II (1410ff01)
sa0   Available       LPAR Virtual Serial Adapter
scsi0 Available 4b-08 Wide/Ultra-3 SCSI I/O Controller
scsi1 Available 4e-08 Wide/Ultra-3 SCSI I/O Controller
scsi2 Available 4e-09 Wide/Ultra-3 SCSI I/O Controller
ssa0  Available 4Q-08 IBM SSA 160 SerialRAID Adapter (14109100)

root@lpar06:/ # lsdev -Ccdisk
hdisk0 Available 4b-08-00-8,0  16 Bit LVD SCSI Disk Drive
hdisk1 Available 4b-08-00-10,0 16 Bit LVD SCSI Disk Drive
hdisk2 Available 4Q-08-L       SSA Logical Disk Drive
hdisk3 Available 4Q-08-L       SSA Logical Disk Drive
hdisk4 Available 4Q-08-L       SSA Logical Disk Drive
hdisk5 Available 4Q-08-L       SSA Logical Disk Drive
```

9.3.6 DLPAR operation to remove an I/O slot

In this section we show how to remove an I/O slot with an adapter in it. We first remove its definition in the partition and then remove the resource using the **chhwres** command on the HMC.

Since many adapters can be added to an I/O drawer, the following section focuses on only two adapter types:

- Network adapter
- Storage adapter

DLPAR operation on removal of a network adapter

The following steps describe how to remove a network adapter from a partition:

1. Identify the adapter to remove.
2. Un-configure the adapter.
3. Remove the AIX definition of the adapter.
4. Remove the adapter from the partition using the HMC.

Operation on partition

In Example 9-25 on page 296, we want to remove the network adapter with the following IP address: 10.0.0.10. In order to do that, we follow these steps:

- Identify the adapter.

  First we have to identify the network adapter name to remove with the **netstat** command, then find the PCI slot in witch the adapter is connected using the **lsdev** command, and finally find the location code of the slot using the **lsslot** command.

- Un-configure the adapter.

  We down and detach the interface using the **ifconfig** command.

- Remove the AIX definition of the adapter.

  We un-configure the adapter and remove the adapter definition and any of its children with the **rmdev** command.

Example 9-25 Remove a network adapter

```
root@lpar06:/ # netstat -in
Name  Mtu   Network     Address            Ipkts Ierrs    Opkts Oerrs  Coll
en0   1500  link#2      0.2.55.af.17.e8   250769     0   114379     0     0
en0   1500  9.3.4       9.3.4.70          250769     0   114379     0     0
en1   1500  link#3      0.2.55.af.19.5       304     0        4     0     0
en1   1500  10          10.0.0.10            304     0        4     0     0
lo0   16896 link#1                         59357     0    59462     0     0
lo0   16896 127         127.0.0.1          59357     0    59462     0     0
lo0   16896 ::1                            59357     0    59462     0     0

root@lpar06:/ # lsdev -Cl ent1 -F parent
pci14

root@lpar06:/ # lsslot -c slot -l pci14
# Slot      Description  Device(s)
U1.5-P1-I6  DLPAR slot   pci14 ent1

root@lpar06:/ # ifconfig en1 down

root@lpar06:/ # ifconfig en1 detach

root@lpar06:/ # rmdev -dRl pci14
ent1 deleted
pci14 deleted
```

Operation on HMC

Now that the adapter definition has been removed from the partition, we can remove the adapter from the partition using the **chhwres** command, as shown in Example 9-26.

Example 9-26 Adapter removal on HMC

```
[hscroot@itsohmc]$ chhwres -m ITSO_p690 -p lpar06 -r slot -o r -l U1.5-P1-I6
DrawerId[0]: 7040-61D*02445BA-P1
SlotId[0]: 6
PCIBus[0]: 1
```

```
[hscroot@itsohmc]$ lshwres -m ITSO_p690 -r slot -p lpar06 -F
phys_loc:slot_id:slot_type
U1.5-P1-I10:10:SCSI bus controller
U1.5-P1-I5:5:Ethernet controller
U1.5-P1/Z2:12:SCSI bus controller
```

DLPAR operation on the removal of a storage adapter

The following steps describe how to remove a storage adapter from a partition:

1. Identify the adapter to remove.
2. Identify resources related to the adapter.
3. Remove file systems and volume groups.
4. Remove physical volumes.
5. Remove the AIX definition of the adapter.
6. Remove the adapter from the partition using the HMC.

Operation on partition

In Example 9-27 on page 298, we want to remove the storage adapter ssa0. In order to do that, we follow these steps:

- Identify the adapter.

  First we have to identify the storage adapter to remove with the **lsdev** command, then find the PCI slot in which the adapter is connected using the **lsdev** command, and finally find the location code of the slot using the **lsslot** command.

- Identify the resources connected to this adapter.

  The **lsdev -Ccdisk** command shows each disk address, in this case hdisk2 to hdisk7, is connected to the ssa0 adapter (address 4Q-08). To free those disks, we have to know if there is some data on it. The **lspv** command shows that two disks are part of a volume group named testvg. Using **lsvg -l testvg**, we see that two file systems exist in this volume group.

- Remove resources related to the adapter.

  We have all the information needed to free all the resources that belong to the adapter in the following two steps:

  – Free volume groups.

    Unmount all the file systems using the **umount** command.

  – Free disks.

    Remove the volume group definition using the **exportvg** command. This command does not destroy any data on the disk, so it can be used again for another partition if needed.

- Remove the AIX definition of the adapter.

  We un-configure and un-define the device and any of its children with the **rmdev** command. In the SSA device structure, each hdisk is associated with a pdisk. We use the **ssaxlate** command to do the translation. We first have to remove all pdisks and hdisks, then the pci15 adapter.

Example 9-27 Remove a storage adapter

```
root@lpar06:/ # lsdev -Cc adapter
ent0  Available 4F-08 10/100 Mbps Ethernet PCI Adapter II (1410ff01)
sa0   Available       LPAR Virtual Serial Adapter
scsi0 Available 4b-08 Wide/Ultra-3 SCSI I/O Controller
scsi1 Available 4e-08 Wide/Ultra-3 SCSI I/O Controller
scsi2 Available 4e-09 Wide/Ultra-3 SCSI I/O Controller
ssa0  Available 4Q-08 IBM SSA 160 SerialRAID Adapter (14109100)

root@lpar06:/ # lsdev -Cl ssa0 -F parent
pci15

root@lpar06:/ # lsslot -c slot -l pci15
# Slot      Description  Device(s)
U1.5-P1-I7  DLPAR slot   pci15 ssa0

root@lpar06:/ # lsdev -Ccdisk
hdisk0 Available 4b-08-00-8,0  16 Bit LVD SCSI Disk Drive
hdisk1 Available 4b-08-00-10,0 16 Bit LVD SCSI Disk Drive
hdisk2 Available 4Q-08-L       SSA Logical Disk Drive
hdisk3 Available 4Q-08-L       SSA Logical Disk Drive
hdisk4 Available 4Q-08-L       SSA Logical Disk Drive

root@lpar06:/ # ssaxlate -l hdisk2
pdisk0
root@lpar06:/ # ssaxlate -l hdisk3
pdisk1
root@lpar06:/ # ssaxlate -l hdisk4
pdisk2

root@lpar06:/ # lspv
hdisk0         0021768a3b1655ee                    rootvg          active
hdisk1         0021768a21525a2a                    None
hdisk2         00050592247553da                    testvg          active
hdisk3         00000000035d72e7                    testvg          active
hdisk4         0021768a9378cb88                    None
```

```
root@lpar06:/ # lsvg -l testvg
testvg:
LV NAME             TYPE       LPs   PPs   PVs  LV STATE      MOUNT POINT
fslv01              jfs2       64    64    1    open/syncd    /tst
loglv02             jfslog     1     1     1    open/syncd    N/A
lv06                jfs        64    64    1    open/syncd    /work

root@lpar06:/ # umount /tst
root@lpar06:/ # umount /work

root@lpar06:/ # varyoffvg testvg
root@lpar06:/ # exportvg testvg

root@lpar06:/ # rmdev -dl pdisk0
pdisk0 deleted
root@lpar06:/ # rmdev -dl hdisk2
hdisk2 deleted
root@lpar06:/ # rmdev -dl pdisk1
pdisk1 deleted
root@lpar06:/ # rmdev -dl hdisk3
hdisk3 deleted
root@lpar06:/ # rmdev -dl pdisk2
pdisk2 deleted
root@lpar06:/ # rmdev -dl hdisk4
hdisk4 deleted

root@lpar06:/ # rmdev -dRl pci15
ssa0 deleted
pci15 deleted
```

Operation on HMC

After the AIX adapter definition is removed from the partition, we can remove the adapter definition from the partition using the **chhwres** command and verify that lpar06 has no more adapters with location code U1.5-P1-I7, as shown in Example 9-28.

Example 9-28 Adapter removal on HMC

```
[hscroot@itsohmc hscroot]$ chhwres -m ITSO_p690 -p lpar06 -r slot -o r -l
U1.5-P1-I7 -d 5
DrawerId[0]: 7040-61D*02445BA-P1
SlotId[0]: 7
PCIBus[0]: 1

[hscroot@itsohmc hscroot]$ lshwres  -m ITSO_p690 -r slot -p lpar06 -F
phys_loc:slot_id:slot_type
U1.5-P1-I10:10:SCSI bus controller
```

```
U1.5-P1-I5:5:Ethernet controller
U1.5-P1-I6:6:Ethernet controller
U1.5-P1/Z2:12:SCSI bus controller
```

9.3.7 DLPAR operation to move a CD/DVD device

A managed system is equipped with at least one CD/DVD device (CD/DVD here refers to any CD-ROM, DVD-ROM, or DVD-RAM device). This device is used for all the partitions. Therefore, we need to be able to move it from one partition to another easily. In this section we create a script to move the CD/DVD using DLPAR operations. This script is executed on a third machine, which should have remote connections enabled to all the partitions and should be able to connect to the HMC using SSH.

The following steps describe how to move the CD/DVD from partition A to partition B:

1. Get the managed system name.
2. Get the partition name to which the CD/DVD is assigned.
3. Get the CD/DVD device name on partition A.
4. Unmount the CD/DVD.
5. Get the SCSI adapter name.
6. Get the PCI slot.
7. Remove the CD/DVD definitions.
8. Remove the CD/DVD from partition A.
9. Add the CD/DVD to partition B.
10. Configure the new CD/DVD on partition B.

In order to move the CD/DVD, we create the shell script shown in Example 9-29, called MoveCD. It has one option: the name of the partition to which we want to assign the CD/DVD. It finds whether the CD/DVD is allocated to a partition or not. Then, if necessary, it removes all resources related to the CD/DVD and moves the CD/DVD using DLPAR operations. We use **rsh** to issue commands from the third machine to partitions. In order to use another connection method, you just need to modify the RSHCmd variable.

Example 9-29 MoveCD shell script

```
#!/bin/ksh
# This script moves the CD/DVD from the partition which using it to the
# partition given in parameter
```

```
if [[ $# != 1 ]]
then
  print "Syntax : $0 partition_name"
  exit 1
fi

TargetPartitionName=$1          # the partition to move the CD to

User="hscroot"                  # the user to connect to HMC
HMC="itsohmc"                   # the HMC name
SSHCmd="ssh $User@$HMC"         # the SSH command to connect to HMC
CmdPath="/opt/hsc/bin"          # the path for the command on HMC
RSHCmd="rsh"                    # the remote command to connect to the lpar
CDROMPhysLoc="U1.9-P1-I10"      # Default CD-ROM physical location

# Get the first manage system name
ManagedSystem=$($SSHCmd $CmdPath/lssyscfg -r sys --all -Fname | head -1)

# Get the partition name to which the CD-ROM is assigned to
Partition=$($SSHCmd $CmdPath/lshwres -m $ManagedSystem -r slot -Fphys_loc:assigned_to | grep
$CDROMPhysLoc | cut -f2 -d ":")

# check if source and target are the same
if [[ $TargetPartitionName = $Partition ]]
then
  print "CD-ROM is already on $Partition"
  exit
fi

if [[ $Partition = "null" ]]
then
  # CD-ROM is not allacted
  print "CD-ROM is not allacted"
  print "Adding CD-ROM to $TargetPartitionName"
  $SSHCmd $CmdPath/chhwres -m $ManagedSystem -r slot -p $TargetPartitionName -o a -l
    $CDROMPhysLoc
  $RSHCmd $TargetPartitionName cfgmgr
else
  # CD-ROM is allocated to a partition
  print "CD-ROM is on $Partition"

  # returns the only CD-ROM connected to "U1.9-P1-I10"
  CDROM=$($RSHCmd $Partition lscfg | grep "cd.*$CDROMPhysLoc" | cut -f2 -d " ")
  if (( $? != 0 ))
  then
    print "Cannot connect to partition $Partition"
    exit 1
  fi
  if [[ ! -z $CDROM ]]
  then
    # Check if CDROM is mounted
    MountPoint=$($RSHCmd $Partition mount | grep $CDROM | awk '{print $2}')
    if [[ ! -z $MountPoint ]]
    then
      # Unmount CD-ROM
      $RSHCmd $Partition umount $MountPoint

      # Check if CD-ROM has been correctly unmounted
      MountPoint=$($RSHCmd $Partition mount | grep $CDROM | awk '{print $2}')
      if [[ ! -z $MountPoint ]]
      then
        print "CD-ROM is busy, can't unmount $MountPoint "
        exit 1
      fi
    fi

    SCSIadapt=$($RSHCmd $Partition lsdev -Cl $CDROM -F parent)
    # retunrs the PCI slot related to the CD-ROM
    PCISlot=$($RSHCmd $Partition lsdev -Cl $SCSIadapt -F parent)
    $RSHCmd $Partition rmdev -dRl $PCISlot > /dev/null
    PCISlot=$($RSHCmd $Partition lsdev -Cl $SCSIadapt -F parent)
    if [[ ! -z $PCISlot ]]
    then
      print "PCI slot $PCISlot is busy"
      exit 1
    fi
  fi
  print "Removing CD-ROM from $Partition"
  $SSHCmd $CmdPath/chhwres -m $ManagedSystem -r slot -p $Partition -o r -l $CDROMPhysLoc
  print "Adding CD-ROM to $TargetPartitionName"
  $SSHCmd $CmdPath/chhwres -m $ManagedSystem -r slot -p $TargetPartitionName -o a -l
    $CDROMPhysLoc
  $RSHCmd $TargetPartitionName cfgmgr
fi

exit 0
```

9.4 Dynamic logical partitioning resources reassignment scheduling

In this section we show how to schedule resource reassignments using DLPAR operations.

In some cases, the workload can change during the day. A partition may need more resources for some hours. Let's consider an example with two partitions running two applications, both partitions having the same resources during the day. Partition A needs more resources during the night and partition B does not use those resources during the night. It could be very useful to move those resources using DLPAR operations at a defined time of the day.

We use the **`chhwres`** command to reassign the resources between the two partitions. We move a defined quantity of processors and memory from partition A to partition B for a part of the day, and then we move the resources back to partition A for the rest of the day.

A scheduler triggers the resource reassignment twice a day.

Figure 9-2 on page 304 shows the relationships between the machines. The scheduler issues commands to the HMC using SSH. The HMC issues DLPAR operations to the partitions. The resources are moved between partition A and partition B.

In this section, to give some simple examples. We use an ssh key generated with a null passphase. For more information on the remote connection between the scheduler and the HMC, refer to 9.1.2, "Connection to the HMC for automated operations" on page 279.

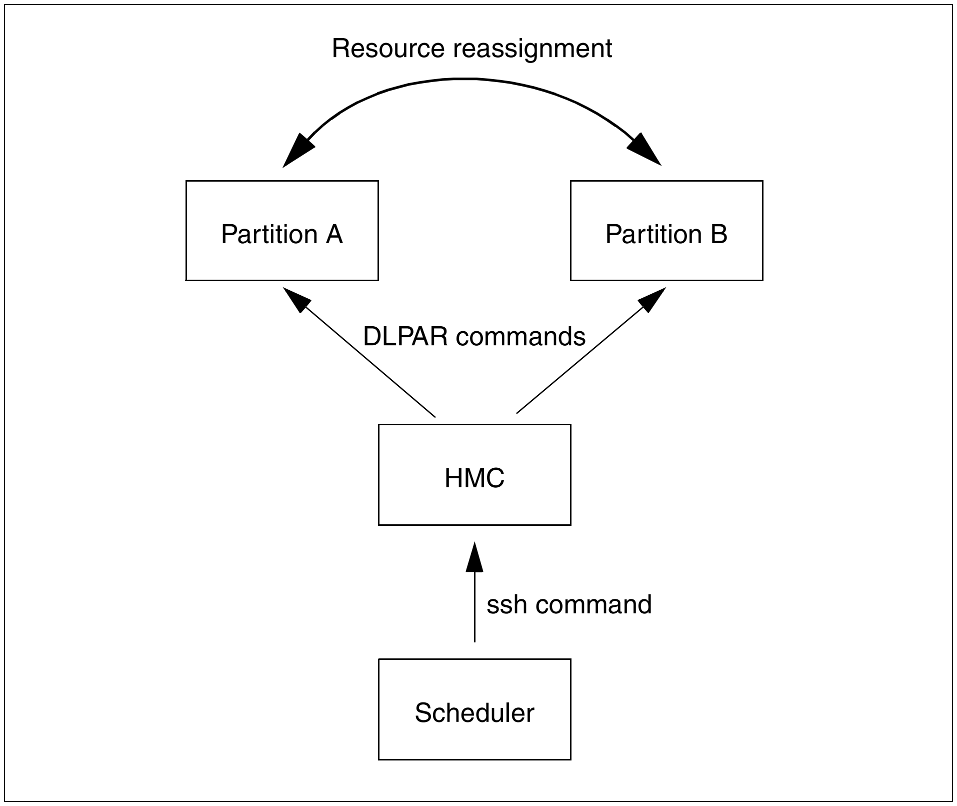


Figure 9-2 Working environment

9.4.1 Partition configuration

We use partition lpar05, which has seven processors and 20 GB of memory, and partition lpar06, which has 1 processor and 4 GB of memory. Example 9-30 shows the profile of each partition.

Example 9-30 CPU and memory for each partition

```
[hscroot@itsohmc]$ lssyscfg -r prof -m ITSO_p690 -p lpar05 -n AIX52
Name   BootMode  DesiredCPU  DesiredMEM  MaxCPU  MaxMEM  MinCPU  MinMEM
AIX52  1         7           20480       8       22528   1       4096

[hscroot@itsohmc]$ lssyscfg -r prof -m ITSO_p690 -p lpar06 -n AIX52
Name   BootMode  DesiredCPU  DesiredMEM  MaxCPU  MaxMEM  MinCPU  MinMEM
AIX52  1         1           4096        8       22528   1       4096
```

9.4.2 Script example to move the resources

In this paragraph we give a simple Korn shell script for the resource reassignment. There is no checking for the available resources. The profile of each partition provides the necessary limits.

The script has only one option, with two values:

- `day` - This moves the resources from partition lpar06 to partition lpar05.
- `night` - This moves the resources from partition lpar05 to partition lpar06.

The script exits with return code 0 if every reassignment has been successful.

Example 9-31 shows the source code of the script MoveRes. It first checks the syntax, then selects the source and target partitions, defines every variable needed, and moves the memory and CPU resources.

Example 9-31 ksh reassignment script (MoveRes)

```
#!/bin/ksh
# This script moves resources between two partitions

print "Starting at : $(date)"

# Verify the syntax, only one option
if [[ $# != 1 ]]
then
  print "Syntax : $0 day|night"
  exit 1
fi

Mode=$1

# Select source and target partition depending on the mode
case $Mode in
  day)
    SourcePartitionName="lpar06"
    TargetPartitionName="lpar05" ;;
  night)
    SourcePartitionName="lpar05"
    TargetPartitionName="lpar06" ;;
  *)
print "Syntax : $0 day|night"
    exit 1 ;;
esac

memToMove=64                 # the number of LMB to move
cpuToMove=6                  # the number of CPU to move
ManagedSystem="ITSO_p690"    # the managed system name
```

```
User="hscroot"                  # the user to connect to HMC
HMC="itsohmc"                   # the HMC name
SSHCmd="ssh $User@$HMC"         # the SSH command to connect to HMC
CmdPath="/opt/hsc/bin"          # the path for the command on HMC
(( RC=0 ))

# Move each resources for one partition to another
for resource in mem cpu
do
  (( QtyToMove=$resource"ToMove" ))
  print "Moving $resource ($QtyToMove) from $SourcePartitionName to
$TargetPartitionName"
  $SSHCmd $CmdPath/chhwres -m $ManagedSystem -r $resource -p
$SourcePartitionName -o m -q $QtyToMove -t $TargetPartitionName -d 5
  (( RC=RC + $? ))
done

print "Return code is : $RC"
print "============================================="

exit $RC
```

In order to make that script more generic, we may replace some variables by parameters—for example, the name of partitions and the quantity of resources to move. For simplicity reasons, we use the script as it is in the following sections.

9.4.3 Reassignment tests using the script

We move six processors and 16 GB of memory from lpar05 to lpar06 for the night, as shown in Example 9-32.

Example 9-32 Resources reassignment for the night

```
user@host1:/ # MoveRes night
Starting at : Fri Aug  1 19:21:03 CDT 2003
Moving resources for the night
Moving mem (64) from lpar05 to lpar06
Moving cpu (6) from lpar05 to lpar06
Return code is : 0
```

Example 9-33 shows the resources after the reassignment for the night; partition lpar06 has seven processors and 20 GB of memory.

Example 9-33 Night resource list for each partition

```
[hscroot@itsohmc]$ lshwres -m ITSO_p690 -r mem | grep lpar0[56]
20992      512        006*7040-681*021768A  lpar06
4608       512        005*7040-681*021768A  lpar05
```

```
[hscroot@itsohmc]$ lshwres -m ITSO_p690 -r cpu | grep lpar0[56]
23  Configured by System  006*7040-681*021768A  lpar06
1   Configured by System  005*7040-681*021768A  lpar05
0   Configured by System  006*7040-681*021768A  lpar06
20  Configured by System  006*7040-681*021768A  lpar06
21  Configured by System  006*7040-681*021768A  lpar06
6   Configured by System  006*7040-681*021768A  lpar06
19  Configured by System  006*7040-681*021768A  lpar06
4   Configured by System  006*7040-681*021768A  lpar06
```

Example 9-34 shows the resources from the partition point of view. This output confirms the HMC output.

Example 9-34 Resources in partition lpar06

```
root@lpar05:/ # lsattr -El mem0
goodsize 20480 Amount of usable physical memory in Mbytes False
size     20480 Total amount of physical memory in Mbytes  False

root@lpar06:/ # lsdev -Cc processor
proc0  Available 00-00 Processor
proc4  Available 00-04 Processor
proc19 Available 00-19 Processor
proc20 Available 00-20 Processor
proc21 Available 00-21 Processor
proc23 Available 00-23 Processor
proc6  Available 00-06 Processor
```

We move the resources back to partition lpar05 for the day in Example 9-35

Example 9-35 Resources reassignment for the day

```
root@lpar05:/ # MoveRes day
Starting at : Fri Aug  2 08:16:00 CDT 2003
Moving resources for the day
Moving mem (64) from lpar06 to lpar05
Moving cpu (6) from lpar06 to lpar05
Return code is : 0
```

Example 9-36 shows the resources after the reassignment for the day; partition lpar06 now has one processor and 4 GB of memory.

Example 9-36 Day resource list for each partition

```
[hscroot@itsohmc]$ lshwres -m ITSO_p690 -r mem | grep lpar0[56]
4608      512        006*7040-681*021768A  lpar06
20992     512        005*7040-681*021768A  lpar05
```

```
[hscroot@itsohmc]$ lshwres -m ITSO_p690 -r cpu | grep lpar0[56]
23  Configured by System  005*7040-681*021768A  lpar05
1   Configured by System  005*7040-681*021768A  lpar05
0   Configured by System  005*7040-681*021768A  lpar05
20  Configured by System  005*7040-681*021768A  lpar05
21  Configured by System  005*7040-681*021768A  lpar05
6   Configured by System  005*7040-681*021768A  lpar05
19  Configured by System  005*7040-681*021768A  lpar05
4   Configured by System  006*7040-681*021768A  lpar06
```

9.4.4 Scheduling example to move the resources using cron

In this section we use the cron daemon as scheduler to activate the resource reassignments. We use the script created in 9.4.2, "Script example to move the resources" on page 305.

Example 9-37 shows the crontab file. The MoveRes script is executed twice a day, first at 21h30 to move resources for the night, and again at 5h45 to move the resources for the day.

The script output is redirected to the log file /home/guest/MoveRes.log.

Example 9-37 crontab

```
# resources reassignment test
30 21 * * * /home/guest/MoveRes night >> /home/guest/MoveRes.log 2>&1
45 05 * * * /home/guest/MoveRes day >> /home/guest/MoveRes.log 2>&1
```

Example 9-38 shows the content of the log file MoveRes.log. Every night at 21h30 the resources are reassigned for the night (from lpar05 to lpar06), and every morning at 5h45 the resources are reassigned for the day (back to lpar05).

Example 9-38 Log file output

```
Starting at : Sat Aug  2 21:30:00 CDT 2003
Moving resources for the night
Moving mem (8) from lpar05 to lpar06
Moving cpu (3) from lpar05 to lpar06
Return code is : 0
==============================================
Starting at : Sun Aug  3 05:45:00 CDT 2003
Moving resources for the day
Moving mem (8) from lpar06 to lpar05
Moving cpu (3) from lpar06 to lpar05
Return code is : 0
==============================================
Starting at : Sun Aug  3 21:30:00 CDT 2003
Moving resources for the night
Moving mem (8) from lpar05 to lpar06
Moving cpu (3) from lpar05 to lpar06
Return code is : 0
==============================================
Starting at : Mon Aug  4 05:45:00 CDT 2003
Moving resources for the day
Moving mem (8) from lpar06 to lpar05
Moving cpu (3) from lpar06 to lpar05
Return code is : 0
==============================================
```

9.4.5 Scheduling example to move resources using IBM Tivoli Workload Scheduler

In this section we use IBM Tivoli Workload Scheduler to activate the resource reassignment. We define jobs and job streams, and update the production plan. We assume there is already a configured IBM Tivoli Workload Scheduler master.

IBM Tivoli Workload Scheduler helps you to automate activities by planning every phase of production, resolving dependencies, and controlling the execution of every job. A production plan is created every day using objects, schedules, and dependencies (jobs or job streams). IBM Tivoli Workload Scheduler launches the production plan by default at 6:00 a.m.

For further information about IBM Tivoli Workload Scheduler, refer to *What is New with IBM Tivoli Workload Scheduler 8.2?*, SG24-6628.

Job definitions

A job is a script or command run and controlled by IBM Tivoli Workload Scheduler.

We have to create two jobs, one for the day (DLPARDAY) and one for the night (DLPARNIGHT). Example 9-39 shows the file used to define IBM Tivoli Workload Scheduler jobs. In this example, the script **/home/guest/MoveRes** shown in Example 9-31 on page 305 is launched by user IBM Tivoli Workload Scheduler.

Example 9-39 Jobs file example (DLPARJobs.txt)

```
$JOBS
MASTER#DLPARNIGHT SCRIPTNAME "/home/guest/MoveRes night >>
/home/guest/MoveRes.log"
  STREAMLOGON tws
  DESCRIPTION "DLPAR operation for night"
  RECOVERY STOP

MASTER#DLPARDAY SCRIPTNAME "/home/guest/MoveRes day >> /home/guest/MoveRes.log"
  STREAMLOGON tws
  DESCRIPTION "DLPAR operation for day"
  RECOVERY STOP
```

Job stream definitions

A job stream is an object that groups a list of jobs for the same function or application on a defined date and time. Those jobs may have relationships.

In our example, we create one job for one job stream, but you can add more jobs in the same job stream. For example, you can add a job that starts an application to the job that makes the resources for that job available.

We create two job streams, one for the day beginning at 6h30, starting job DLPARDAY, and another beginning at 21h45, starting the job DLPARNIGHT. Example 9-40 shows the text file used to define IBM Tivoli Workload Scheduler job streams.

Example 9-40 Job streams file example (DLPARJobStreams.txt)

```
SCHEDULE DLDAY ON EVERYDAY
         AT 0630
:
  DLPARDAY
END

SCHEDULE DLNIGHT ON EVERYDAY
         AT 2145
:
  DLPARNIGHT
END
```

Job and job stream creation

The following steps describe how to create jobs and job streams, shown in Example 9-41 on page 311:

1. Create jobs.

   We first add the new jobs with the **`composer add DLPARJobs.txt`** command. A job list beginning with DL, located on the master server, is displayed with the **`composer display job=MASTER#DL@`** command.

2. Create job streams.

We add the new job stream with the **`composer add DLPARSJobStreams.txt`** command. A job stream list beginning with DL, located on the master server, is displayed with **`composer display sched=MASTER#DL@`**.

Example 9-41 New job and job stream addition

```
$ composer add DLPARJobs.txt
TWS for UNIX (AIX)/COMPOSER 8.2 (1.18.2.1)
AWSBIA013I Job MASTER#DLPARNIGHT added.
AWSBIA013I Job MASTER#DLPARDAY added.
AWSBIA090E Total errors in DLPARJobs.txt: 0, warnings 0.

$ composer display job=MASTER#DL@
TWS for UNIX (AIX)/COMPOSER 8.2 (1.18.2.1)
CPU id.          Job                                     logon    LastRunDate
---------------- ---------------------------------------- -------- -----------
MASTER           DLPARDAY                                tws
MASTER#DLPARDAY SCRIPTNAME "/home/guest/MoveRes day >> /home/guest/MoveRes.log"
         STREAMLOGON "tws"
         DESCRIPTION "DLPAR operation for day"
         RECOVERY STOP

CPU id.          Job                                     logon    LastRunDate
---------------- ---------------------------------------- -------- -----------
MASTER           DLPARNIGHT                              tws
MASTER#DLPARNIGHT SCRIPTNAME "/home/guest/MoveRes night >>
/home/guest/MoveRes.log"
         STREAMLOGON "tws"
         DESCRIPTION "DLPAR operation for night"
         RECOVERY STOP
AWSBIA202I Found 2 jobs for MASTER#DL@

$ composer add DLPARSJobStreams.txt
TWS for UNIX (AIX)/COMPOSER 8.2 (1.18.2.1)
AWSBIA019E For MASTER#DLDAY Errors 0, warnings 0.
AWSBIA015I Schedule MASTER#DLDAY added.
AWSBIA019E For MASTER#DLNIGHT Errors 0, warnings 0.
AWSBIA015I Schedule MASTER#DLNIGHT added.
AWSBIA090E Total errors in /usr/TWS/tmp/TWSkj497a: 0, warnings 0.

$ composer display sched=MASTER#DL@
TWS for UNIX (AIX)/COMPOSER 8.2 (1.18.2.1)
CPU id.          Schedule          Creator                    Last Updated
----------------  ----------------  --------------------------  ------------
MASTER           DLDAY             tws                          08/05/03
SCHEDULE MASTER#DLDAY ON EVERYDAY
         AT 0630
:
  DLPARDAY
```

```
END

CPU id.          Schedule          Creator                     Last Updated
---------------- ----------------  --------------------------   ------------
MASTER           DLNIGHT           tws                           08/05/03

SCHEDULE MASTER#DLNIGHT ON EVERYDAY
         AT 2145
:
  DLPARNIGHT
END
AWSBIA035I Found 2 schedules in MASTER#DL@
```

Production plan update

The new job streams have to be added now to the next production plan. This is done with the **`Jnextday`** command, as shown in Example 9-42. To verify that the jobs are correctly scheduled, use **`conman sj`**.

Example 9-42 Next production plan creation

```
$ Jnextday
TWS for UNIX/JNEXTDAY 8.2
* Start of schedules for CPU MASTER
Schedule DLDAY selected
Schedule DLNIGHT selected
Schedule FINAL selected
*2 schedules selected for CPU MASTER

AWSBHZ028I 2 schedules were selected for all cpus.

$ conman sj MASTER#DL@
TWS for UNIX (AIX)/CONMAN 8.2 (1.36.1.7)
                                    (Est)  (Est)
CPU      Schedule   Job    State Pr Start  Elapse  Dependencies  Return Code

MASTER  #DLDAY    ******** ABEND 10 17:48  00:01
            DLPARDAY ABEND 10 17:48  00:01  #J30824           2

MASTER  #DLNIGHT  ******** HOLD  10(21:45)
                  DLPARNI+ HOLD  10
```

Note: We recommend special care when using the **`Jnextday`** command. This command reruns all scheduled jobs in the production plan that have a starting time between the beginning of the processing day and the new **`Jnextday`** execution.

Example 9-43 shows the results for each scheduled job. The DLPARDAY job has started at 6h30, took 4 minutes to complete, and has a return code of 0. The DLPARNIGHT job is in the HOLD state, because it hasn't been executed yet.

Example 9-43 Job status

```
$ conman sj MASTER#DL@
TWS for UNIX (AIX)/CONMAN 8.2 (1.36.1.7)
                                    (Est)  (Est)
CPU      Schedule   Job    State Pr Start  Elapse  Dependencies  Return Code

MASTER  #DLDAY    ******** SUCC  10 06:30  00:04
                  DLPARDAY SUCC  10 06:30  00:04   #J30844           0

MASTER  #DLNIGHT  ******** HOLD  10(21:45)
                  DLPARNI+ HOLD  10
```

IBM Tivoli Workload Scheduler also provides a graphical user interface called Job Scheduling Console (JCS). Examples are provided to define and register jobs and job streams using JCS in Appendix D, "Using the Job Scheduling Console" on page 471.

9.5 Dynamic logical partitioning integration with HACMP

This section explains how to use DLPAR commands in an HACMP environment to move resources during a takeover. We assume that you have some knowledge of HACMP clustering.

High Availability Cluster Multi-Processing (HACMP) ensures that critical resources are available for processing. A cluster is a group of nodes, shared disks, and networks. A cluster node is an AIX machine (a partition in our case), running HACMP software, which is able to acquire resource groups.

A resource group defines relationships among cluster nodes. It is a set of hardware and software resources. An application server is a component of a resource group. The application server is a cluster resource that associates a name and user-provided scripts to start and stop an application.

In this section we use *cluster node* for things related to HACMP clusters and *partition* for things related to the managed system.

We have a two-sided takeover cluster using cascading resource groups. Each node of the cluster runs on a different partition and has different amounts of resources allocated. We want to move the resources (processors and memory) from one partition to the other if a cluster node fails.

There are many possibilities in HACMP to configure events or add some scripts to move the resources. We chose to use application servers to move resources of partitions because they run at each start or stop of applications. It's a good solution to update the resources of the partition at the same time that the application is started or stopped.

Note: Since HACMP mostly covers hardware problems, it may not be a good idea to create each cluster node in the same managed system.

During the start or stop of an application server, the cluster node sends requests to the HMC to get a quantity of processors and memory. Each partition issues commands to the HMC using SSH, which means that the partitions have to be able to connect to the HMC automatically (no password prompted). In this section, for simplicity, we use an SSH key generated with a null passphrase. For more information on the remote connection to the HMC, refer to 9.1.2, "Connection to the HMC for automated operations" on page 279.

A prerequisite is to have a dedicated network for DLPAR operations between partitions and HMC, for the following reasons:

- In an HACMP cluster, each cluster node has at least two network adapters, one for boot and service addresses, the other for standby addresses. A service address may move to any network adapters in the cluster.
- DLPAR operations are issued from the HMC to partitions through the network. A partition and the HMC establish a path for communication, and will use only that path. Therefore, in a DLPAR environment, each partition of a managed system must have a stable network connectivity with the HMC.

Since a service address of a cluster node may be anywhere in the cluster, the HMC might not be able to issue DLPAR operations to a partition. The solution is to separate the network used by HACMP from the network used by the HMC.

Figure 9-3 on page 315 shows the network topology we use for an HACMP cluster. Cluster nodes A and B have three network adapters each—one for communication between partitions and HMC, the two others for HACMP (boot, service, and standby).

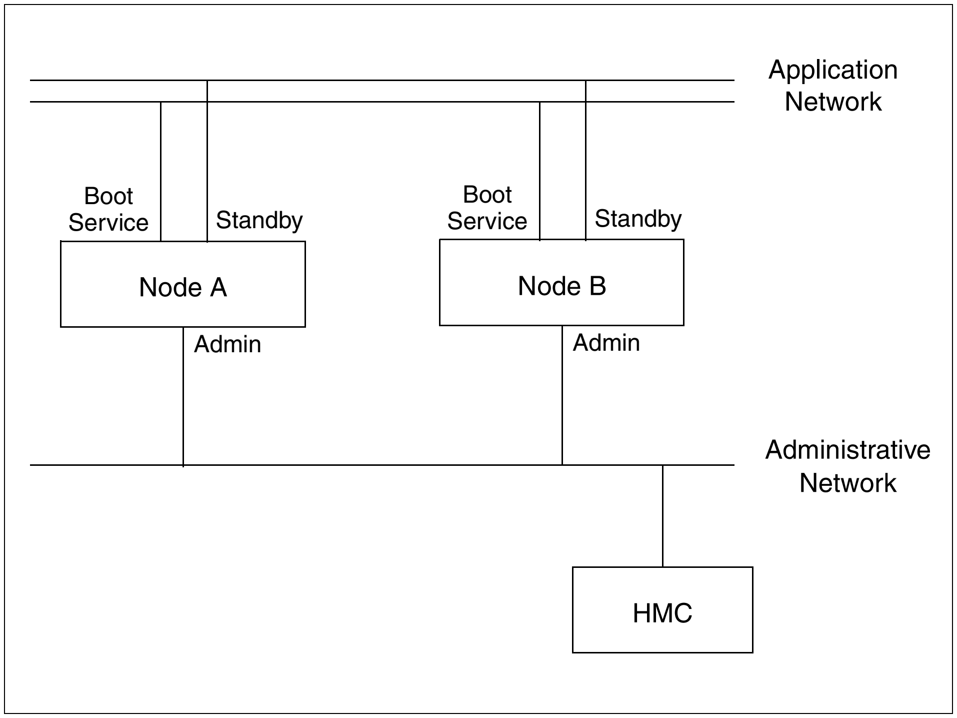


Figure 9-3 Cluster network topology

The HMC needs to communicate with all partitions, so we have to add the administrative network IP addresses of the partitions to the HMC /etc/hosts file. We don't need to add all other IP addresses of the HACMP cluster to that file.

We have to add every IP address of the HACMP cluster to the /etc/hosts file of each partition.

9.5.1 Resource assignment

The amount of resources is different for each partition and has to be updated when a resource group moves to another cluster node.

We develop a Perl script to automate the partition configuration update. It makes requests to the HMC, using the `ssh` command, to add or remove processors and memory for partitions. The script is given a configuration file name as input. This configuration file describes the quantity of processors and memory needed for partitions. It has one line per partition. The syntax of a line is as follow:

- Partition name
- Number of CPUs

- Keyword: CPU
- Number of LMBs
- Keyword: LMB

Example 9-44 shows an example of a configuration file. Partition lpar05 has three processors and 6 GB of memory, and partition lpar06 has two processors and 8 GB of memory.

Example 9-44 Configuration file (test.cfg)

```
lpar05  3 CPU   24 LMB
lpar06  2 CPU   32 LMB
```

The following steps explain what the script does:

1. Check for allocated resources.

   It uses **lshwres** to get processor and memory information from each partition.
2. Evaluate the quantity of resources to add or remove.
3. Check if there are enough available resources.
4. Update the resources.

   It uses **chhwres** to add or remove resources (processors or memory) to/from each partition.

Example 9-45 shows the Perl script listing; the script name is `ChangeResConfig`.

Example 9-45 Perl script file (ChangeResConfig)

```
#!/usr/bin/perl
#
# A program to generate "chhwres" command to run on a machine, which
# issue remote command to the HMC using ssh.
#
# Usage: ChangeResConfig <configfile>
#
# Return code - 88 if config file format is not correct:
#               Lparname n1 CPU n2 LMB
# Return code - 77 if CPU configuration is not possible
# Return code - 66 if MEM configuration is not possible
# However, if target CPU/MEM config is more/less than max/min,
# that line will be ignored
#

$system="ITSO_p690";            # the managed system name
@part=();

$configfile=$ARGV[0];
```

```
$User="hscroot";                # the user to connect to HMC
$HMC="itsohmc";                 # the HMC name
$SSHCmd="ssh $User\@$HMC";      # the SSH command to connect to HMC
$CmdPath="/opt/hsc/bin";        # the path for the command on HMC

# Open and check the config file
open (IN,$configfile) || die "Cannot open $configfile - $!";
while (<IN>) {
        exit 88 unless /(\w+)\s+(\d+)\s+CPU\s+(\d+)\s+LMB/;
        $part=ucfirst($1);
        $part=$1;
        push(@part,$part);
        $target_cpu{$part}=$2;
        $target_mem{$part}=$3;
}

# Get informations from each partition for CPU
foreach $part (@part) {
        print "\tPlease wait.... Listing CPU resource for $part....\n";
        chomp($cpustat=`$SSHCmd $CmdPath/lshwres -m $system -p $part -r cpu
-Fmin:allocated:max:free`);
        ($min_cpu{$part},$current_cpu{$part},$max_cpu{$part},$freecpu)=split(/:
/,$cpustat);

# Get informations from each partition for memory
        print "\tPlease wait.... Listing MEM resource for $part....\n";
        chomp($memstat=`$SSHCmd $CmdPath/lshwres -m $system -p $part -r mem
-Fmin:allocated:max:free:lmb_size`);
        ($min_mem{$part},$current_mem{$part},$max_mem{$part},$freemem)=split(/:
/,$memstat);
}

# Display collected informations
foreach $part (@part) {
        write;
}

format STDOUT_TOP =
============================================================================
                CPU                             MEM
Current   Target   Min   Max  Free      Current  Target  Min  Max  Free LMB
============================================================================
.
format STDOUT =
@<<<<     @<<<<  @<<<<  @<<<<  @<<<<    @>>>>>  @>>>>>  @>>>  @>>>  @>>> @>>>
$current_cpu{$part},$target_cpu{$part},$min_cpu{$part},$max_cpu{$part},$freecpu
,$current_mem{$part},$target_mem{$part},$min_mem{$part},$max_mem{$part},$freeme
m,$lmb_size
```

```
.

$cpusum=0;
$memsum=0;

# Evaluate the quantity of CPU needed
foreach $part (@part) {
        $needed_cpu{$part}=$target_cpu{$part}-$current_cpu{$part};
        $cpusum += $needed_cpu{$part};
}
if ($cpusum > $freecpu) {
        print "\tThere is not enough CPU to satisfy the request\n\n";
        exit 77;
}

# Evaluate the quantity of memory needed
foreach $part (@part) {
        $needed_mem{$part}=$target_mem{$part}-$current_mem{$part};
        $memsum += $needed_mem{$part};
}
if ($memsum > $freemem) {
        print "\tThere is not enough MEM to satisfy the request\n\n";
        exit 66;
}

# Check if resources are available, create the command to be executed
foreach $part (@part) {
        $CPUCMD{$part}="";
        next if $target_cpu{$part} > $max_cpu{$part};
        next if $target_cpu{$part} < $min_cpu{$part};
        next if $needed_cpu{$part} == 0;
        print "\t***NEEDED CPU for $part is $needed_cpu{$part}***\n";
        ($needed_cpu{$part} > 0) ? ($op = "a") : ($op = "r");
        $needed_cpu{$part} = abs($needed_cpu{$part});
        $CPUCMD{$part}="$SSHCmd $CmdPath/chhwres -m $system -o $op -r cpu -q
$needed_cpu{$part} -p $part";
}

# Check if resources are available, create the command to be executed
foreach $part (@part) {
        $MEMCMD{$part}="";
        next if $target_mem{$part} > $max_mem{$part};
        next if $target_mem{$part} < $min_mem{$part};
        next if $needed_mem{$part} == 0;
        print "\t***NEEDED MEM for $part is $needed_mem{$part}***\n";
        ($needed_mem{$part} > 0) ? ($op = "a") : ($op = "r");
        $needed_mem{$part} = abs($needed_mem{$part});
        $MEMCMD{$part}="$SSHCmd $CmdPath/chhwres -m $system -o $op -r mem -q
$needed_mem{$part} -p $part";
}

# Run commands for CPU
# Reverse sort will rearrange "-o r" command to be run before "-o a" command!
foreach $cpucmd (reverse sort values %CPUCMD) {
        if (length($cpucmd) != 0) {
           print "Running $cpucmd...\n";
           system "$cpucmd";
        }
}

# Run commands for memory
# Reverse sort will rearrange "-o r" command to be run before "-o a" command!
foreach $memcmd (reverse sort values %MEMCMD) {
        if (length($memcmd) != 0) {
           print "Running $memcmd...\n";
           system "$memcmd";
        }
}
```

The output of the Perl script execution is shown in Example 9-46. Partition lpar06 has one processor allocated. The configuration file requests two, so one processor has to be added. Partition lpar06 has 9 GB of memory and the configuration file request is for 8 GB, so 1 GB has to be removed.

Example 9-46 Script execution output

```
root@lpar05:/ # ChangeResConfig test.cfg
        Please wait.... Listing CPU resource for lpar05....
        Please wait.... Listing MEM resource for lpar05....
        Please wait.... Listing CPU resource for lpar06....
        Please wait.... Listing MEM resource for lpar06....
============================================================================
                CPU                               MEM
Current   Target   Min   Max  Free      Current  Target  Min  Max  Free LMB
============================================================================
3         3      1      8      4           20      24    12    88   158
1         2      1      8      4           36      32     8    88   158
        ***NEEDED CPU for lpar06 is 1***
        ***NEEDED MEM for lpar05 is 4***
        ***NEEDED MEM for lpar06 is -4***
Running ssh hscroot@itsohmc /opt/hsc/bin/chhwres -m ITSO_p690 -o a -r cpu -q 1
-p lpar06 -d 5...
Running ssh hscroot@itsohmc /opt/hsc/bin/chhwres -m ITSO_p690 -o r -r mem -q 4
-p lpar06 -d 5...
Running ssh hscroot@itsohmc /opt/hsc/bin/chhwres -m ITSO_p690 -o a -r mem -q 4
-p lpar05 -d 5...
```

9.5.2 Integration with HACMP

We want to update resources for each start and stop of an application server. We call the ChangeResConfig script in every start and stop script.

Steps an application server follows when starting:

1. Select the configuration file, depending on the cluster node and the application.
2. Add resources to the partition.
3. Start all the applications.

Steps an application server follows when stopping:

1. Stop all the applications running on the cluster node.
2. Select the configuration file, depending on the cluster node and the application.
3. Remove resources from the partition.

The DLPAR operations may take a long time, which may lead to timeouts in HACMP. There are two ways to avoid this:

- Use the `at` command to launch the ChangeResConfig script. The application should be able to start even if all resources are not available yet.
- Change the timeout delay for the event config_too_long in the HACMP configuration.

9.5.3 Application server scripts

In this section we create start and stop scripts, using DLPAR operations, for an application server. We have two application servers: app1 and app2. Each has a start and a stop script and each of those scripts loads a configuration file, depending on the situation. Application server app1 is started by default on partition lpar05, application server app2 on partition lpar06. Application server app1 starts application 1 and application server app2 starts application 2.

Configuration files

The configuration files define the amount of processors and memory to allocate to partitions. We use the following configuration files in our cluster:

- app1.cfg: resources to run application 1
- app2.cfg: resources to run application 2
- app1app2lpar05.cfg: resources to run application 1 and 2 on lpar05
- app1app2lpar06.cfg: resources to run application 1 and 2 on lpar06

- app1_stby.cfg: resources to run AIX with no application
- app2_stby.cfg: resources to run AIX with no application

Example 9-47 shows configuration files on lpar05. For example, lpar05 needs three processors and 5 GB of memory in configuration file app1.cfg.

Example 9-47 Configuration files on lpar05

```
root@lpar05:/ # cat app1.cfg
lpar05  3 CPU   20 LMB
root@lpar05:/ # cat app1_stby.cfg
lpar05  1 CPU   8 LMB
root@lpar05:/ # cat app1app2lpar05.cfg
lpar05  4 CPU   30 LMB
```

Example 9-48 shows configuration files on lpar06.

Example 9-48 Configuration files on lpar06

```
root@lpar06: # cat app2.cfg
lpar06  1 CPU   12 LMB
root@lpar06: # cat app2_stby.cfg
lpar06  1 CPU   8 LMB
root@lpar06: # cat app1app2lpar06.cfg
lpar06  4 CPU   30 LMB
```

Application server start script

Example 9-49 shows an application server start script. It adds resources before beginning the application. If the start script is executed on the original cluster node (the cluster node to which the resource group belongs), it loads the configuration file app1.cfg. Otherwise (a fallover has occurred), it loads the app1app2NodeB.cfg configuration file.

Example 9-49 Application server start script

```
#!/bin/ksh
# Start application server app1
#

NodeName="lpar05"
LocalNodeName=$(/usr/es/sbin/cluster/utilities/get_local_nodename)
HACMPScript="/exploit"


banner "Begin"

print "Start app1"
```

```
# Select the right config file
if [[ $NodeName = $LocalNodeName ]]
then
   $HACMPScript/ChangeResConfig $HACMPScript/app1.cfg
else
   $HACMPScript/ChangeResConfig $HACMPScript/app1app2NodeB.cfg
fi

####################################################
#
# Add the start scripts for your application here
#
####################################################

print "Starting application 1"
print "..."
print "Application 1 started"

banner "End"
```

Application server stop script

Example 9-50 shows an application server stop script. It removes resources just after the stop of the application. If the stop script is executed on the original cluster node, it loads the configuration file app1_stby.cfg. Otherwise, it loads the app2.cfg configuration file.

Example 9-50 Application server stop script

```
#!/bin/ksh
# Stop application server app1
#

NodeName="lpar05"
LocalNodeName=$(/usr/es/sbin/cluster/utilities/get_local_nodename)
HACMPScript="/exploit"


banner "Begin"
print "Stop app1\n"

####################################################
#
# Add the stop scripts for your application here
#
####################################################

print "Stopping application 1"
print "..."
print "Application 1 stopped"

# Select the right config file
if [[ $NodeName = $LocalNodeName ]]
then
   $HACMPScript/ChangeResConfig $HACMPScript/app1_stby.cfg
else
   $HACMPScript/ChangeResConfig $HACMPScript/app2.cfg
fi

banner "End"
```

Some improvements to the scripts can be made, such as:

- Verify that all resources have been allocated to the partition before starting applications.
- Check whether another application is running, to select the right configuration file. For example, if app1 is running on lpar06 (after a fallover) and we want to stop app2, the stop script will not leave the needed resources for app1.
- Add configuration files to start only app2 on lpar05.

HACMP and DLPAR operation tests

In order to test DLPAR operations in an HACMP cluster, we do the following:

- Start HACMP on partition lpar05.
- Force a fallover from partition lpar05 to partition lpar06.

  This stops application server app1 on partition lpar05 and starts it on partition lpar06.

Example 9-51 shows the hacmp.out file when starting application server app1 on lpar05. The partition lpar05 needs two processors to meet the configuration file requirements, which will be added to the partition.

Example 9-51   hacmp.out file

```
Aug 13 16:32:23 EVENT START: start_server app1

RG1:start_server[50] [[ high = high ]]
RG1:start_server[50] version=1.4.1.9
RG1:start_server[51] RG1:start_server[51] cl_get_path
...
... omitted lines ...
...
######
#     #  ######   ####      #    #    #
#     #  #       #    #     #    ##   #
######   #####   #          #    # #  #
#     #  #       #  ###     #    #  # #
#     #  #       #    #     #    #   ##
######   ######   ####      #    #    #

Start app1
        Please wait.... Listing CPU resource for lpar05....
        Please wait.... Listing MEM resource for lpar05....
============================================================================
                CPU                             MEM
Current   Target   Min   Max  Free      Current  Target  Min  Max  Free LMB
============================================================================
1         3      1      8      6            20      20    12    88   182
        ***NEEDED CPU for lpar05 is 2***
Running ssh hscroot@itsohmc /opt/hsc/bin/chhwres -m ITSO_p690 -o a -r cpu -q 2
-p lpar05...
Starting application 1
...
Application 1 started
#######
#        #    #  #####
#        ##   #  #    #
#####    # #  #  #    #
#        #  # #  #    #
#        #   ##  #    #
#######  #    #  #####
...
... omitted lines ...
...
RG1:start_server[140] exit 0
Aug 13 16:32:25 EVENT COMPLETED: start_server app1
...
... omitted lines ...
...
RG1:process_resources[1520] [ 0 -ne 0 ]
RG1:process_resources[1760] break
RG1:process_resources[1771] [[ FALSE = TRUE ]]
RG1:process_resources[1778] exit 0
:node_up_complete[278] [ 0 -ne 0 ]
:node_up_complete[286] exit 0
Aug 13 16:32:28 EVENT COMPLETED: node_up_complete lpar05
```

In Example 9-52, we can verify that on partition lpar05, three processors and 5 GB of memory are allocated.

Example 9-52 Resources on lpar05 after starting application server

```
root@lpar05:/ # lsdev -Ccprocessor
```

```
proc0 Available 00-00 Processor
proc5 Available 00-05 Processor
proc6 Available 00-06 Processor

root@lpar05:/ # lsattr -El mem0
goodsize 5120 Amount of usable physical memory in Mbytes False
size     5120 Total amount of physical memory in Mbytes  False
```

We stop cluster services on lpar05 using takeover mode. This moves the resource group that was on lpar05 to lpar06, exactly as if a problem had occurred on lpar05. This is the fallover of lpar05 to lpar06.

Example 9-53 shows part of the hacmp.out log file on lpar05. The application server app1 is requested to stop. The resources of lpar05 have to be reduced to meet the configuration file requirement, which is to have only one processor and 2 GB of memory.

Example 9-53 hacmp.out for lpar05

```
Stop app1
Stopping application 1
...
Application 1 stopped
        Please wait.... Listing CPU resource for lpar05....
        Please wait.... Listing MEM resource for lpar05....
=============================================================================
                CPU                             MEM
Current   Target   Min   Max  Free      Current  Target  Min  Max  Free LMB
=============================================================================
3         1       1      8      4           20       8     8    88   182
        ***NEEDED CPU for lpar05 is -2***
        ***NEEDED MEM for lpar05 is -12***
Running ssh hscroot@itsohmc /opt/hsc/bin/chhwres -m ITSO_p690 -o r -r cpu -q 2
-p lpar05...
Running ssh hscroot@itsohmc /opt/hsc/bin/chhwres -m ITSO_p690 -o r -r mem -q 12
-p lpar05...
```

We can verify in Example 9-54 that on partition lpar05, only one processor and 2 GB of memory are now allocated.

Example 9-54 Resource on lpar05 after stopping application server

```
root@lpar05:/ # lsdev -Ccprocessor
proc5 Available 00-05 Processor

root@lpar05:/ # lsattr -El mem0
goodsize 2048 Amount of usable physical memory in Mbytes False
```

```
size     2048 Total amount of physical memory in Mbytes  False
```

Example 9-55 shows part of the hacmp.out log file on lpar06. The application server starts on lpar06 after the failure of lpar05. The start script increases the resources as defined in the configuration file so that partition lpar06 can run application 1 while application 2 is already running.

Example 9-55   hacmp.out for lpar06

```
Start app1
        Please wait.... Listing CPU resource for lpar06....
        Please wait.... Listing MEM resource for lpar06....
============================================================================
                CPU                                MEM
Current   Target   Min   Max  Free      Current  Target  Min  Max  Free LMB
============================================================================
1         4      1      8      6            12      30     8    88   182
        ***NEEDED CPU for lpar06 is 3***
        ***NEEDED MEM for lpar06 is 18***
Running ssh hscroot@itsohmc /opt/hsc/bin/chhwres -m ITSO_p690 -o a -r cpu -q 3
-
p lpar06...
Running ssh hscroot@itsohmc /opt/hsc/bin/chhwres -m ITSO_p690 -o a -r mem -q 18
-p lpar06...
Starting application 1
...
Application 1 started
```

The backup node not only takes the HACMP resources (resource group) from the failed node, but also takes the partition resources (processors and memory).

Part 3

Advanced programming examples

10

Dynamic reconfiguration using DLPAR scripts

In this chapter, we offer a sample program that shows how an application can autonomically adjust its resource sets to dynamic reconfiguration of a partition.

First we discuss the benefits of dynamically reconfiguring applications when a DLPAR event occurs. We discuss briefly the type of application that is suited for this. Then we introduce the script dr_httpd_reconfig.sh, which is developed from the IBM_template.sh script that you can find in the appendix.

We explain the necessary changes to the template script so that the script autonomically changes the configuration of the IBM HTTP Server whenever the number of processors in a partition is changed.

10.1 Type of applications that benefit from DLPAR

There is quite a number of applications which will benefit from an automatic reconfiguration whenever a change in resources occurs. We do not want show you a complete list of applications here, but we will give you a brief overview which type of application will take advantage of a dynamic reconfiguration.

Software licences are sometimes bound to the number of processors or even to the CPU IDs. Therefor you might have to change the configuration of your licences when a processor is removed from or added to a partition.

Another example would be a http proxy server running in a partition. Using paging space for caching in a proxy server has severe impact on the performance. For preventing the server from paging and therefor keeping all the cached data in memory you might have to reduce the cache size whenever the memory size is reduced in a partition. For benefitting from an enlarged memory size you might want to enlarge the cache size when a DLPAR event occurs.

10.2 A sample script to reconfigure the IBM HTTP Server

For this example we need the IBM HTTP Server (IHS) installed in the partition. First we show how to install the IHS in the partition. Then we explain four scripts that we use to modify the configuration of the IHS. Finally we show how to modify the sample drmgr script that comes with AIX 5L Version 5.2 so the IHS is reconfigured dynamically.

Before beginning with the examples, make sure that the IBM HTTP Server is installed. Use the following command:

```
# lslpp -l http_server.\*
lslpp: 0504-132  Fileset http_server.* not installed.
```

Note: If you see this message, the IHS has not been installed using the AIX installation procedure, but it could still have been installed manually.

If the IHS is not installed on the system, refer to 10.2.1, “Installation of the IBM HTTP Server” on page 331. If it is installed in the partition, skip the following section and proceed to 10.2.2, “Configuration of the httpd processes” on page 332.

10.2.1 Installation of the IBM HTTP Server

This section explains how to install the IBM HTTP Server on an AIX partition.

Product contents

The IBM HTTP Server is shipped as part of the AIX expansion pack CD-ROM or can be downloaded from:

```
http://www.software.ibm.com/webservers/httpservers
```

The IBM HTTP Server comes in several file packages that contain the IBM HTTP Server and SSL filesets, as follows:

- Base package, without SSL security:
  - http_server.base - Contains the IBM HTTP Server base and source filesets
- SSL module and SSL library packages (required for SSL):
  - http_server.ssl - Contains the IBM HTTP Server SSL module fileset

Pre-installation setup

Before starting the installation of the IBM HTTP Server, you should make sure that the following are in place:

- Have the install packages available. These either come as a CD-ROM or as separate files.
- You must have root privileges on the server where you want to install the IHS.
- You should meet the minimum hardware and software requirements for the IHS.
- Determine the filesets you need.

Installation

For installation of the IHS only, you need the http_server.base.core fileset. To install the IHS, change to the directory where the fileset is located and use the following command:

```
# installp -aYd . http_server.base
```

To make sure the installation is complete and the package is installed properly, use the following command:

```
# lslpp -l http_server.base.\*
Fileset                      Level  State      Description
  ----------------------------------------------------------------------------
Path: /usr/lib/objrepos
  http_server.base.license  1.3.19.3  COMMITTED  HTTP Server Licenses
  http_server.base.rte      1.3.19.3  COMMITTED  HTTP Server Base Run-Time
  http_server.base.source   1.3.19.3  COMMITTED  HTTP Server Source Code

Path: /etc/objrepos
  http_server.base.rte      1.3.19.3  COMMITTED  HTTP Server Base Run-Time
```

Your output may differ depending on the version of the IHS you have installed.

Starting the IHS

We want to make sure the IHS is running. The following command restarts the IHS if it is already running and starts it if it is not running:

```
# /usr/HTTPServer/bin/apachectl restart
```

10.2.2 Configuration of the httpd processes

The settings of the IHS are configured in the /usr/HTTPServer/conf/http.conf file.

We identify two parameters in the configuration file that will change the performance of the IHS. In case of a DLPAR event, these parameters will be changed in the configuration file and the IHS will be restarted. We use these parameters just as example, and of course you can use this technique to change other parameters as needed.

To define the number of httpd processes started by the IHS at startup, you can change the value of StartServers. The default value in httpd.conf is:

```
StartServers 5
```

This number defines the minimum number of httpd processes that the IHS spawns at startup.

You can define the maximum number of connections by changing the following variable in the configuration file:

```
MaxClients 150
```

As the load on the IHS increases with the number of requests it receives, it will start more httpd processes to serve the incoming requests. This number defines the maximum number of httpd processes running.

The values shown are the default values you will find in the configuration file after you install the IHS.

Reloading the configuration file

You can tell the IHS to reload its configuration file in two ways.

HUP Signal: restart now

If you send the HUP signal to the IHS, it will cause the httpd parent to *kill* all its child processes. The parent rereads the configuration file, reopens the log files, and starts a new set of child processes.

USR1 signal: graceful restart

The USR1 signal causes the httpd parent process to *advise* its children to exit after their current request. The parent rereads its configuration files and reopens its log files. As each child dies, the parent starts a new process with the new configuration parameters.

The graceful restart method is designed to minimize the time in which the server is unable to serve requests. We will use this method for reconfiguring the IHS server in the following examples.

10.2.3 Scripts to reconfigure the IHS

For each value we want to modify, we have two scripts—one for reading the current value from the configuration file, and one for changing the value in the configuration file.

Reading values from the configuration file

We first explain the scripts we use for reading the values from the configuration file. The Perl script lsStartServers.pl shown in Example E-1 on page 484 reads the httpd.conf file and finds the StartServers stanza with a regular expression. The value of this regular expression is printed to standard out.

The following example shows how you can check the value for StartServers in the httpd.conf file:

```
# lsStartServers.pl
5
```

The lsMaxClients.pl script in Example E-2 on page 485 is an almost identical Perl script that reads the value in the MaxClients stanza from the httpd.conf file. In the following example we present an analog example for the value of MaxClients:

```
# lsMaxClients.pl
150
```

Set values

For setting the values for StartServers and MaxClients we have two scripts that open the configuration file and write the changed value to the file. For setting the value for StartServers we use chStartServers.pl. You can find the source code in Example E-1 on page 484.

For changing the MaxClients value we use chMaxClients.pl. The source code for this script is in Example E-3 on page 486.

The following examples show how to use these two scripts:

```
# chStartServers.pl 10
# chMaxClients.pl 100
```

In the following examples we define 10 processes per CPU and a maximum number of 100 connections per CPU for the IHS.

10.2.4 Add one CPU and reconfigure the IHS

In this example we start with one processor in the partition and then add another one with an DLPAR operation.

We use the IBM_template.sh file (see Appendix B-3, “DLPAR script template: Korn shell” on page 422) and modify it so that it reconfigures and restarts the IHS in case of a dynamic reconfiguration of the partition. We only show the subroutines changed from the IBM_template.sh file. We call our modified Korn shell script dr_httpd_reconfig.sh.

We introduce two variables that define the values of StartServers per CPU and MaxClients per CPU. We use these two variables in our sample Perl scripts:

```
dStartServers=10
dMaxClients=100
```

These are the values we decrease or increase StartServers or MaxClients by if a processor is taken out or added to the partition.

When we increase the number of processors in the partition, we change the configuration of the IHS after the processor is actually taken out of the system. Therefore, the change has to be made in the post acquire phase. The subroutine we change in the script is process_postacquire().

The steps the script has to do are as follows:

1. Enlarge StartServers by dStartServers in the httpd.conf file.
   - Get the current number for StartServers.
   - Add dStartServers to the current StartServers.
   - Set the new value for StartServers in httpd.conf.
2. Enlarge MaxClients by dMaxClients in the httpd.conf.
   - Get the current number for MaxClients.
   - Add dMaxClients to the current MaxClients.

   – Set the new value for MaxClients in httpd.conf.
3. Restart the IHS.
   – Do a graceful restart of httpd.

Example Example 10-1 shows the code for the process_postacquire() subroutine in the dr_httpd_reconfig.sh script. To simplify the code, we use the Perl scripts we introduced before for reading and writing the httpd.conf file.

Example 10-1 Example code for the process_postacquire() subroutine

```
process_postacquire()
{
  dStartServers=10
  dMaxClients=100

  case"$1" in
      "cpu")
          # get current configuration for StartServers
          nStartServers=`/tmp/lsStartServers.pl`
          # get new number for StartServers
          newStartServers=$(($nStartServers + $dStartServers))
          # change configuration for StartServers in httpd.conf
          /tmp/chStartServers.pl $newStartServers

          # get current configuration for MaxClients
          nMaxClients=`/tmp/lsMaxClients.pl`
          # get new number for MaxClients"
          newMaxClients=$(($nMaxClients + $dMaxClients))
          # change configuration for MaxClients in httpd.conf"
          /tmp/chMaxClients.pl $newMaxClients

          # reload apache config with graceful restart
          /usr/HTTPServer/bin/apachectl graceful

          ;;
      "mem")
          dbg "Resource : mem"
          ;;
      *)
          echo "DR_ERROR=Script does not support Resource $1"
          ;;
  esac

  return 0
}
```

Registering the script

Change to the directory where the dr_httpd_reconfig.sh script is located and issue the following command:

```
lpar06# drmgr -i dr_http_reconfig.sh
```

You can check if the script is properly registered with:

```
lpar06# drmgr -l
DR Install Root Directory: /usr/lib/dr/scripts
Syslog ID: DRMGR
------------------------------------------------------------
/usr/lib/dr/scripts/all/dr_httpd_reconfig.sh          AIX DR ksh example
script
        Vendor:IBM,     Version:1,      Date:10182002
        Script Timeout:10,      Admin Override Timeout:0
        Resources Supported:
                Resource Name: cpu       Resource Usage: cpu binding for
performance
                Resource Name: mem       Resource Usage: Shared(Pinned) memory
for app XYZ
------------------------------------------------------------
```

Sample DLPAR action

The following shows the output when we add one processor to the partition.

Before we start the DLPAR operation, we check the status of the processor and the configuration of the IHS:

```
lpar06 # lsdev -Cc processor
proc20 Available 00-20 Processor
lpar06 # lsStartServers.pl
10
lpar06 # lsMaxClients.pl
100
```

Then we initiate the DLPAR operation with the following command on the HMC:

```
hmc $ chhwres -m ITSO_p690 -m lpar07 -r cpu -o a -q 1 -W 4 -d 4
```

After the DLPAR operation has completed, we again check the number of CPUs and the configuration of the IHS to verify the changes:

```
lpar06# lsdev -Cc processor
proc20 Available 00-20 Processor
proc4 Available 00-04 Processor
lpar06# lsStartServers.pl
20
lpar06# lsMaxClients.pl
200
```

We can see that the drmgr has autonomously reconfigured the IHS. The new configuration is adjusted to the available resources.

10.2.5 Remove one CPU and reconfigure the IHS

In the next example we start with two processors in the partition and then remove one processor with a DLPAR operation.

Again we use the two variables that define the values of StartServers per CPU and MaxClients per CPU:

```
dStartServers=10
dMaxClients=100
```

When we reduce the number of processors in the partition, we change the configuration of the IHS before the processor is actually taken out of the system. Therefore, the change has to be made in the prerelease phase. The subroutine we change from the template script is the process_prerelease().

The steps the script has to do are as follows:

1. Decrease StartServers by dStartServers in the httpd.conf file.
   - Get the current number for StartServers.
   - Subtract dStartServers from the current StartServers.
   - Set the new value for StartServers in httpd.conf.
2. Decrease MaxClients by dMaxClients in the httpd.conf.
   - Get the current number for MaxClients.
   - Subtract dMaxClients from the current MaxClients.
   - Set the new value for MaxClients in httpd.conf.
3. Restart the IHS.
   - Do a graceful restart of httpd.

Example 10-2 shows the code for the process_prerelease() subroutine in the dr_httpd_reconfig.sh script. Again, we use our Perl scripts for reading and writing the httpd.conf file.

Example 10-2 Example code for the process_prerelease() subroutine

```
process_prerelease()
{
  dStartServers=10
  dMaxClients=100

  case"$1" in
```

```
    "cpu")
        # get current configuration for StartServers
        nStartServers=`/tmp/lsStartServers.pl`
        # get new number for StartServers
        newStartServers=$(($nStartServers - $dStartServers))
        # change configuration for StartServers in httpd.conf
        /tmp/chStartServers.pl $newStartServers

        # get current configuration for MaxClients
        nMaxClients=`/tmp/lsMaxClients.pl`
        # get new number for MaxClients"
        newMaxClients=$(($nMaxClients - $dMaxClients))
        # change configuration for MaxClients in httpd.conf"
        /tmp/chMaxClients.pl $newMaxClients

        # reload apache config with graceful restart
        /usr/HTTPServer/bin/apachectl graceful

        ;;
    "mem")
        dbg "Resource : mem"
        ;;
    *)
        echo "DR_ERROR=Script does not support Resource $1"
        ;;
  esac

  return 0
}
```

Registering the script

Change to the directory where the dr_httpd_reconfig.sh script is located and issue the following command:

```
lpar06 # drmgr -i dr_http_reconfig.sh
```

With the following command you can check that the script is properly registered:

```
lpar06 # drmgr -l
DR Install Root Directory: /usr/lib/dr/scripts
Syslog ID: DRMGR
------------------------------------------------------------
/usr/lib/dr/scripts/all/dr_httpd_reconfig.sh           AIX DR ksh example
script
        Vendor:IBM,     Version:1,      Date:10182002
        Script Timeout:10,      Admin Override Timeout:0
        Resources Supported:
                Resource Name: cpu       Resource Usage: cpu binding for
performance
                Resource Name: mem       Resource Usage: Shared(Pinned) memory
for app XYZ
------------------------------------------------------------
```

Sample DLPAR action

As an example, we show the output when we remove one processor from the partition.

First we check the status of the processors and the configuration of the IHS before we start the DLPAR operation:

```
lpar06 # lsdev -Cc processor
proc20 Available 00-20 Processor
proc4  Available 00-04 Processor

lpar06 # lsStartServers.pl
20
lpar06 # lsMaxClients.pl
200
```

Then we initiate the DLPAR operation with the following command on the HMC:

```
hmc$ chhwres -m ITSO_p690 -p lpar07 -r cpu -o r -q 1 -W 4 -d 4
```

After the DLPAR operation has completed, we again check the number of processors and the configuration of the IHS to verify the changes:

```
lpar06 # lsdev -Cc processor
proc4 Available 00-04 Processor
lpar06 # lsStartServers.pl
10
lpar06 # lsMaxClients.pl
100
```

We can see that the drmgr has autonomously reconfigured the IHS. The new configuration is adjusted to the available resources.

11

Resource sets

A resource set is a bundle of system resources. It can consist of one or more processors, one or more memory pools, or a mixture of memory pools and processors.

In this chapter we describe how to work with resource sets. First we show how to use the system commands **`lsrset`**, **`mkrset`**, and **`rmrset`** to list, create, and delete resource sets, respectively. Then we show how to use the rset API to manipulate resource sets from C source files.

Finally we give a sample program that shows how an application can autonomously adjust its resource sets to dynamic reconfiguration of the partition.

11.1 rset commands

In this section we explain how you can work with resource sets using the rset commands **lsrset**, **mkrset**, and **rmrset**.

11.1.1 lsrset

Now we explain how you can use the **lsrset** command to display the resource sets registered in the system global repository. If you issue the command with the **-a** option without having any resource sets created, it will give you the default resource sets, as follows:

```
# lsrset -a
sys/sys0
sys/node.01.00000
sys/node.02.00000
sys/node.03.00000
sys/node.04.00000
sys/node.05.00000
sys/node.05.00001
sys/mem.00000
sys/cpu.00000
sys/cpu.00001
```

These predefined resource sets show basically the various system detail levels. In this system we have 7 System Detail Levels (SDL). The first line represents the whole system on SDL 0. Then follow 5 lines for the SDLs 1 to 5 where the MCM, for example, is on SDL 3. The CPU is on the lowest level SDL 6, which is represented by the sys/cpu.00000 and sys/cpu.00001 resource sets. These two resource sets contain the two processors in our partition.

With the **-r** option you can view a particular resource set, and the **-v** option gives you a more detailed output:

```
# lsrset -vr sys/sys0
T  Name               Owner   Group   Mode    CPU  Memory
r  sys/sys0           root    system  r-r-r-    2    2048
     CPU: 0-1
     MEM: 0
```

The **-o** option on the **lsrset** command is used to show only the resources that are active in the partition. If we use that command on the sys/sys0 resource set, we see the same result as without the **-o** option:

```
# lsrset -vor sys/sys0
T  Name               Owner   Group   Mode    CPU  Memory
r  sys/sys0           root    system  r-r-r-    2    2048
     CPU: 0-1
     MEM: 0
```

This is because our partition can have a maximum of two processors and both processors are available.

```
# lsdev -Cc processor
proc3 Available 00-03 Processor
proc4 Available 00-04 Processor
```

We can configure one of the processors out of the partition and remain with only one available processor.

```
hmc $ chhwres -m ITSO_p690 -p lpar08 -r cpu -o r -q 1 -W 4 -d 4

# lsdev -Cc processor
proc3 Available 00-03 Processor
```

The output shows that we have only one processor left in our partition. Now we see a difference between the output with and without the **-o** option on the **lsrset** command:

```
# lsrset -vr sys/sys0
T  Name               Owner   Group   Mode    CPU  Memory
r  sys/sys0           root    system  r-r-r-    2    2048
     CPU: 0-1
     MEM: 0

# lsrset -vor sys/sys0
T  Name               Owner   Group   Mode    CPU  Memory
r  sys/sys0           root    system  r-r-r-    1    2048
     CPU: 0
     MEM: 0
```

Without the **-o** option we see all the processors in the resource set but when we issue the **lsrset** command with the **-o** option, we only see the available resources. Only the processor with the logical CPU number 0 is shown because the other one has been deconfigured by the DLPAR operation.

This can be used to get the logical CPU numbers of the processors available in the partition. Since the logical CPU numbers are not necessarily in consecutive order, this can be a useful command.

You can create a resource set with all possible processors in it, even if not all of these processors are available in the partition:

```
# lsrset -vr test/rset
T  Name               Owner   Group   Mode    CPU  Memory
a  test/rset          root    system  rwr-r-  256       0
     CPU: 0-255
     MEM: <empty>
```

The **lsrset** command with the **-o** option shows all available processors with their logical CPU numbers:

```
# lsrset -ovr test/rset
T  Name               Owner   Group   Mode    CPU  Memory
a  test/rset          root    system  rwr-r-    2       0
     CPU: 0-1
     MEM: <empty>
```

In the next section the **lsrset** command is used to display the resource sets we create with the **mkrset** command.

11.1.2 mkrset

The **mkrset** command is used to create resource sets. In the following example, we create a resource set named rset in the namespace test with one processor in it. The processor in the resource set is the one that has the logical CPU number 0.

```
# mkrset -c 0 test/rset
1480-353 rset test/rset created.
```

We see our resource set listed together with the default resource sets in the system global repository:

```
# lsrset -a
sys/sys0
sys/node.01.00000
sys/node.02.00000
sys/node.03.00000
sys/node.04.00000
sys/node.05.00000
sys/node.05.00001
sys/mem.00000
sys/cpu.00000
sys/cpu.00001
test/rset
```

We can view the configuration of the newly created resource set with the **lsrset** command. The output shows our resource set with logical CPU 0:

```
# lsrset -vr test/rset
T  Name               Owner   Group   Mode    CPU  Memory
a  test/rset          root    system  rwr-r-    1       0
     CPU: 0
     MEM: <empty>
```

Note: You cannot call **mkrset** to modify an existing resource set. You have to delete the existing resource set first.

Processors contained in the resource sets you create don't have to be available in a partition. In the next example we create a resource set with ten processors while there are only two processors available in the partition:

```
# mkrset -c 0-9 test/rset
1480-353 rset test/rset created.
```

The **lsrset** command shows the resource set we have created. We can see that the resource set contains ten processors and there are two processors available.

```
# lsrset -vr test/rset
T  Name              Owner   Group   Mode    CPU  Memory
a  test/rset         root    system  rwr-r-   10       0
     CPU: 0-9
     MEM: <empty>

# lsrset -vor test/rset
T  Name              Owner   Group   Mode    CPU  Memory
a  test/rset         root    system  rwr-r-    2       0
     CPU: 0-1
     MEM: <empty>
```

The next example shows you how to create a resource set containing memory:

```
# mkrset -c 0 -m 0 test/rset
1480-353 rset test/rset created.

# lsrset -vr test/rset
T  Name              Owner   Group   Mode    CPU  Memory
a  test/rset         root    system  rwr-r-    1    2048
     CPU: 0
     MEM: 0
```

With the **mkrset** command you can only create resource sets that have at least one processor in it. The creation of a resource set without any processors in the resource set will fail.

```
# mkrset -m 0 test/rset
1480-352 Specify the -c option.
1480-350 Usage: mkrset -c cpuNum [cpuNum] [-m memNum [memNum]] rsetname
```

You have to specify a processor even if the logical CPU you specify is not available in the partition:

```
# mkrset -c 255 -m 0 test/rset
1480-353 rset test/rset created.

# lsrset -vr test/rset
T  Name              Owner   Group   Mode    CPU  Memory
a  test/rset         root    system  rwr-r-    1    2048
     CPU: 255
```

```
    MEM: 0
```

Note: For creating a resource set containing memory with **mkrset**, there must be at least one processor in the resource set.

Even if the syntax of the **mkrset** command allows you to create resource sets with one or more memory regions in AIX 5L Version 5.2, you can only create resource sets with either all the available memory or no memory at all. It is not possible to create a resource set specifying one or more regions with the **-m** option even if the **mkrset** command will not fail.

11.1.3 rmrset

The **rmset** command is used to remove a resource set from the system global repository:

```
# rmrset test/rset
1480-401 rset 'test/rset' deleted.
```

Note: You cannot remove one of the predefined resource sets from the system global repository.

```
# rmrset sys/sys0
1480-407 The requested operation is not permitted.
```

11.2 The rset API

The rset API is used to manipulate resource sets from within C source code.

In AIX 5L Version 5.2 the rset API allows us to restrict an application to the resources contained in a resource set (this can also be done with the **attachrset** and **detachrset** system commands). In a partitioned environment you can run applications on separate partitions and therefore assign separate system resources to different applications. But even in a partitioned environment the use of resource sets might be useful. It can be used to isolate compatible workloads inside a single partition.

It can also be used to create overlapping resource sets, and therefore you can run applications with only partly common resources.

If a processor that is part of a resource set is removed, it is necessary to take that processor first out of the resource set before it can be removed from the partition. Otherwise, the processor will be busy and can't be removed from the partition.

11.2.1 The rset subroutines

In this section we describe the syntax of the rset subroutines.

ra_attachrset()

This subroutine is used to attach a process to a resource set. It attaches a process specified by the rstype and rsid parameters to a resource set that is specified by the rset parameter.

The process is identified by the process ID. A process ID value of RS_MYSELF indicates the attachment applies to the current process.

The following conditions must be met to successfully attach a process to a resource set:

- The resource set must contain processors that are available in the system.
- The calling process must either have root authority or have CAP_NUMA_ATTACH capability.
- The calling process must either have root authority or the same effective user ID as the target process.
- The target process must not contain any threads that have bindprocessor bindings to a processor.
- The resource set must be contained in (be a subset of) the target process' partition resource set.

If any of these conditions are not met, the attachment fails.

Once a process is attached to a resource set, the threads in the process will only run on processors contained in the resource set.

```
int ra_attachrset (rstype, rsid, rset, flags)
rstype_t rstype;
rsid_t rsid;
rsethandle_t rset;
unsigned int flags;
```

ra_detachrset

The ra_detachrset() subroutine is used to detach a work process specified by rstype and rsid from a resource set. The process is identified by the process ID. A process ID value of RS_MYSELF indicates the detach command applies to the current process.

```
int ra_detachrset (rstype, rsid, flags)
rstype_t rstype;
rsid_t rsid;
unsigned int flags;
```

ra_getrset

The ra_getrset() subroutine returns the resource set to which a specified process is attached. The process is identified by the process ID. A process ID value of RS_MYSELF indicates the resource set attached to the current process is requested.

```
int ra_getrset (rstype, rsid, flags, rset)
rstype_t rstype;
rsid_t rsid;
unsigned int flags;
rsethandle_t rset;
```

rs_alloc

The rs_alloc() subroutine allocates a resource set and initializes it according to the information specified by the flags parameter. The value of the flags parameter determines how the new resource set is initialized.

The handle for the new resource set is returned by the subroutine.

```
rsethandle_t rs_alloc (flags)
unsigned int flags;
```

The `flags` parameter specifies how the new resource set is initialized. It takes one of the following values, defined in rset.h:

RS_EMPTY (or 0 value)	The resource set is initialized to contain no resources.
RS_SYSTEM	The resource set is initialized to contain all available system resources.
RS_ALL	The resource set is initialized to contain all resources.
RS_PARTITION	The resource set is initialized to contain the resources in the caller's process partition resource set.

rs_discardname

The rs_discardname() subroutine removes the definition of the resource set from the system global repository. The resource set is identified by the namespace and rsname parameters. The specified resource set is removed from the registry, and can no longer be shared with other applications.

In order to be able to discard a name from the global repository, the calling process must have root authority or attachment privilege, and an effective user ID equal to that of the rsname parameter's creator.

```
int rs_discardname(namespace, rsname)
char *namespace, *rsname;
```

rs_free

The rs_free() subroutine frees a resource set identified by the rset parameter. The resource set must have been allocated by the rs_alloc() subroutine.

```
void rs_free(rset)
rsethandle_t rset;
```

rs_getinfo

The rs_getinfo() subroutine retrieves information about the resource set identified by the rset parameter. Depending on the value of the info_type parameter, the rs_getinfo() subroutine returns information about the number of available processors, the number of available memory pools, or the amount of available memory contained in the resource rset. The subroutine can also return global system information such as the maximum system detail level, the symmetric multiprocessor (SMP) and multiple chip module (MCM) system detail levels, and the maximum number of processor or memory pool resources in a resource set.

```
int rs_getinfo(rset, info_type, flags)
rsethandle_t rset;
rsinfo_t info_type;
unsigned int flags;
```

The info_type parameter specifies the type of information being requested. One of the following values (defined in rset.h) can be used:

R_NUMPROCS	The number of available processors in the resource set is returned.
R_NUMMEMPS	The number of available memory pools in the resource set is returned.
R_MEMSIZE	The amount of available memory (in MB) contained in the resource set is returned.
R_MAXSDL	The maximum system detail level of the system is returned.
R_MAXPROCS	The maximum number of processors that may be contained in a resource set is returned.
R_MAXMEMPS	The maximum number of memory pools that may be contained in a resource set is returned.
R_SMPSDL	The system detail level that corresponds to the traditional notion of an SMP is returned. A system detail level of 0 is returned if the hardware system does not provide system topology data.
R_MCMSDL	The system detail level that corresponds to resources packaged in an MCM is returned. A system detail level of

0 is returned if the hardware system does not have MCMs or does not provide system topology data.

rs_getnamedrset

The rs_getnamedrset() subroutine retrieves a resource set definition from the system registry. The namespace and rsname parameters identify the resource set to be retrieved. The rset parameter identifies where the retrieved resource set should be returned. The namespace and rsname parameters identify a previously registered resource set definition.

The calling process must have root authority or read access rights to the resource set definition in order to retrieve it.

The rset parameter must be allocated (using the rs_alloc() subroutine) prior to calling the rs_getnamedrset() subroutine.

```
int rs_getnamedrset (namespace, rsname, rset)
char *namespace, *rsname;
```

rs_op

The rs_op() subroutine performs the operation specified by the command parameter on resource set rset1, or both resource sets rset1 and rset2.

```
int rs_op (command, rset1, rset2, flags, id)
unsigned int command;
rsethandle_t rset1, rset2;
unsigned int flags;
unsigned int id;
```

The command parameter specifies the operation to apply to the resource sets identified by rset1 and rset2. One of the following values, defined in rset.h, can be used:

RS_UNION	The resources contained in either rset1 or rset2 are stored in rset2.
RS_INTERSECTION	The resources that are contained in both rset1 and rset2 are stored in rset2.
RS_EXCLUSION	The resources in rset1 that are also in rset2 are removed from rset2. On completion, rset2 contains all the resources that were contained in rset2 but were not contained in rset1.
RS_COPY	All resources in rset1 whose type is flags are stored in rset2. If rset1 contains no resources of this type, rset2 will be empty. The previous content of rset2 is lost, while the content of rset1 is unchanged.

RS_FIRST	The first resource whose type is flags is retrieved from rset1 and stored in rset2. If rset1 contains no resources of this type, rset2 will be empty.
RS_NEXT	The resource from rset1 whose type is flags and that follows the resource contained in rset2 is retrieved and stored in rset2. If no resource of the appropriate type follows the resource specified in rset2, rset2 will be empty.
RS_NEXT_WRAP	The resource from rset1 whose type is flags and that follows the resource contained in rset2 is retrieved and stored in rset2. If no resource of the appropriate type follows the resource specified in rset2, rset2 will contain the first resource of this type in rset1.
RS_ISEMPTY	Test if resource set rset1 is empty.
RS_ISEQUAL	Test if resource sets rset1 and rset2 are equal.
RS_ISCONTAINED	Test if all resources in resource set rset1 are also contained in resource set rset2.
RS_TESTRESOURCE	Test if the resource whose type is flags and index is id is contained in resource set rset1.
RS_ADDRESOURCE	Add the resource whose type is flags and index is id to resource set rset1.
RS_DELRESOURCE	Delete the resource whose type is flags and index is id from resource set rset1.

The `flags` parameter specifies the type of the resource that will be retrieved from rset1 or rset2. This parameter is constructed by logically ORing one or more of the following values, defined in rset.h:

R_PROCS	Processors
R_MEMPS	Memory pools
R_ALL_RESOURCES	Processors and memory pools

rs_registername

The rs_registername() subroutine registers, in the system resource registry (within the name space identified by namespace), the definition of the resource set identified by the rset handle. It does this by associating with it the name specified by the null-terminated string structure pointed to by rsname.

If rsname does not exist, the owner and group IDs of rsname are set to the caller's owner and group IDs, and the access control information for rsname is set according to the mode parameter.

If rsname already exists, its owner and group IDs and its access control information are left unchanged, and the mode parameter is ignored. This name can be shared with any applications to identify a dedicated resource set.

Using the command parameter, you can ask to overwrite or not to overwrite the rsname parameter's registration if it already exists in the global repository within the name space identified by namespace. If rsname already exists within the specified name space and the command parameter is set to not overwrite, an error is reported to the calling process.

```
int rs_registername(rset, namespace, rsname, mode, command)
rsethandle_t rset;
char *namespace, *rsname;
unsigned int mode, command;
```

The command parameter specifies whether the rsname parameter's registration should be overwritten if it already exists in the global repository. This parameter takes one of the following values, defined in rset.h:

RS_REDEFINE	The rsname parameter should be redefined if it already exists in the name space identified by namespace. In such a case, the calling process must have write access to rsname.
RS_DEFINE	The rsname parameter should not be redefined if it already exists in the name space identified by namespace. If this happens, an error is reported to the calling process

11.2.2 Working with the rset API

In this section we explain how to work with the rset API and give sample code for adding resources to or removing them from a resource set named test/rset.

We first create the resource set and then add or remove resources. Then we give examples of how you can delete a resource set with the rset API. We show how to compile[1] and run the examples and how to verify the results with the **lsrset** command.

Creating a resource set

The resource set is named rset and we create a new namespace test for it. We allocate an empty resource set and register it in the system global repository.

[1] Compiling these examples requires that the C compiler is installed and available on your AIX system.

Example 11-1 Creating an empty resource set

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

int main(int argc, char **argv) {
    rsethandle_t    rset;
    int             rc;

    /* allocate new rset */
    rset = rs_alloc(RS_EMPTY);

    /* register resource set in global system repository */
    rc = rs_registername(rset, "test", "rset", RS_IRUSR | RS_IWUSR, RS_DEFINE);
    if (rc != 0) perror("rs_registername()");

    return 0;
}
```

The following example shows how to compile and run the example. With the **lsrset** command we verify that we have created a resource set with no processors and no memory contained.

```
# cc -o rset_create rset_create.c
# rset_create
# lsrset -vr test/rset
T  Name               Owner   Group   Mode    CPU  Memory
a  test/rset          root    system  rw----    0       0
     CPU: <empty>
     MEM: <empty>
```

In this example we created an empty resource set. It is not possible to create an empty resource set using the **mkrset** command.

Note: Before you can use another example to create a resource, you either have to choose a different name for the resource set or delete it from the system global repository.

Adding a CPU to a resource set

In this example we create a resource set with one processor in it. We first allocate a new resource set that is empty. Then we add one processor to that resource set. Finally we register the resource set.

Example 11-2 Adding a processor to an new resource set

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

int main(int argc, char **argv) {
    rsethandle_t    rset;
    int             rc;

    /* allocate new rset */
    rset = rs_alloc(RS_EMPTY);

    /* add a CPU to the resource set */
    rc = rs_op(RS_ADDRESOURCE, rset, 0, R_PROCS, 0);
    if (rc != 0) perror("rs_op()");

    /* register resource set */
    rc = rs_registername(rset, "test", "rset", RS_IRUSR | RS_IWUSR, RS_DEFINE);
    if (rc != 0) perror("rs_registername()");

    return 0;
}
```

The following example shows how to compile and run the code. We use the **lsrset** command to verify the result:

```
# cc -o rset_create_cpu rset_create_cpu.c
# rset_create_cpu
# lsrset -vr test/rset
T  Name               Owner   Group   Mode    CPU  Memory
a  test/rset          root    system  rw----    1       0
     CPU: 0
     MEM: <empty>
```

We don’t necessarily have to register a new resource set. We can also modify an existing one. Example 11-3 shows how you can add a second processor to an existing resource set. We first read the resource set named test/rset. Then we add one processor and register the resource set again using the rs_registername() subroutine. For an existing resource set you have to give the RS_REDEFINE command to the rs_registername() subroutine.

Example 11-3   Adding a processor to an existing resource set

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

int main(int argc, char **argv) {
    rsethandle_t    rset;
    int             rc;

    /* allocate new rset */
    rset = rs_alloc(RS_EMPTY);

    /* read rset */
    rc = rs_getnamedrset("test", "rset", rset);
    if (rc != 0) perror("rs_getnamedrset()");
    /* add a CPU to the resource set */
    rc = rs_op(RS_ADDRESOURCE, rset, 0, R_PROCS, 1);
    if (rc != 0) perror("rs_op()");

    /* register resource set */
    rc = rs_registername(rset, "test", "rset", RS_IRUSR | RS_IWUSR,
RS_REDEFINE);
    if (rc != 0) perror("rs_registername()");

    return 0;
}
```

We first create a resource set with one processor using the **mkrset** command:

```
# mkrset -c 0 test/rset
1480-353 rset test/rset created.
# lsrset -vr test/rset
T  Name              Owner   Group   Mode    CPU  Memory
a  test/rset         root    system  rwr-r-    1       0
     CPU: 0
     MEM: <empty>
```

Then we compile the example and run it. With the **lsrset** command we can verify that our code has added a second processor to the resource set:

```
# cc -o rset_addcpu rset_addcpu.c
# rset_addcpu
# lsrset -vr test/rset
T  Name              Owner   Group   Mode    CPU  Memory
a  test/rset         root    system  rwr-r-    2       0
     CPU: 0-1
     MEM: <empty>
```

Removing a CPU from a resource set

Example 11-4 shows how you can remove a processor from the resource set using the rset API subroutines.

We read the resource set test/rset and remove one processor from the resource set with the rs_op() subroutine. Then we register the resource set in the system global repository using the rs_registername() subroutine with the RS_REDEFINE command.

Example 11-4 Removing a processor from a resource set

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

int main(int argc, char **argv) {
    rsethandle_t    rset;
    int             rc;
    /* allocate new rset */
    rset = rs_alloc(RS_EMPTY);

    /* read rset */
    rc = rs_getnamedrset("test", "rset", rset);
    if (rc != 0) perror("rs_getnamedrset()");

    /* remove a CPU from the resource set */
    rc = rs_op(RS_DELRESOURCE, rset, 0, R_PROCS, 0);
    if (rc != 0) perror("rs_op()");

    /* register resource set */
    rc = rs_registername(rset, "test", "rset", RS_IRUSR | RS_IWUSR,
RS_REDEFINE);
    if (rc != 0) perror("rs_registername()");

    return 0;
}
```

With the **mkrset** command we create a resource set with one processor and memory in it:

```
# mkrset -c 0 -m 0 test/rset
1480-353 rset test/rset created.
# lsrset -vr test/rset
T  Name               Owner   Group   Mode    CPU  Memory
a  test/rset          root    system  rwr-r-    1    2048
     CPU: 0
     MEM: 0
```

Then we compile our program and remove the processor from the resource set. With the **lsrset** command you can verify that there are no processors left and only the memory remains in the resource set:

```
# cc -o rset_remcpu rset_remcpu.c
# rset_remcpu
# lsrset -vr test/rset
T  Name               Owner   Group   Mode    CPU  Memory
a  test/rset          root    system  rwr-r-    0    2048
     CPU: <empty>
     MEM: 0
```

Adding memory to a resource set

Example 11-5 adds memory to a resource set. The memory can only be added in total, not in chunks.

We create a new resource set and add memory with the rs_op() subroutine. Then we register the resource set in the system global repository.

Example 11-5 Adding memory to a new resource set

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

int main(int argc, char **argv) {
    rsethandle_t    rset;
    int             rc;

    /* allocate new rset */
    rset = rs_alloc(RS_EMPTY);

    /* add memory to the resource set */
    rc = rs_op(RS_ADDRESOURCE, rset, 0, R_MEMPS, 0);
    if (rc != 0) perror("rs_op()");

    /* register resource set */
    rc = rs_registername(rset, "test", "rset", RS_IRUSR | RS_IWUSR, RS_DEFINE);
    if (rc != 0) perror("rs_registername()");

    return 0;
}
```

We first compile the code and run it:

```
# cc -o rset_create_mem rset_create_mem.c
# rset_create_mem
```

Then we verify the results with the **lsrset** command:

```
# lsrset -vr test/rset
T  Name              Owner    Group   Mode    CPU  Memory
a  test/rset         root     system  rw----    0    2048
     CPU: <empty>
     MEM: 0
```

We see a resource set with all the available memory and no processors contained.

We can also add memory to an existing resource set. The following code adds memory to a resource set we created before. We first read the resource set named test/rset and add memory with the rs_op() subroutine. Then we register the resource set in the system global repository. For an existing resource set you have to give the RS_REDEFINE command to the rs_registername() subroutine.

Example 11-6 Adding memory to an existing resource set

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

```
int main(int argc, char **argv) {
    rsethandle_t    rset;
    int             rc;

    /* allocate new rset */
    rset = rs_alloc(RS_EMPTY);

    /* read rset */
    rc = rs_getnamedrset("test", "rset", rset);
    if (rc != 0) perror("rs_getnamedrset()");

    /* add memory to the resource set */
    rc = rs_op(RS_ADDRESOURCE, rset, 0, R_MEMPS, 0);
    if (rc != 0) perror("rs_op()");

    /* register resource set */
    rc = rs_registername(rset, "test", "rset", RS_IRUSR | RS_IWUSR,
RS_REDEFINE);
    if (rc != 0) perror("rs_registername()");

    return 0;
}
```

We first create a resource set with the **mkrset** command. This resource set contains one processor.

```
# cc -o rset_addmem rset_addmem.c
# mkrset -c 0 test/rset
1480-353 rset test/rset created.
# lsrset -vr test/rset
T  Name               Owner   Group   Mode    CPU  Memory
a  test/rset          root    system  rwr-r-    1       0
     CPU: 0
     MEM: <empty>
```

Then we compile the example code and run it:

```
# cc -o rset_addmem rset_addmem.c
# rset_addmem
```

With the **lsrset** command we can verify that the resource set contains the processor and the available system memory:

```
# lsrset -vr test/rset
T  Name               Owner   Group   Mode    CPU  Memory
a  test/rset          root    system  rwr-r-    1    2048
     CPU: 0
     MEM: 0
```

Removing memory from a resource set

Example 11-7 shows how to remove memory from a resource set with the rset API.

We first read the resource set test/rset. Then we remove the memory from the resource set with the rs_op() subroutine. With the rs_registername() subroutine we register the resource set in the global repository. Again you have to use the **RS_REDEFINE** command with the rs_registername() subroutine.

Example 11-7 Removing memory from a resource set

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

int main(int argc, char **argv) {
    rsethandle_t   rset;
    int            rc;

    /* allocate new rset */
    rset = rs_alloc(RS_EMPTY);

    /* read rset */
    rc = rs_getnamedrset("test", "rset", rset);
    if (rc != 0) perror("rs_getnamedrset()");

    /* remove memory from the resource set */
    rc = rs_op(RS_DELRESOURCE, rset, 0, R_MEMPS, 0);
    if (rc != 0) perror("rs_op()");

    /* register resource set */
    rc = rs_registername(rset, "test", "rset", RS_IRUSR | RS_IWUSR,
RS_REDEFINE);
    if (rc != 0) perror("rs_registername()");

    return 0;
}
```

With the **mkrset** command we create a resource set with one processor and memory in it:

```
# mkrset -c 0 -m 0 test/rset
1480-353 rset test/rset created.
# lsrset -vr test/rset
T  Name               Owner   Group   Mode    CPU  Memory
a  test/rset          root    system  rwr-r-    1    2048
     CPU: 0
     MEM: 0
```

Then we compile our program and remove the memory from the resource set. With the **lsrset** command you can verify that only the processor is left in the resource set:

```
# cc -o rset_remmem rset_remmem.c
# rset_remmem
# lsrset -vr test/rset
T  Name               Owner   Group   Mode    CPU  Memory
a  test/rset          root    system  rwr-r-    1       0
     CPU: 0
     MEM: <empty>
```

Deleting a resource set

Finally we delete the resource set named test/rset from the system global repository. This can only be done if the resource set has been registered in the global repository with the rs_registername() subroutine.

The resource set can be removed from the global repository by calling the rs_discardname() subroutine and giving the name and namespace of the resource set as arguments.

Example 11-8 Deleting a resource set from the system global repository

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

int main(int argc, char **argv) {
    int         rc;

    rc = rs_discardname("test", "rset");
    if (rc != 0) perror("rs_discardname()");

    return 0;
}
```

With the **mkrset** command we create a resource set:

```
# mkrset -c 0 test/rset
1480-353 rset test/rset created.
# lsrset -vr test/rset
T  Name               Owner   Group   Mode    CPU  Memory
a  test/rset          root    system  rwr-r-    1       0
     CPU: 0
     MEM: <empty>
```

Then we compile our program and delete the resource set with it:

```
# cc -o rset_delete rset_delete.c
# rset_delete
# lsrset -vr test/rset
```

```
1480-157 'test/rset':  No such rset
T  Name              Owner   Group   Mode    CPU  Memory
```

Listing all available CPU numbers

Example 11-9 shows how you can list all available logical CPUs using the rset API.

Example 11-9 Listing all available logical CPU numbers

```
#include <stdio.h>
#include <sys/rset.h>

int main(int argc, char **argv) {
    rsethandle_t    rset;
    int             rc, maxprocs;
    unsigned int    i;

    /* allocate new rset with all system resources */
    rset = rs_alloc(RS_SYSTEM);

    /* get maximum number of cpus */
    maxprocs = rs_getinfo(NULL, R_MAXPROCS, 0);
    if (maxprocs < 0) perror("rs_getinfo()");

    /* step through all possible cpu numbers and check if it is in system */
    for(i = 0; i < maxprocs; i++) {
        rc = rs_op(RS_TESTRESOURCE, rset, 0, R_PROCS, i);
        if (rc < 0) perror("rs_op()");
        if (rc == 1) printf("cpu id: %u\n", i);
    }

    return 0;
}
```

If you compile this example and run it in a partition with two processors available, it will give you the following output:

```
# cc -O  block_cpu.c -o block_cpu
# rset_list_cpu
cpu id: 0
cpu id: 1
```

You see that we have two processors available in our partition and that they have the logical CPU numbers 0 and 1.

11.2.3 A DLPAR-aware application that is using the rset API

In this section we describe an example of a DLPAR-aware application. For creating this application we use the sample application shown in B.1, “Sample DLPAR-aware application using a signal handler” on page 433. We modify the code so it will create a resource set at startup. When the partition is reconfigured, the application will receive the SIGRECONFIG and start the signal handler. The signal handler is able to detect whether the modified resource is part of the resource set it is attached to.

If a processor that is taken out of the partition is contained in the resource set, the signal handler removes that processor from the resource set and adds a different processor to it. The application hereby is moved to another processor.

If an additional processor is added to the system, the signal handler removes the current processor from the resource set and puts the newly added processor into the resource set.

Reconfiguration application example

We create an application that reconfigures its resource set in case of a dynamic reconfiguration. We start with the example in Example B-4 on page 434 and modify the necessary parts.

First we have to modify the main program so it creates a resource set and attaches its process ID to it.

The include statement is best located near the other include statements.

Example 11-10 Modified main program

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

/*==================================================================*/
/*  Main application                                                */
/*==================================================================*/

/*==================================================================*/
/* Some Application that is registered for DR signals               */
/*                                                                  */
/* Detailed Description:                                            */
/*     This is a sample program that registers the signal handler   */
/*     function that will response to the SIGRECONFIG signal.       */
/*                                                                  */
/* Output: None                                                     */
/* Inputs: None                                                     */
/*==================================================================*/
int
```

```
main(int argc, char *argv[], char *envp[])
{
    int                 rc;
    struct sigaction    sigact_save, sigact;

    rsethandle_t        rset;
    rsid_t              rsid;
    unsigned int        cpu_id;

/* Start: register this application for a DR signal. */
    if ((rc = sigemptyset(&sigact.sa_mask)) != 0) {
        perror_msg("sigemptyset()", errno, __LINE__);
        exit(1);
    }
    if ((rc = sigemptyset(&sigact_save.sa_mask)) != 0) {
        perror_msg("sigemptyset()", errno, __LINE__);
        exit(1);
    }

    /* register the signal handler function dr_func. */
    sigact.sa_handler = dr_func;
    sigact.sa_flags |= SA_SIGINFO;


    if ((rc = sigaction(SIGRECONFIG, &sigact, &sigact_save)) != 0) { /* #A */
        perror_msg("sigaction()", errno, __LINE__);
        exit(1);
    }

/* define and register resource set */
    /* allocate empty resource set */
    if ((rset = rs_alloc(RS_EMPTY)) == 0) {;
        perror_msg("rs_alloc()", errno, __LINE__);
        exit(1);
    }

    /* get logical cpu id in partition */
    cpu_id = next_cpu(0);

    /* add cpu to resource set */
    if ((rc = rs_op(RS_ADDRESOURCE, rset, 0, R_PROCS, cpu_id)) != 0) {
        perror_msg("rs_op()", errno, __LINE__);
        exit(1);
    }

    /* attach process to resource set */
    rsid.at_pid = RS_MYSELF;
    if ((rc = ra_attachrset(R_PROCESS, rsid, rset , 0)) != 0) {
        perror_msg("rs_attachrset()", errno, __LINE__);
        exit(1);
    }

/* Finish:  registered the signal handler. */
    while (1) { /* #B */
        ;
        /* your application logic goes here. */
    }

    exit(0);

}
```

As we have already seen, the logical CPU numbers are not always consecutive. For creating a resource set with one processor in it we must find a logical CPU that is in the system. We do that with our own subroutine, shown in Example 11-11.

This subroutine is also used for finding a new logical CPU ID. It takes a logical CPU ID as argument and tries to find another logical CPU ID. If it can't find one, it returns the current logical CPU number as output, which means there is only one processor in the partition.

The subroutine allocates a resource set with all available system resources in it. Then it steps through all possible logical CPU IDs until it finds a CPU ID that is available and is different from the old one.

Example 11-11   New subroutine for finding the logical CPU number

```
/*=====================================================================*/
/* Description:  get new logical cpu number                            */
/*      This function searches for a new logical cpu number in the     */
/*      partition. If there is no other cpu the old number is          */
/*      returned as output.                                            */
/*                                                                     */
/* Output: new logical cpu number                                      */
/* Inputs: current logical cpu number                                  */
/*=====================================================================*/
unsigned int next_cpu(unsigned int old_id) {
    rsethandle_t     rset;
    rsid_t           rsid;

    unsigned int     new_id, i;
    int              rc, maxprocs;

    /* return old logical cpu number if no new cpu is found*/
    new_id = old_id;
    /* allocate resource set with all system resources */
    rset = rs_alloc(RS_SYSTEM);

    /* get maximum number of cpus */
    if ((maxprocs = rs_getinfo(NULL, R_MAXPROCS, 0)) < 0) {
          perror_msg("rs_getinfo()", errno, __LINE__);
          exit(1);
    }

    /* choose new cpu */
    i = 0;
    while ((new_id == old_id) && (i < maxprocs)) {
        if ((rc = rs_op(RS_TESTRESOURCE, rset, 0, R_PROCS, i)) < 0) {
            perror_msg("rs_op()", errno, __LINE__);
            exit(1);
        }
        if (rc == 1) {
            new_id = i;
        }
        i++;
    }

    return new_id;
}
```

The modified preRelease_cpu() subroutine detects when a processor in the resource set is removed from the partition. It then modifies the resource set, so the removed logical CPU is taken out of the resource set and another logical CPU in the partition is added to the resource set.

Example 11-12 The modified preRelease_cpu() subroutine

```
/*==================================================================*/
/* Cpu pre release phase                                            */
/*                                                                  */
/* Detailed Description:                                            */
/*      This should prepare the application for cpu resources being */
/*      removed.                                                    */
/*                                                                  */
/* Output: integer value indicating status of DR operation          */
/*     Values: DR_API_SUCCESS                                       */
/*             DR_API_FAIL                                          */
/* Inputs: None                                                     */
/*==================================================================*/
int preRelease_cpu(void)
{
    int l_rc = DR_API_SUCCESS;
```

```
rsethandle_t    rset;
rsid_t          rsid;
unsigned int    old_id, new_id;
int             rc;

fprintf(l_dbgFd, "*****Entered PreRelease_cpu*****\n");

/* Perform actions here */

/* get current logical cpu number */
old_id = dr_info.lcpu;
printf("logical CPU %d is removed\n", old_id);

/* get resource set our pid is attached to */
rset = rs_alloc(RS_EMPTY);
rsid.at_pid = RS_MYSELF;
if ((rc = ra_getrset(R_PROCESS, rsid, 0, rset)) < 0) {
    perror_msg("rs_getrset()", errno, __LINE__);
    exit(1);
}

/* test if cpu is in our resource set*/
if ((rc = rs_op(RS_TESTRESOURCE, rset, 0, R_PROCS, old_id)) < 0) {
    perror_msg("rs_op()", errno, __LINE__);
    exit(1);
}
/* cpu is in resource set and we have to move it */
if (rc == 1) {
    /* change cpu in resource set */
    new_id = next_cpu(old_id);
    /* remove old cpu from resource set */
    if ((rc = rs_op(RS_DELRESOURCE, rset, 0, R_PROCS, old_id)) != 0) {
        perror_msg("rs_op()", errno, __LINE__);
        exit(1);
    }
    /* put new cpu in resource set */
    if ((rc = rs_op(RS_ADDRESOURCE, rset, 0, R_PROCS, new_id)) != 0) {
        perror_msg("rs_op()", errno, __LINE__);
        exit(1);
    }
    /* reattach resource set to our pid */
    rsid.at_pid = RS_MYSELF;
    if ((rc = ra_attachrset(R_PROCESS, rsid, rset , 0)) != 0) {
        perror_msg("rs_attachrset()", errno, __LINE__);
        exit(1);
    }
}

return l_rc;
```

```
}
```

The changes we need when we add a processor to the partition go into the postAcquire_cpu() subroutine.

The process modifies the resource set each time a processor is added, so that the process runs on the newly added logical CPU.

Example 11-13 The modified postAcquire_cpu() subroutine

```
/*===================================================================*/
/* CPU post acquire phase                                            */
/*                                                                   */
/* Detailed Description:                                             */
/*      After a cpu addition, this function allows the application  */
/*      access to the new resources.  If the application was stopped, */
/*      iy should be restarted here.                                 */
/*                                                                   */
/*  Output: integer value indicating status of DR operation          */
/*     Values: DR_API_SUCCESS                                        */
/*             DR_API_FAIL                                           */
/* Inputs: None                                                      */
/*===================================================================*/
int postAcquire_cpu(void)
{
    int l_rc = DR_API_SUCCESS;

    rsethandle_t    rset;
    rsid_t          rsid;
    unsigned int    old_id, new_id;
    int             rc, maxprocs, i;

    fprintf(l_dbgFd, "*****Entered PostAcquire_cpu*****\n");

    /* Perform actions here */

    /* get logical cpu number to be added */
    new_id = dr_info.lcpu;
    printf("logical CPU %d is added\n", new_id);

    /* get resource set our pid is attached to */
    rset = rs_alloc(RS_EMPTY);
    rsid.at_pid = RS_MYSELF;
    if ((rc = ra_getrset(R_PROCESS, rsid, 0, rset)) < 0) {
        perror_msg("rs_getrset()", errno, __LINE__);
        exit(1);
    }

    /* get logical cpu number of current cpu in resource set */
    /* get maximum number of cpus */
    if ((maxprocs = rs_getinfo(NULL, R_MAXPROCS, 0)) < 0) {
          perror_msg("rs_getinfo()", errno, __LINE__);
          exit(1);
    }
    /* find current cpu */
    for (i = 0; i < maxprocs; i++) {
        if ((rc = rs_op(RS_TESTRESOURCE, rset, 0, R_PROCS, i)) < 0) {
            perror_msg("rs_op()", errno, __LINE__);
            exit(1);
        }
        if (rc == 1) {
            old_id = i;
        }
    }

    /* remove current cpu from our resource set */
    if ((rc = rs_op(RS_DELRESOURCE, rset, 0, R_PROCS, old_id)) != 0) {
        perror_msg("rs_op()", errno, __LINE__);
        exit(1);
    }
    /* add new cpu to our resource set */
    if ((rc = rs_op(RS_ADDRESOURCE, rset, 0, R_PROCS, new_id)) != 0) {
        perror_msg("rs_op()", errno, __LINE__);
        exit(1);
    }
    /* reattach resource set to our pid */
    rsid.at_pid = RS_MYSELF;
    if ((rc = ra_attachrset(R_PROCESS, rsid, rset , 0)) != 0) {
        perror_msg("rs_attachrset()", errno, __LINE__);
        exit(1);
    }

    return l_rc;
}
```

The sample code in Example B-4 on page 434 checks for CPU bindings and fails the dynamic reconfiguration in the check phase. Therefore, we have to modify the code so that the checkRelease_cpu() subroutine will return successfully, even if it detects a soft or hard pset dependency. Nevertheless, we keep the debug messages.

Example 11-14 The modified checkRelease_cpu() subroutine

```
/*====================================================================*/
/* Handles post release Error phase for cpu                           */
/*                                                                    */
/* Detailed Description:                                              */
/*      If an error should occur, this phase should handle undoing the */
/*      prerelease actions taken for cpu rmovals.                      */
/*                                                                     */
/* Output:  integer value indicating status of DR operation            */
/*     Values: DR_API_SUCCESS                                          */
/*           l DR_API_FAIL                                             */
/* Inputs: None                                                        */
/*=====================================================================*/
int checkRelease_cpu(void)
{
    int l_rc = DR_API_SUCCESS;

    fprintf(l_dbgFd, "*****Entered CheckRelease_cpu*****\n");

    /* Check for processor dependencies */
    if (dr_info.bindproc) {
        fprintf(l_dbgFd, "\t-- process has bindprocessor() dependency --\n");
        l_rc = DR_API_FAIL;
    }

    /* these to checks will not fail anymore because dependency is */
    /* resolved by changing resource set                           */
    if (dr_info.softpset) {
        /* print debug info but will not fail anymore */
        fprintf(l_dbgFd, "\t-- process has soft pset() dependency --\n");
        l_rc = DR_API_SUCCESS;
    }
    if (dr_info.hardpset) {
        /* print debug info but will not fail anymore */
        fprintf(l_dbgFd, "\t-- process has hard pset() dependency --\n");
        l_rc = DR_API_SUCCESS;
    }

    return l_rc;
}
```

After you have made the modifications, you can compile the code as described in B.1.1, “How to compile and run the application” on page 434.

```
# cc DLPAR_appl_autonomic.c -o DLPAR_appl_autonomic
```

A not DLPAR-aware example

We now show an example with an application that is not DLPAR-aware. We show how the DLPAR operation fails if the application does not adjust its resource sets when a dynamic reconfiguration is requested.

We modify the code in Example 11-2 on page 353 so that it is running an endless loop. It takes the logical CPU number that will be put in the resource set as a command line argument. The program does not terminate and thereby blocks the CPU contained in the resource set.

We do not register the resource set in the system global repository. Instead we use the process ID and the **-p** option with the **lsrset** command to verify that the resource set is properly attached to our process.

Example 11-15 Sample program that blocks a CPU

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

int main(int argc, char **argv) {
    rsethandle_t    rset;
    rsid_t          rsid;
    int             rc;
    unsigned int    cpu_id;

    /* get command line argument, terminate if not present */
    if (argc == 2) {
        cpu_id = atoi(argv[1]);
    } else {
        return 1;
    }

    /* allocate new rset */
    rset = rs_alloc(RS_EMPTY);

    /* add a CPU to the resource set */
    rc = rs_op(RS_ADDRESOURCE, rset, 0, R_PROCS, cpu_id);
    if (rc != 0) perror("rs_op()");

    /* attach process to resource set */
    rsid.at_pid = RS_MYSELF;
    rc = ra_attachrset(R_PROCESS, rsid, rset , 0);
    if (rc != 0) perror("rs_attachrset()");

    /* endless loop */
    while (1);

    return 0;
}
```

First we compile the program and then we start two instances of it. We attach one instance to logical CPU 0 and the other process to logical CPU 1.

```
lpar08# cc block_cpu.c -o block_cpu
lpar08# block_cpu 0 &
lpar08# block_cpu 1 &
```

You can either start the program with the ampersand (&) to start it in the background, as we do, or you can start it in a separate terminal window.

We get the process IDs of our processes with the **ps** command:

```
lpar08# ps -ef | grep block_cpu
    root 204988 319670 112 14:30:13  pts/1  0:24 ./block_cpu 1
    root 233712 360634 115 14:28:13  pts/2  2:24 ./block_cpu 0
```

We check the configuration of the attached resource sets with the **lsrset** command:

```
lpar08# lsrset -vp 233712
Effective rset:  1 CPU,  0 Memory
     CPU: 0
     MEM: <empty>

lpar08# lsrset -vp 204988
Effective rset:  1 CPU,  0 Memory
     CPU: 1
     MEM: <empty>
```

Now we initiate a DLPAR operation to take one processor out of the partition:

```
hmc$ chhwres -m ITSO_p690 -p lpar08 -r cpu -o r -q 1 -W 4 -d 4
aixErr: true
HMCERRV3DLPAR020: r operation for cpu has completed, but only 0 out of 1 were
successful.
The AIX command is:
drmgr -r -c cpu -w 5 -d 4

The AIX standard output is:
...
... omitted lines ...
...
The AIX standard error is:

0930-047 Detected DR Unsafe application.  DR operation failed.

         Consult AIX error log for more information


The return code is 0. The AIX return code is 1.
```

We can also take a look at the information in the system error log:

```
lpar08# errpt -a
---------------------------------------------------------------------------
LABEL:          CPU_DEALLOC_ABORTED
IDENTIFIER:     4056F04C

Date/Time:       Wed Aug 13 14:32:54 CDT
Sequence Number: 232
Machine Id:      0021768A4C00
Node Id:         lpar08
Class:           S
Type:            TEMP
Resource Name:   proc4

Description
CPU DEALLOCATION ABORTED

Probable Causes
SOFTWARE PROGRAM

Detail Data
DEALLOCATION ABORTED CAUSE
0000 0002
DEALLOCATION ABORTED DATA
0000 0000 0003 20BC
Duplicates
Number of duplicates
           1
Time of first duplicate
Wed Aug 13 14:32:53 CDT
Time of last duplicate
Wed Aug 13 14:32:54 CDT
---------------------------------------------------------------------------
```

This operation fails because both available processors are blocked by our running programs. The system cannot free one processor for removal, and so the requested dynamic reconfiguration fails.

DLPAR-aware example

This example shows how the DLPAR-aware application *autonomically* reconfigures the resource set it is attached to. The application detects that a processor, which is contained in its resource set, is about to be removed from the system. The application reconfigures the resource set so that it contains another processor. The processor is freed and the requested DLPAR operation succeeds.

We also show how the DLPAR-aware application reconfigures the resource set when a processor is added to the system. We designed the application so that it changes the processor in its resource set to the newly added processor.

```
lpar08# rset_list_cpu
cpu id: 0
cpu id: 1
```

We start the CPU blocking process from Example 11-15 on page 370 again. We start the process on logical CPU 0:

```
lpar08# block_cpu 0
lpar08# ps -ef | grep block_cpu
    root 245858 319670  62 16:29:07  pts/1  0:08 block_cpu 0
```

We verify that the resource set contains logical CPU 0 using the **lsrset** command with the **-p** option:

```
lpar08# lsrset -vp 245858
Effective rset:  1 CPU,  0 Memory
     CPU: 0
     MEM: <empty>
```

Then we start the DPLAR-aware application and get the process ID using the **ps** command:

```
lpar08# DLPAR_appl_autonomic
lpar08# ps -ef | grep DLPAR_appl_autonomic
    root 311444 360634 120 16:32:20  pts/2  0:06 DLPAR_appl_autonomic
```

The DLPAR-aware application always uses the lowest available logical CPU number that is greater than 0. That means the process is running on logical CPU 1. We can verify this using the **lsrset** command with the **-p** option:

```
lpar08# lsrset -vp 311444
Effective rset:  1 CPU,  0 Memory
     CPU: 1
     MEM: <empty>
```

We now initiate a DLPAR operation on the HMC to take one processor out of the partition.

```
hmc$ chhwres -m ITSO_p690 -p lpar08 -r cpu -o r -q 1 -W 4 -d 4
```

The requested operation completes successfully, which means that the DLPAR-aware application has reconfigured its resource set and is no longer using logical CPU 1. We verify this with the **lsrset** command:

```
lpar08# lsrset -vp 311444
Effective rset:  1 CPU,  0 Memory
     CPU: 0
     MEM: <empty>
```

We verify that only one processor is left in the partition:

```
lpar08# lsdev -Cc processor
proc3 Available 00-03 Processor
```

The blocking process is still running on logical CPU 0. Now both processes are running on the remaining processor:

```
lpar08# lsrset -vp 245858
Effective rset:  1 CPU,  0 Memory
     CPU: 0
     MEM: <empty>
```

Now we add the processor to the partition again. As already explained, the application is designed so that it switches its resource set over to the newly added processor.

```
hmc$ chhwres -m ITSO_p690 -p lpar08 -r cpu -o a -q 1 -W 4 -d 4
```

We use the **lsrset** command to verify that the DLPAR-aware process has reconfigured its resource set. Now both processes are running on separate processors again.

```
lpar08# lsrset -vp 245858
Effective rset:  1 CPU,  0 Memory
     CPU: 0
     MEM: <empty>

lpar08# lsrset -vp 311444
Effective rset:  1 CPU,  0 Memory
     CPU: 1
     MEM: <empty>
```

12

Autonomic applications

The dynamic reconfiguration application framework (DRAF) introduced with AIX 5L Version 5.2 poses a new programming paradigm defined as follows:

- ► Application resources must be controllable..
- ► Query the currently available and usable resources dynamically..
- ► Self-adjust the resource consumption..
- ► Performance scalability is based on the given resources..

Carefully designed and properly implemented DLPAR-aware applications can satisfy all of these requirements. We call such applications *autonomic applications*, since they can adjust themselves in the autonomic computing environment where resources are dynamically changed based on various business and management reasons.

In this chapter, we explain how the autonomic applications should be designed and discuss what types of applications can be autonomic. An autonomic application example is also provided to demonstrate how it solves the challenging requirements posed by the paradigm.

12.1 Design considerations

When designing autonomic applications, the following should be carefully considered.

- Application resources must be controllable.

  An autonomic application must provide some ways to control (or set) its resource usage, such as the number of threads or amount of memory. For example, it may define the default resource usage in the program code, but should provide a configuration file that defines the resource usage.

  If your application is hard-coded to allocate huge amounts of memory in one chunk, it is hard to modify it to be an autonomic application.

- Query the currently available and usable resources dynamically.

  An autonomic application must dynamically query the currently available and usable resources upon program startup. It must also query the available and usable resources whenever the operating system notifies the application of the resource change.

  In other words, an autonomic application should be aware of the environment it is running on.

- Self-adjust the resource consumption.

  An autonomic application must adjust its resource usage based on the currently available resources. Because resource change events can asynchronously happen unrelated to the current application program status, it needs to validate the proposed resource change, and then decide whether to accept or reject the change. The rejection of a resource change should be made if the proposed resources are inappropriate to maintain the application's performance.

  In other words, an autonomic application must have a resource usage policy such that it can dynamically react to resource changes. The policy should not be hard-coded in the application program. It is ideally defined by an external configuration file.

- Performance scalability is based on the given resources.

  An autonomic application's performance should scale with the given resources.

12.2 Possible autonomic applications

In the current DLPAR implementation on AIX, three kinds of resources, CPU, memory, and I/O slots, are dynamically reconfigured. As for the I/O slot DLPAR operations, since human intervention would be required in most cases, it is difficult to design autonomic applications that can react to DLPAR operations for I/O resources. Therefore, we only consider CPU and memory resources as target of DLPAR operations in the following discussion.

As for the DLPAR CPU operations, it is not so difficult to design and code autonomic applications that can react to a CPU resource change. In fact, your application is usually not aware of, or does not have to be aware of, the currently available processors in a partition. The kernel scheduler does a good job of dispatching user threads on to available processors. The difficult part is to make your application sensitive to the number of available processors.

For example, your application creates a number of threads depending on the number of available processors. It is not difficult to increase or decrease the number of threads in your application to react to DLPAR CPU operations. However, the increase or decrease of threads does not necessarily mean that your application has the performance scalability based on the available processors. Sometimes your threads are just sitting in memory without any tasks to do, or sleeping in the scheduler queue because they are waiting for other threads' completion, release of the mutex locks, or there simply is not enough memory.

As for the DLPAR memory operations, it is quite easy for applications to allocate additional memory in case of DLPAR memory addition. However, it is difficult (sometimes impossible) for applications to unallocate already allocated memory in case of DLPAR memory removal. For example, an autonomic application allocates 1 GB of memory as buffer, then allocates an additional 3 GB of memory upon a DLPAR memory addition operation.

The application starts to use all the memory, in total 4 GB, as application buffer area. Then it is requested to unallocate 2 GB of memory from its buffer since the administrator has requested a DLPAR memory removal operation. At this moment you, the programmer of that autonomic application, have the following three actions to choose from:

- Reject

  The application can reject the memory removal request so that the entire DLPAR removal operation would fail. This is most likely selected while the application is using the buffer, since the buffer is filled with data of the transaction or computation.

- Suspend

  If the requirement of the application performance or transaction response is not severe, and if the amount of data in the buffer is not huge, it is technically possible to store the data into a temporary file, then relinquish the buffer so that the application can *suspend* the use of memory. If the application gets memory later, it can *resume* by reading the data from the temporary file to the buffer.

- Abandon

  The application can abandon the data in the buffer and unallocate memory, if the data can be reproduced later and the application performance or transaction response is not severe.

The autonomic application must provide a policy that selects either of the above actions when memory removal is requested. It would be better if the policy can be set in a configuration file and the application can read the configuration file when requested.

12.3 A sample autonomic application

We developed a sample autonomic application (the complete program source code written in the C language is listed in Example F-1 on page 490). Basically, the subject of the sample program is summarized as follows:

- Prepare random data in shared memory segments.
- Seek the character string `IBM` from the random data in the shared memory segments.
- Print how many times `IBM` is found.

The program is a 64-bit multithreaded application. It is assumed that readers are familiar with the multithreaded programming model. Therefore, primitive data synchronization facilities in multithreaded programming, such as mutex, are not explained in this section.

12.3.1 Tasks

To solve the challenging requirements posed by the autonomic computing environment, the sample program introduces conceptual objects, called *tasks*.

Tasks are used in this program only in order to logically divide the program's subject. Each task can run concurrently and independently from the other tasks. A task is composed of a user thread and shared memory segments.

A task is represented by a task status table entry, which has either of the following values (each of the table elements is initialized to ST_READY in setup()):

```
enum st {
    ST_READY,
    ST_RUNNING,
    ST_STOPPING,
    ST_DONE
} *st = NULL;
```

The table entry values have the following meaning:

ST_READY	A task in the corresponding task entry can be started.
ST_RUNNING	A task in the corresponding task entry is started and a thread is running for this task.
ST_STOPPING	A task in the corresponding task entry is stopping. No new thread is allowed to run for this task.
ST_DONE	A task in the corresponding task is done. No new thread is allowed to run for this task.

The table provides the task status, and in addition is used as a communication method between the task scheduler function and worker threads.

12.3.2 Task scheduler

The sample program has a function called *task scheduler* for adjusting its resource usage. This function is called periodically and in case of DLPAR operations.

If there is a task status table entry with the value of ST_READY, the task scheduler function selects it and then creates a worker thread as follows:

```
pthread_create(&ch, NULL, worker, (void *) no);
```

The worker thread determines for which task it is working from the argument `no`, which is cast into the variable task as follows:

```
int    task = *((int *) arg);
```

In order to process tasks, the task scheduler tries to create as many threads as it can based on the given resources (number of processors and amount of allowable memory) until all the tasks are completed. If resources are added before the process terminates, the task scheduler creates additional threads to process tasks. If allocated resources are reduced, the scheduler reduces the number of worker threads to satisfy the new resource condition. The tasks for which stopped worker threads were processing return to the ST_READY state. The scheduler creates threads to process those tasks later when enough resources are available.

12.3.3 Pseudo program algorithm

Once invoked, the program does the following:

1. The program sets the process-wide signal mask as follows:

   ```
   sigemptyset(&set);
   sigaddset(&set, SIGINT);
   sigaddset(&set, SIGRECONFIG);
   pthread_sigmask(SIG_BLOCK, &set, NULL);
   ```

   This signal mask is used for all the threads in this process except for the signal watcher thread explained later in this section.

2. If a configuration file name is specified, the main() function calls setup() to read it. Otherwise, it assumes that the following parameters are specified:

   ```
   mem_for_os=1024
   task_per_cpu=2
   mem_per_task=256
   mem_allowable_diff=512
   max_task=8
   ```

   The configuration file can be specified as the first command line argument as follows:

   ```
   $ ./seek_IBM ./seek_IBM.conf
   ```

3. The setup() function checks the number of online processors and the amount of currently available memory[1], as follows:

   ```
   cpu_units = sysconf(_SC_NPROCESSORS_ONLN);

   if ((fp = popen("lsattr -EOl mem0", "r")) == NULL ||
       fgets(buf, 20, fp) == NULL ||
       fgets(buf, 20, fp) == NULL || (p = strchr(buf, ':')) == NULL) {
       printf("Can't get memory size from lsattr -EOl mem0.\n");
   }
   *p = 0;
   mem_units = atoi(buf);
   ```

   Next, the setup() function initializes the task scheduler table (st[]) as follows:

   ```
   if (stmax <= 0 || (st = (enum st *) malloc(sizeof(int) * stmax)) == NULL) {
       printf("Can't allocate task table.\n");
       exit(1);
   }
   for (c = 0; c < stmax; c++) {
       st[c] = ST_READY;
   }
   ```

   Finally, the setup() function calls the reconfig_check() function to check if the current available resources (number of processors and amount of memory) are well balanced.

   If reconfig_check() returns less than 0, the program exits.

[1] The amount of real memory can also be checked using sysconf(_SC_AIX_REALMEM).

```
if ((c = reconfig_check(cpu_units, mem_units)) < 0) {
    printf("CPU and memory are out of balance. "
        "(mem_units=%d MB, cpu_units=%d).\n", mem_units, cpu_units);
    exit(3);
}
```

4. The reconfig_check() function implements the application resource usage policy. This policy is controlled by several parameters set in the configuration file, or by the default values explained in step 2.
   a. The function calculates required memory for each *task* as follows:

   ```
   int mem_for_task = mem_per_task * task_per_cpu * cpu;
   ```

   b. If the amount of currently available memory (mem) is smaller than the amount of memory size reserved for the operating system (mem_for_os), then reconfig_check() returns -1, as follows:

   ```
   if (mem < mem_for_os) {
       prt_log
         ("** Hey! You will have only %d MB of memory, that is not less\n"
         "** than your setting in mem_for_os=%d\n.\n", mem, mem_for_os);
         return -1;
   } else if ....
   ```

   c. If either of the following conditions is satisfied, the function returns -1. These checks are introduced to the program to give a margin to memory usage:
      - Margin condition #1

      ```
      } else if (mem_for_task + mem_for_os - mem_allowable_diff > mem) {
          prt_log
          ("** Hey! You will have %d MB of memory that is much less than\n"
          "** required memory size, %d MB.  Allowable difference between\n"
          "** these two values is less than %d MB.\n"
          , mem, mem_for_task + mem_for_os, mem_allowable_diff);
          return -1;
      } else if ....
      ```

      - Margin condition #2

      ```
      } else if (mem_for_task + mem_for_os + mem_allowable_diff < mem) {
          prt_log
          ("** Hey! You will have %d MB of memory that is much more than\n"
          "** required memory size, %d MB.  Allowable difference between\n"
          "** these two values is less than %d MB.\n"
          , mem, mem_for_task + mem_for_os, mem_allowable_diff);
          return -1;
      }
      ```

   As for an example of how the reconfig_check() calculates the parameters, let us assume that the following default parameter values are set:

   ```
   mem_for_os=1024
   ```

```
task_per_cpu=2
mem_per_task=256
mem_allowable_diff=512
max_task=8
```

If the program is invoked on an AIX partition with four processors and 3 GB of memory assigned, the function calculates:

```
mem_for_task = mem_per_task * task_per_cpu * cpu
    = 256 * 2 * 4 = 2 GB
```

Because mem_for_task (2 GB) plus mem_for_os (1 GB) is exactly 3 GB, the function determines that the balance of the number of processors and memory size in this partition is adequate. If the assigned memory size is less than 2.5 GB (mem_for_task + mem_for_os - mem_allowable_diff) or larger than 3.5 GB (mem_for_task + mem_for_os + mem_allowable_diff), the function determines that the balance is inadequate.

d. If the function evaluates all the above conditions as false, it calculates the following two values:

```
int memslots = (mem - mem_for_os) / mem_per_task;
int cpuslots = task_per_cpu * cpu;
```

These two values specify the number of available slots that can run tasks.

The function always returns the smaller value of memslots or cpuslots. For example, if the default parameter values are set and the program is invoked on an AIX partition with four processors and 3.25 GB of memory assigned, the function calculates:

```
memslots = (3.25 GB - 1 GB) / 256 MB = 9
cpuslots = 2 * 4 = 8
```

Since memslots is larger than cpuslots, the function returns 8.

5. The main() function creates a user thread, called *signal watcher thread*, that executes the sigwatcher() function only:

```
pthread_create(&ch, NULL, sigwatcher, NULL);
```

Once executed, the signal watcher thread sets a signal mask for the SIGRECONFIG and SIGINT signals and sleeps in the highlighted *while* loop shown in Example 12-1. Because the process-wide signal mask ignores these signals, only this thread in the process is responsible for handling these signals.

Example 12-1 sigwatcher

```
void *sigwatcher(void *arg)
{
    int rc;
    sigset_t newset;
    sigemptyset(&newset);
    sigaddset(&newset, SIGRECONFIG);
    sigaddset(&newset, SIGINT);
    while (!sigwait(&newset, &rc)) {
        fprintf(stderr, "sigwatcher: rc=%d\n", rc);
        if (rc == SIGINT) {
            pthread_mutex_lock(&st_l);
            runflag = SEEK_IBM_STOP;
            pthread_mutex_unlock(&st_l);
            task_scheduler(0);
            break;
        } else if (rc == SIGRECONFIG) {
            reconfig_manager();
        }
    }
    pthread_exit(NULL);
}
```

If SIGINT is sent to the process, sigwatcher() is awakened and calls task_scheduluer(0). If SIGRECONFIG is sent to the process, sigwatcher() is awaken to calls reconfig_manager().

In case of DLPAR events, the signal handler function, reconfig_manager(), queries the proposed resource change using the dr_recofing() system call, then validates the current resource usage and instructs how many tasks should be increased or decreased using the task_scheduler() function. Because this function is lengthy, see Example F-1 on page 490 for the full listing of this function.

6. The main() function calls the task_scheduler() function with the promoter c, which is returned from setup(); therefore, c has the smaller value of cpuslots and memslots:

   ```
   task_scheduler(c);
   ```

   The task_scheduler() function sets the task status table (st[]) according to the passed parameter n, as shown in Example 12-2 on page 383.

Example 12-2 task_scheduler

```
void task_scheduler(int n)
{
    pthread_t ch;
    int c, d = 0;
    int *no;

    pthread_mutex_lock(&st_l);
    for (c = 0; c < stmax; c++) {
        if (st[c] == ST_RUNNING) {
            d++;
```

```
                continue;
            }
            if (st[c] == ST_READY && d < n && runflag == SEEK_IBM_RUN) {
                no = malloc(sizeof(int));
                *no = c;
                pthread_create(&ch, NULL, worker, (void *) no);
                pthread_detach(ch);
                prt_log("create thread %d %d\n", *no, d);
                st[c] = ST_RUNNING;
                d++;
            } else if (st[c] == ST_RUNNING && d >= n) {
                st[c] = ST_STOPPING;
            }
        }
        pthread_mutex_unlock(&st_l);
}
```

If the condition highlighted in Example 12-2, the task_scheduler() function creates a new thread, called *worker*, as follows:

```
pthread_create(&ch, NULL, worker, (void *) no);
```

7. A worker thread executes only one function, worker(). Once executed, the function does the following:

   a. It opens the /dev/urandom device to get a random seed number:

   ```
   if ((fp = fopen("/dev/urandom", "r")) == NULL) {
       prt_log("Can't open /dev/urandom.\n");
   }
   fread(&seed, 1, 4, fp);
   fclose(fp);
   ```

   b. It allocates a 256 MB shared memory segment:

   ```
   if ((id = shmget(IPC_PRIVATE, ONE_SEG, IPC_CREAT | 0600)) == 0) {
       prt_log("%d: shmget error.\n", task);
       pthread_exit(NULL);
   }
   prt_log("%d: id=%d\n", task, id);
   if ((buf = shmat(id, 0, 0)) == NULL) {
       prt_log("%d: shmat error.\n", task);
       pthread_exit(NULL);
   }
   ```

   c. It fills the shared memory segment with the random data produced by random().

   d. It scans the shared memory segment to seek the character string `IBM`. If it is found, the ibmc counter increments.

   e. Once it finishes the scan, it updates the task table and the IBM_found variable as follows:

```
st[task] = ST_DONE;
if (IBM_found < 0) {
    IBM_found = ibmc;
} else {
    IBM_found += ibmc;
}
```

f. Before it exits, it unallocates the shared segment.

8. The main() function then loops in the infinity *while* loop shown in Example 12-3. Every five seconds, it prints how many times `IBM` is found in the random data and calls task_scheduler() to recalculate whether the current resource usage is adequate.

Example 12-3 An infinity while loop in main()

```
while (1) {
    while (1) {
        if (!(c = threadnum())) {
            break;
        }
        if (IBM_found < 0) {
            prt_log("+ CPU:%d, memory:%d [MB], threads: %d, tasks: %d\n"
                , cpu_units, mem_units, c, n);
        } else {
            prt_log("+ CPU:%d, memory:%d [MB], threads: %d, tasks: %d,"
                " \"IBM\" found %d times so far.\n"
                , cpu_units, mem_units, c, n, IBM_found);
        }
        sleep(5);
    }
    if ((n = taskleft()) <= 0) {
        break;
    }
    task_scheduler(reconfig_check(cpu_units, mem_units));
}
```

Part 4

Appendixes

A

Test environment

In this appendix, we describe the following:

- Hardware configuration
- Network configuration
- Partition configuration
- The relationship between physical and AIX location codes

We excerpted the necessary information from *IBM @server pSeries 690 Service Guide*, SA38-0589 to make this redbook a comprehensive guide:

- pSeries 670- and pSeries 690-dependent information
- AIX location codes

Hardware configuration

This redbook was created and verified with several tests using the following test environment:

- A pSeries 690 (standard processor feature)
- Two multichip modules (MCMs) (16 processors)
- 64 GB of memory (4 x 16 GB memory)
- Two 7040-61D I/O drawers
- A media drawer containing a 4 mm tape drive and a DVD-RAM drive
- A Hardware Management Console (HMC) (FC 7315)

Media drawer SCSI connection

The front SCSI bus of the media drawer is connected to the first SCSI port of the SCSI adapter, which is inserted into the tenth PCI slot of the left-half planner of the first I/O drawer, as shown in Figure A-1.

The physical location code of this adapter is U1.9-P1-I10 (see Table A-3 on page 396), and the first SCSI port is identified as P1-I10-Z1.

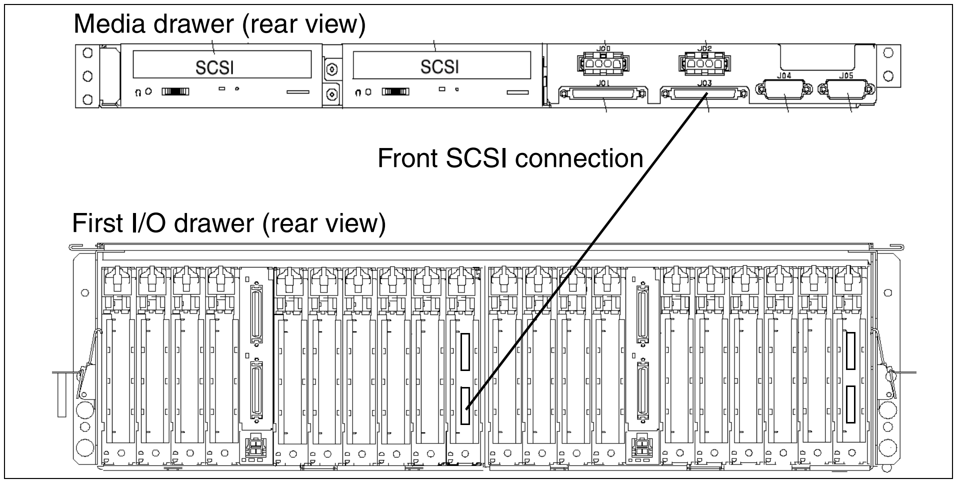


Figure A-1 Media drawer SCSI connection

Network configuration

In our test environment, the HMC and all partitions are connected to the single 100 Mbps Ethernet network, as shown in Figure A-2.

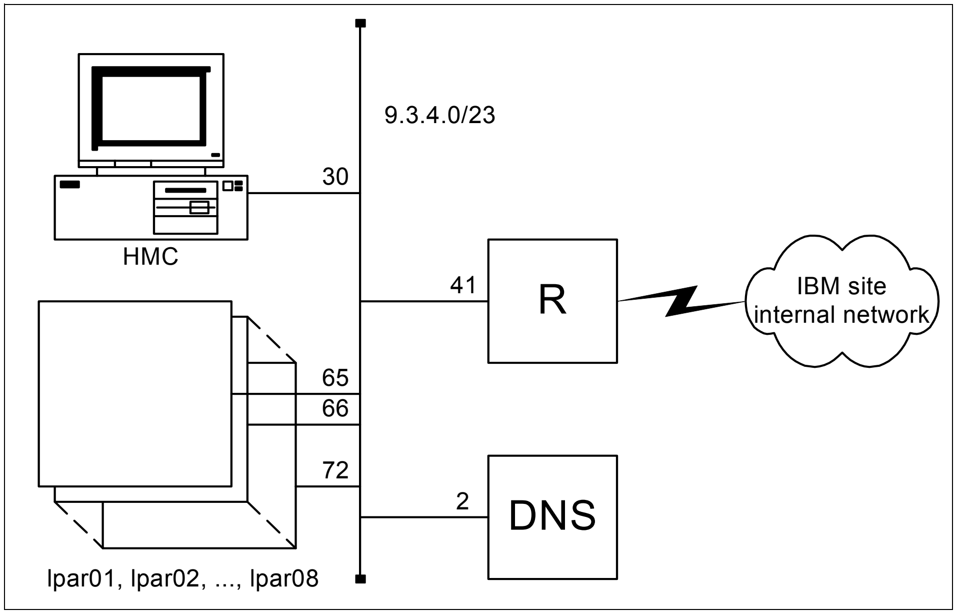


Figure A-2 Physical network configuration in the test environment

The network configuration is as follows:

- The subnetwork address is 9.3.4.0 (subnet mask: 255.255.254.0).
- There is only one IP router, configured at 9.3.4.41.
- The primary DNS server for the DNS domain, itsc.austin.ibm.com, is configured at 9.3.4.2.

Name resolution

The host name resolution is done using the primary DNS server configured at 9.3.4.2, regardless of whether it is on the HMC or partitions. Example A-1 on page 392 lists IP addresses and host names defined on the HMC and all partitions so that the reverse name resolution always returns FQDN. For example, IP address 9.3.4.65 always returns FQDN, lpar01.itsc.austin.ibm.com, in our test environment:

```
# host 9.3.4.65
lpar01.itsc.austin.ibm.com is 9.3.4.65
```

Example: A-1 Local host table in our test environment

```
9.3.4.30       itsohmc.itsc.austin.ibm.com itsohmc
9.3.4.65       lpar01.itsc.austin.ibm.com lpar01
9.3.4.66       lpar02.itsc.austin.ibm.com lpar02
9.3.4.67       lpar03.itsc.austin.ibm.com lpar03
9.3.4.68       lpar04.itsc.austin.ibm.com lpar04
9.3.4.69       lpar05.itsc.austin.ibm.com lpar05
9.3.4.70       lpar06.itsc.austin.ibm.com lpar06
9.3.4.71       lpar07.itsc.austin.ibm.com lpar07
9.3.4.72       lpar08.itsc.austin.ibm.com lpar08
```

Partition configuration

The partitions are configured as follows:

- Processor
  - Minimum: 1
  - Desired: 2
  - Maximum: 4
- Memory
  - Minimum: 1 GB
  - Desired: 4 GB
  - Maximum: 8 GB
- Adapter
  - Required: The adapters that are specified with an asterisk (*) in Table A-1 on page 393 and Table A-2 on page 394.
  - Desired: The adapters that are not specified with an asterisk (*) in Table A-1 on page 393 and Table A-2 on page 394.

> **Note:** The SCSI adapter that is inserted into P1.9-P1-I10, is defined as desired on all the partitions.

The inserted adapters shown in Table A-1 on page 393 and Table A-2 on page 394 are identified with the following adapter identifiers:

A-F	FC 4962 Ethernet/LAN Encryption 10/100 Base T
4-Y	FC 6203 Dual-Channel Ultra3 SCSI

Table A-1 Partition configuration: I/O drawer 1

Partition name	Physical location code	Adapter identifier
lpar01*	U1.9-P1-I1	A-F
lpar01	U1.9-P1-I2	A-F
lpar01	U1.9-P1-I3	
lpar01	U1.9-P1-I4	
lpar02	U1.9-P1-I5	A-F
lpar02	U1.9-P1-I6	A-F
lpar02	U1.9-P1-I7	
lpar02	U1.9-P1-I8	
lpar02	U1.9-P1-I9	
N/A	**U1.9-P1-I10**	**4-Y**
lpar01*	U1.9-P1-Z1	Internal Ultra3 SCSI
lpar02*	U1.9-P1-Z2	Internal Ultra3 SCSI
lpar03*	U1.9-P2-I1	A-F
lpar03	U1.9-P2-I2	A-F
lpar03	U1.9-P2-I3	
lpar03	U1.9-P2-I4	
lpar04*	U1.9-P2-I5	A-F
lpar04	U1.9-P2-I6	A-F
lpar04	U1.9-P2-I7	
lpar04	U1.9-P2-I8	
lpar04	U1.9-P2-I9	
lpar04	U1.9-P2-I10	4-Y
lpar03*	U1.9-P2-Z1	Internal Ultra3 SCSI
lpar04*	U1.9-P2-Z2	Internal Ultra3 SCSI

Table A-2 Partition configuration: I/O drawer 2

Partition name	Physical location code	Adapter identifier
lpar05*	U1.5-P1-I1	A-F
lpar05	U1.5-P1-I2	A-F
lpar05	U1.5-P1-I3	
lpar05	U1.5-P1-I4	
lpar06	U1.5-P1-I5	A-F
lpar06	U1.5-P1-I6	A-F
lpar06	U1.5-P1-I7	
lpar06	U1.5-P1-I8	
lpar06	U1.5-P1-I9	
lpar06	U1.5-P1-I10	4-Y
lpar05*	U1.5-P1-Z1	Internal Ultra3 SCSI
lpar06*	U1.5-P1-Z2	Internal Ultra3 SCSI
lpar07*	U1.5-P2-I1	A-F
lpar07	U1.5-P2-I2	A-F
lpar07	U1.5-P2-I3	
lpar07	U1.5-P2-I4	
lpar08*	U1.5-P2-I5	A-F
lpar08	U1.5-P2-I6	A-F
lpar08	U1.5-P2-I7	
lpar08	U1.5-P2-I8	
lpar08	U1.5-P2-I9	
lpar08	U1.5-P2-I10	4-Y
lpar07*	U1.5-P2-Z1	Internal Ultra3 SCSI
lpar08*	U1.5-P2-Z2	Internal Ultra3 SCSI

Note: This adapter configuration is solely used to provide the maximum number of partitions, eight, in our test environment. Therefore, it is presented as information only and may not always be suitable in your environment.

The relationship between physical and AIX location codes

The physical location codes for pSeries systems are different from the AIX location codes. There are several ways to identify the relationship between the AIX and physical location codes, including the following:

- Using AIX commands
- Using the physloc field identifier of lsdev (AIX 5L Version 5.2)

Using AIX commands

If the partition is up and running, you can identify the relationship using the following method.

If the **lsdev** command is used for identifying the AIX location code of a component, you will see an output similar to the following:

```
# lsdev -Cl ent0
ent0 Available 3F-08 10/100 Mbps Ethernet PCI Adapter II (1410ff01)
```

In this example, the AIX location code of ent0 is 3F-08.

When you look at the configuration using the **lscfg** command, you will see an output similar to the following example, where the physical location code is U1.9-P2-I1/E1:

```
# lscfg -vl ent0
  ent0             U1.9-P2-I1/E1  10/100 Mbps Ethernet PCI Adapter II
(1410ff01)

      10/100 Mbps Ethernet PCI Adapter II:
        Part Number.................09P5023
        FRU Number..................09P5023
        EC Level....................H10971A
        Manufacture ID..............YL1021
        Network Address.............0002556AAC19
        ROM Level.(alterable).......SCU001
        Product Specific.(Z0).......A5204205
        Device Specific.(YL)........U1.9-P2-I1/E1
```

Using the physloc field identifier of lsdev (AIX 5L Version 5.2)

In AIX 5L Version 5.2, an enhancement was made to the **lsdev** command to display physical location codes. As shown in the following example, if you specify the format identifier `physloc` with the `-F` option, the command displays the physical location code of the specified device:

```
# lsdev -Cl ent0 -F physloc
U1.9-P2-I1/E1
```

Using the service guide

If the partition is not running, you can identify the relationship using the reference table provided in the service guide for your partioning-capable pSeries server. For the pSeries 670 and pSeries 690, see the following section.

pSeries 670- and pSeries 690-dependent information

Table A-3 shows the relationship only in the first I/O drawer of pSeries 670 and pSeries 690.

Table A-3 Physical and AIX location codes: I/O drawer 1

Field Replacement Unit (FRU) name	Physical location codes	AIX location code
I/O Subsystem 1 Chassis and Midplane Card	(MT/M Serial #) U1.9	
I/O subsystem left I/O backplane assembly	**U1.9-P1**	
I/O subsystem left I/O backplane assembly VPD	U1.9-P1-N1	
EADS 1 - PCI contoller	U1.9-P1	2U-58, 2U-5A, 2U-5C, 2U-5E
PCI Slot 1 Content	U1.9-P1-I1	2V-08 to 2V-0F or 2W-xx or 2X-xx
PCI Slot 2 Content	U1.9-P1-I2	2Y-08 to 2Y-0F or 2Z-xx or 2a-xx
PCI Slot 3 Content	U1.9-P1-I3	2b-08 to 2b-0F or 2c-xx or 2d-xx
PCI Slot 4 Content	U1.9-P1-I4	2e-08 to 2e-0F or 2f-xx or 2g-xx
EADS 2 - PCI Controller	U1.9-P1	2j-58, 2j-5A, 2j-5E
PCI Slot 5 Content	U1.9-P1-I5	2k-08 to 2k-0F or 2m-xx or 2n-xx
PCI Slot 6 Content	U1.9-P1-I6	2p-08 to 2p-0F or 2q-xx or 2r-xx

Field Replacement Unit (FRU) name	**Physical location codes**	**AIX location code**
PCI Slot 7 Content	U1.9-P1-I7	2v-08 to 2v-0F or 2w-xx or 2x-xx
EADS 3 - PCI Controller	U1.9-P1	30-58, 30-5A, 30-5E
PCI Slot 8 Content	U1.9-P1-I8	31-08 to 31-0F or 32-xx or 32-xx
PCI Slot 9 Content	U1.9-P1-I9	34-08 to 34-0F or 35-xx or 36-xx
PCI Slot 10 Content	U1.9-P1-I10	3A-08 to 3A-0F or 3B-xx or 3C-xx
I/O Subsystem Right I/O backplane assembly	**U1.9-P2**	
I/O Subsystem Right I/O backplane assembly VPD	U1.9-P2-N1	
EADS 1 - PCI Controller	U1.9-P2	3E-58, 3E-5A, 3E-5C, 3E-5E
PCI Slot 1 Content	U1.9-P2-I1	3F-08 to 3F-0F or 3G-xx or 3H-xx
PCI Slot 2 Content	U1.9-P2-I2	3J-08 to 3J-0F or 3K-xx or 3L-xx
PCI Slot 3 Content	U1.9-P2-I3	3M-08 to 3M-0F or 3N-xx or 3P-xx
PCI Slot 4 Content	U1.9-P2-I4	3Q-08 to 3Q-0F or 3R-xx or 3S-xx
EADS 2 - PCI Controller	U1.9-P2	3U-58, 3U-5A, 3U-5E
PCI Slot 5 Content	U1.9-P2-I5	3V-08 to 3V-0F or 3W-xx or 3X-xx
PCI Slot 6 Content	U1.9-P2-I6	3Y-08 to 3Y-0F or 3Z-xx or 3a-xx
PCI Slot 7 Content	U1.9-P2-I7	3e-08 to 3e-0F or 3f-xx or 3a-xx
EADS 3 - PCI Controller	U1.9-P2	3j-58, 3j-5A, 3j-5E
PCI Slot 8 Content	U1.9-P2-I8	3k-08 to 3k-0F or 3m-xx or 3n-xx
PCI Slot 9 Content	U1.9-P2-I9	3p-08 to 3p-0F or 3q-xx or 3r-xx
PCI Slot 10 Content	U1.9-P2-I10	3v-08 to 3v-0F or 3w-xx or 3x-xx
I/O Subsystem SCSI controller 1 on P1	**U1.9-P1/Z1**	**2s-08**
I/O Subsystem SCSI controller 2 on P1	**U1.9-P1/Z2**	**37-08**
I/O Subsystem SCSI controller 1 on P2	**U1.9-P2/Z1**	**3b-08**

Field Replacement Unit (FRU) name	Physical location codes	AIX location code
I/O Subsystem SCSI controller 2 on P2	**U1.9-P2/Z2**	**3s-08**
DASD 4 Pack Cage and card (1)	U1.9-P3	
DASD 4 Pack Cage and card (1) VPD	U1.9-P3-N1	
SCSI DASD 1 hdisk at ID 8 connected to controller 2 on P2	U1.9-P2/Z2-A8	3s-08-00-8,0
SCSI DASD 2 hdisk at ID 9 connected to controller 2 on P2	U1.9-P2/Z2-A9	3s-08-00-9,0
SCSI DASD 3 hdisk at ID A connected to controller 2 on P2	U1.9-P2/Z2-Aa	3s-08-00-10,0
SCSI DASD 4 hdisk at ID A connected to controller 2 on P2	U1.9-P2/Z2-Ab	3s-08-00-11,0
SCSI Enclosure Services SES connected to controller 2 on P2	U1.9-P2/Z2-Af	3s-08-00-15,0
DASD 4 Pack Cage and card (2)	U1.9-P4	
DASD 4 Pack Cage and card (2) VPD	U1.9-P4-N1	
SCSI DASD 1 hdisk at ID 8 connected to controller 1 on P2	U1.9-P2/Z1-A8	3b-08-00-8,0
SCSI DASD 2 hdisk at ID 9 connected to controller 1 on P2	U1.9-P2/Z1-A9	3b-08-00-9,0
SCSI DASD 3 hdisk at ID A connected to controller 1 on P2	U1.9-P2/Z1-Aa	3b-08-00-10,0

Field Replacement Unit (FRU) name	Physical location codes	AIX location code
SCSI DASD 4 hdisk at ID B connected to controller 1 on P2	U1.9-P2/Z1-Ab	3b-08-00-11,0
SCSI Enclosure Services SES connected to controller 1 on P2	U1.9-P2/Z1-Af	3b-08-00-15,0
DASD 4 Pack Cage and card (3)	U1.9-P5	
DASD 4 Pack Cage and card (3) VPD	U1.9-P5-N1	
SCSI DASD 1 hdisk at ID 8 connected to controller 2 on P1	U1.9-P1/Z2-A8	37-08-00-8,0
SCSI DASD 2 hdisk at ID 9 connected to controller 2 on P1	U1.9-P1/Z2-A9	37-08-00-9,0
SCSI DASD 3 hdisk at ID A connected to controller 2 on P1	U1.9-P1/Z2-Aa	37-08-00-10,0
SCSI DASD 4 hdisk at ID B connected to controller 2 on P1	U1.9-P1/Z2-Ab	37-08-00-11,0
SCSI Enclosure Services SES connected to controller 2 on P1	U1.9-P1/Z2-Af	37-08-00-15,0
DASD 4 Pack Cage and card (4)	U1.9-P6	
DASD 4 Pack Cage and card (4) VPD	U1.9-P6-N1	
SCSI DASD 1 hdisk at ID 8 connected to controller 1 on P1	U1.9-P1/Z1-A8	2s-08-00-8,0
SCSI DASD 2 hdisk at ID 9 connected to controller 1 on P1	U1.9-P1/Z1-A9	2s-08-00-9,0

Field Replacement Unit (FRU) name	Physical location codes	AIX location code
SCSI DASD 3 hdisk at ID A connected to controller 1 on P1	U1.9-P1/Z1-Aa	2s-08-00-10,0
SCSI DASD 4 hdisk at ID B connected to controller 1 on P1	U1.9-P1/Z1-Ab	2s-08-00-11,0
SCSI Enclosure Services SES connected to controller 1 on P1	U1.9-P1/Z1-Af	2s-08-00-15,0

Table A-4 Physical and AIX location codes: I/O drawer 2

FRU Name	Physical location codes	AIX location code
I/O Subsystem 2 Chassis and Midplane Card	(MT/M Serial #) U1.5	
I/O Subsystem Left I/O backplane assembly	**U1.5-P1**	
I/O Subsystem Left I/O backplane assembly VPD	U1.5-P1-N1	
EADS 1 - PCI Controller	U1.5-P1	40-58, 40-5A, 40-5C, 40-5E
PCI Slot 1 Content	U1.5-P1-I1	41-08 to 41-0F or 42-xx or 43-xx
PCI Slot 2 Content	U1.5-P1-I2	44-08 to 44-0F or 45-xx or 47-xx
PCI Slot 3 Content	U1.5-P1-I3	47-08 to 47-0F or 48-xx or 49-xx
PCI Slot 4 Content	U1.5-P1-I4	4A-08 to 4A-0F or 4B-xx or 4C-xx
EADS 2 - PCI Controller	U1.5-P1	4E-58, 4E-5A, 4E-5E
PCI Slot 5 Content	U1.5-P1-I5	4F-08 to 4F-0F or 4G-xx or 4H-xx
PCI Slot 6 Content	U1.5-P1-I6	4J-08 to 4J-0F or 4K-xx or 4L-xx
PCI Slot 7 Content	U1.5-P1-I7	4Q-08 to 4Q-0F or 4R-xx or 4S-xx
EADS 3 - PCI Controller	U1.5-P1	4U-58, 4U-5A, 4U-5E
PCI Slot 8 Content	U1.5-P1-I8	4V-08 to 4V-0F or 4W-xx or 4X-xx
PCI Slot 9 Content	U1.5-P1-I9	4Y-08 to 4Y-0F or 4Z-xx or 4a-xx

FRU Name	Physical location codes	AIX location code
PCI Slot 10 Content	U1.5-P1-I10	4e-08 to 4e-0F or 4f-xx or 4g-xx
I/O Subsystem Right I/O backplane assembly	**U1.5-P2**	
I/O Subsystem Right I/O backplane assembly VPD	U1.5-P2-N1	
EADS 1 - PCI Controller	U1.5-P2	4j-58, 4j-5A, 4j-5C, 4j-5E
PCI Slot 1 Content	U1.5-P2-I1	4k-08 to 4k-0F or 4m-xx or 4n-xx
PCI Slot 2 Content	U1.5-P2-I2	4p-08 to 4p-0F or 4q-xx or 4r-xx
PCI Slot 3 Content	U1.5-P2-I3	4s-08 to 4s-0F or 4t-xx or 4t-xx
PCI Slot 4 Content	U1.5-P2-I4	4v-08 to 4v-0F or 4t-xx or 4u-xx
EADS 2 - PCI Controller	U1.5-P2	50-58, 50-5A, 50-5E
PCI Slot 5 Content	U1.5-P2-I5	51-08 to 51-0F or 52-xx or 53-xx
PCI Slot 6 Content	U1.5-P2-I6	54-08 to 54-0F or 55-xx or 56-xx
PCI Slot 7 Content	U1.5-P2-I7	5A-08 to 5A-0F or 5B-xx or 5C-xx
EADS 3 - PCI Controller	U1.5-P2	5E-58, 5E-5A, 5E-5E
PCI Slot 8 Content	U1.5-P2-I8	5F-08 to 5F-0F or 5G-xx or 5H-xx
PCI Slot 9 Content	U1.5-P2-I9	5J-08 to 5J-0F or 5K-xx or 5L-xx
PCI Slot 10 Content	U1.5-P2-I10	5Q-08 to 5Q-0F or 5R-xx or 5S-xx
I/O Subsystem SCSI controller 1 on P1	**U1.5-P1/Z1**	4M-08
I/O Subsystem SCSI controller 2 on P1	**U1.5-P1/Z2**	4b-08
I/O Subsystem SCSI controller 1 on P2	**U1.5-P2/Z1**	**57-08**
I/O Subsystem SCSI controller 2 on P2	**U1.5-P2/Z2**	**5M-08**
DASD 4 Pack Cage and card (1)	U1.5-P3	
DASD 4 Pack Cage and card (1) VPD	U1.5-P3-N1	

FRU Name	Physical location codes	AIX location code
SCSI DASD 1 hdisk at ID 8 connected to controller 2 on P2	U1.5-P2/Z2-A8	5M-08-00-8,0
SCSI DASD 2 hdisk at ID 9 connected to controller 2 on P2	U1.5-P2/Z2-A9	5M-08-00-9,0
SCSI DASD 3 hdisk at ID A connected to controller 2 on P2	U1.5-P2/Z2-Aa	5M-08-00-10,0
SCSI DASD 4 hdisk at ID A connected to controller 2 on P2	U1.5-P2/Z2-Ab	5M-08-00-11,0
SCSI Enclosure Services SES connected to controller 2 on P2	U1.5-P2/Z2-Af	5M-08-00-15,0
DASD 4 Pack Cage and card (2)	U1.5-P4	
DASD 4 Pack Cage and card (2) VPD	U1.5-P4-N1	
SCSI DASD 1 hdisk at ID 8 connected to controller 1 on P2	U1.5-P2/Z1-A8	57-08-00-8,0
SCSI DASD 2 hdisk at ID 9 connected to controller 1 on P2	U1.5-P2/Z1-A9	57-08-00-9,0
SCSI DASD 3 hdisk at ID A connected to controller 1 on P2	U1.5-P2/Z1-Aa	57-08-00-10,0
SCSI DASD 4 hdisk at ID B connected to controller 1 on P2	U1.5-P2/Z1-Ab	57-08-00-11,0
SCSI Enclosure Services SES connected to controller 1 on P2	U1.5-P2/Z1-Af	57-08-00-15,0
DASD 4 Pack Cage and card (3)	U1.5-P5	

FRU Name	Physical location codes	AIX location code
DASD 4 Pack Cage and card (3) VPD	U1.5-P5-N1	
SCSI DASD 1 hdisk at ID 8 connected to controller 2 on P1	U1.5-P1/Z2-A8	4b-80-00-8,0
SCSI DASD 2 hdisk at ID 9 connected to controller 2 on P1	U1.5-P1/Z2-A9	4b-80-00-9,0
SCSI DASD 3 hdisk at ID A connected to controller 2 on P1	U1.5-P1/Z2-Aa	4b-80-00-10,0
SCSI DASD 4 hdisk at ID B connected to controller 2 on P1	U1.5-P1/Z2-Ab	4b-80-00-11,0
SCSI Enclosure Services SES connected to controller 2 on P1	U1.5-P1/Z2-Af	4b-80-00-15,0
DASD 4 Pack Cage and card (4)	U1.5-P6	
DASD 4 Pack Cage and card (4) VPD	U1.5-P6-N1	
SCSI DASD 1 hdisk at ID 8 connected to controller 1 on P1	U1.5-P1/Z1-A8	4M-08-00-8,0
SCSI DASD 2 hdisk at ID 9 connected to controller 1 on P1	U1.5-P1/Z1-A9	4M-08-00-9,0
SCSI DASD 3 hdisk at ID A connected to controller 1 on P1	U1.5-P1/Z1-Aa	4M-08-00-10,0
SCSI DASD 4 hdisk at ID B connected to controller 1 on P1	U1.5-P1/Z1-Ab	4M-08-00-11,0
SCSI Enclosure Services SES connected to controller 1 on P1	U1.5-P1/Z1-Af	4M-08-00-15,0

AIX location codes

The basic formats of the AIX location codes are as follows:

- For non-SCSI devices/drives:

  AB-CD-EF-GH
- For SCSI devices/drives:

  AB-CD-EF-G,H

Non-SCSI devices

For planars (backplanes), cards, and non-SCSI devices, the location code is defined as follows:

```
AB-CD-EF-GH
|  |  |  |
|  |  |  Device/FRU/Port ID
|  |  Connector ID
|  devfunc Number, Adapter Number or Physical Location
Bus Type or PCI Parent Bus
```

Where:

- The AB value identifies a bus type or PCI parent bus as assigned by the firmware.
- The CD value identifies the adapter number, devfunc number, or physical location. The devfunc number is defined as the PCI device number times 8, plus the function number.
- The EF value identifies a connector.
- The GH value identifies a port, address, device, or FRU.

Adapters and cards are identified only with AB-CD.

The possible values for CD depend on the adapter and card. For pluggable PCI adapters and cards, CD is the device's devfunc number (the PCI device number times 8, plus the function number). The C and D are characters in the range of 0 to 9 and A to F (hex numbers). The location codes, therefore, uniquely identify multiple adapters on individual PCI cards.

EF is the connector ID used to identify the adapter's connector to which a resource is attached.

GH is used to identify a port, device, or FRU; for example:

- For asynchronous devices, GH defines the port on the fanout box. The values are 00 to 15.
- For a diskette drive, H identifies either diskette drive 1 or 2. G is always 0.
- For all other devices, GH is equal to 00.

For an integrated adapter, EF-GH is the same as the definition for a pluggable adapter. For example, the location code for a diskette drive is 01-D1-00-00. A second diskette drive is 01-D1-00-01.

SCSI devices

For SCSI devices, the location code is defined as follows:

```
AB-CD-EF-G,H
|  |  |  |  |
|  |  |  |  Logical Unit address of the SCSI Device
|  |  |  Control Unit Address of the SCSI Device
|  |  Connector ID
|  devfunc Number, Adapter Number or Physical Location
Bus Type or PCI Parent Bus
```

Where:

- AB-CD-EF are the same as non-SCSI devices.
- G defines the control unit address of the device. Values of 0 to 15 are valid.
- H defines the logical unit address of the device. Values of 0 to 255 are valid.

A bus location code is also generated as 00-XXXXXXXX, where XXXXXXXX is equivalent to the node's unit address.

Examples of physical location codes and AIX location codes are as follows:

- PCI adapter in first I/O subsystem, slot 1 (primary rack):
  - Physical location code U1.9-P1-I1
  - AIX location code 2V-08
- PCI adapter in fifth I/O subsystem, slot 1 (secondary rack):
  - Location code U2.1-P1-I1
  - AIX location code 8V-08

B

Dynamic logical partitioning program templates

This appendix provides the following sample dynamic logical partitioning (DLPAR) program templates. These templates are provided as samples to integrate DLPAR operations into your applications.

- Perl template
- Korn shell template
- Sample DLPAR-aware application using a signal handler

For further information about DLPAR program and its usage, see Chapter 3, “Dynamic logical partitioning” on page 53.

Note: The source files for these examples are available. See Appendix G, “Additional material” on page 503 for information about how to download example source files.

General information

The templates included in this appendix are as follows:

- "Perl template" on page 409
- "Korn shell template" on page 421
- "Sample DLPAR-aware application using a signal handler" on page 433

If you can access the source code of your application, see the last template example to make your applications DLPAR-aware.

If not, you can choose your favorite programming language from Perl, Korn shell, and the C language in order to integrate DLPAR operations into your applications. The first two templates are slightly modified from the files originally installed in the /usr/samples/dr/scripts directory on AIX 5L Version 5.2 to add a debug output facility and insert comments for readability. The original template files are included in the bos.adt.samples fileset, as shown in the following example:

```
# lslpp -w /usr/samples/dr/scripts/IBM_template.*
  File                                        Fileset               Type
  ----------------------------------------------------------------------------
  /usr/samples/dr/scripts/IBM_template.c      bos.adt.samples       File
  /usr/samples/dr/scripts/IBM_template.pl     bos.adt.samples       File
  /usr/samples/dr/scripts/IBM_template.sh     bos.adt.samples       File
# lslpp -L bos.adt.samples
  Fileset                      Level  State  Type  Description (Uninstaller)
  ----------------------------------------------------------------------------
  bos.adt.samples            5.2.0.0    C     F    Base Operating System
Samples
```

Note: Some scripts in the /usr/samples/dr/scripts directory are provided to demonstrate error situations. For example, IBM_XYZ_fail_dr_2.sh generates the AIX error log shown in Example B-1 in a CPU removal DLPAR event. You should carefully read the *readme* file in this directory, before registering these scripts.

Example: B-1 DR_UNSAFE_PROCESS

```
LABEL:          DR_UNSAFE_PROCESS
IDENTIFIER:     0E2A04B4

Date/Time:       Fri Nov 15 11:04:17 CST
Sequence Number: 114
Machine Id:      0021768A4C00
Node Id:         lpar01
Class:           S
Type:            INFO
Resource Name:   SYSPROC

Description
DR Unsafe application

Detail Data
Process ID
      30499421762355200
```

These templates implement all the necessary command syntaxes explained in Table 3-7 on page 76 that define the DLPAR script syntax.

Perl template

Example B-2 provides the Perl version of the DLPAR script template. It can be used by system administrators as a starting point to integrate DLPAR operations into your applications.

The script is added a file handle, DBG, to be used to print debug information to the debug file, /tmp/dr_IBM_template.pl.dbg. The debug information is very helpful when you have to debug your DLPAR script, because the script should not print any undefined name-value pairs to the standard out.

To display the debug information sent to the file, type the following command:

```
$ tail -f /tmp/dr_IBM_template.pl.dbg
```

Example: B-2 DLPAR script template: Perl

```
#!/usr/bin/perl

# (C) COPYRIGHT International Business Machines Corp. 2000, 2002
# All Rights Reserved
# Licensed Materials - Property of IBM
#
# US Government Users Restricted Rights - Use, duplication or
# disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
#
# FILE NAME: dr_IBM_template.pl
#
# FILE DESCRIPTION:
# This perl script will provide a template for DLPAR script developers
# to develop their custom perl scripts.
#
# It is the basic test script.  It implements all the commands of DLPAR
# script and for all them returns success.  It identifies itself with
# distinct details.

#----------------------------------------------------------------------
# GLOBAL OBJECTS
#----------------------------------------------------------------------

# This hash contains the required commands specified in the DRAF.
# The values assigned to each command is irrelevant.  The script
# converts the text into a number for quick accessing
%DR_script_commands = (
    scriptinfo => 1,
    register => 2,
    usage => 3,
    checkrelease => 4,
    prerelease => 5,
    postrelease => 6,
    undoprerelease => 7,
    checkacquire => 8,
    preacquire => 9,
    postacquire => 10,
    undopreacquire => 11
);
# This hash contains data used by the scriptinfo command.  It lists
# required information about the script
%SCRIPT_DATA = (
    SCRIPT_INFO => “AIX “,
    SCRIPT_VERSION => “1”,
    SCRIPT_VENDOR => “IBM Corp.”,
    SCRIPT_TIMEOUT => 5
);
# This hash contains the resources for which the script will  register.
# In this case, this script wants to register for DR operations that
# involve memory and cpu events.
%REGISTER_DATA = (
    CPU_RESOURCE => “cpu”,
    MEM_RESOURCE => “mem”
);
# This hash contains usage descriptions for each possible resource.
%USAGE_DATA = (
    CPU_USAGE => “Testing DLPAR on CPU resource”,
    MEM_USAGE => “Testing DLPAR on MEM resource”
);

#-----------------------------------------------------------------------
# Helper Functions
#-----------------------------------------------------------------------

#=======================================================================
#  Name:  str_to_cmd
#
#  Description: converts a string to a command value
#
#  Input: command string
#
#  Output: logically mapped command value
#
#  Return Code:  None
#=======================================================================
sub str_to_cmd {

    $s_cmd = $_[0];
    $DR_script_commands{$s_cmd};
}


#-----------------------------------------------------------------------
# Required DRAF commands
#-----------------------------------------------------------------------

#=======================================================================
#  Name:  process_scriptinfo
#
#  Description:  returns information about the script
#
#  Input: none
#
#  Output: name-value pairs
#
#  Return Code:  0 = success
#                1 = failure
#=======================================================================
sub process_scriptinfo {

    print “DR_SCRIPTINFO=$SCRIPT_DATA{SCRIPT_INFO}\n”;
    print “DR_VERSION=$SCRIPT_DATA{SCRIPT_VERSION}\n”;
    print “DR_DATE=19042002\n”;
    print “DR_VENDOR=$SCRIPT_DATA{SCRIPT_VENDOR}\n”;
    print “DR_TIMEOUT=$SCRIPT_DATA{SCRIPT_TIMEOUT}\n”;
    0;
}

#==============================================================================
#  Name:  process_scriptinfo
#
#  Description:  returns information about the script
#
#  Input: none
#
#  Output: name-value pairs
#
#  Return Code:  0 = success
#                1 = failure
#==============================================================================
sub process_register {

    foreach $key ( keys %REGISTER_DATA){
        print “DR_RESOURCE=$REGISTER_DATA{$key}\n”;
    }
    0;
}

#==============================================================================
#  Name:  process_usage
#
#  Description:  returns usage information about the script
#
#  Input: resource
#
#  Output: name-value pairs
#
#  Return Code:  0 = success
#                1 = failure
#==============================================================================
sub process_usage {

    $l_rc = 0;
    $res = $_[0];

    USE_SWITCH: {
        if ($res eq $REGISTER_DATA{CPU_RESOURCE}) {
            print “DR_USAGE=$USAGE_DATA{CPU_USAGE}\n”;
            last USE_SWITCH;
        }
        if ($res eq $REGISTER_DATA{MEM_RESOURCE}) {
            print “DR_USAGE=$USAGE_DATA{MEM_USAGE}\n”;
            last USE_SWITCH;
        }
        print “DR_ERROR=script does not use resource $res\n”;
        $l_rc = 1;
    }
    return $l_rc;
}

#========================================================================
#  Name:  process_checkrelease
#
#  Description:  verifies a resource can be removed without compromises
#                the application.
#
#  Input: resource
#
#  Output: Any DR_ERROR=description or DR_LOG_DEBUG=description variables
#
#  Return Code:  0 = success
#                1 = failure
#========================================================================
sub process_checkrelease {

    $l_rc = 0;
    $res = $_[0];

    print DBG “-- start checkrelease phase --\n”;
    foreach $key (sort keys %ENV) {
        if ($key =~ /^DR_/) {
            print DBG $key, ‘=’, $ENV{$key}, “\n”;
        }
    }

    USE_SWITCH: {
        if ($res eq $REGISTER_DATA{CPU_RESOURCE}) {
            # perform all cpu related checks here and determine
            # if resource remove can proceed.
            last USE_SWITCH;
        }
        if ($res eq $REGISTER_DATA{MEM_RESOURCE}) {
            # perform all memu related checks here and determine
            # if resource remove can proceed.
            last USE_SWITCH;
        }
        print “DR_ERROR=script does not use resource $res\n”;
        $l_rc = 1;
    }

    print DBG “-- end checkrelease phase --\n”;
    return $l_rc;
}

#========================================================================
#  Name:  process_prerelease
#
#  Description:  Prepares for the resource to be removed
#
#  Input: resource
#
#  Output: Any DR_ERROR=description or DR_LOG_DEBUG=description variables
#
#  Return Code:  0 = success
#                1 = failure
#========================================================================
sub process_prerelease {

    $l_rc = 0;
    $res = $_[0];

    print DBG "-- start prerelease phase --\n";
    foreach $key (sort keys %ENV) {
        if ($key =~ /^DR_/) {
            print DBG $key, '=', $ENV{$key}, "\n";
        }
    }

    # before we allow DR manager to proceed, we can do any prerelease
    # actions here.  For instance, we could send a signal from here
    # and wait for the application to take some action.

    # resource specific actions
    USE_SWITCH: {
        if ($res eq $REGISTER_DATA{CPU_RESOURCE}) {
            # release any cpu bindings, etc. here if the resource
            # is being used by the application.
            last USE_SWITCH;
        }
        if ($res eq $REGISTER_DATA{MEM_RESOURCE}) {
            # release any application hold over memory, etc, that
            # is being removed.
            last USE_SWITCH;
        }
        print "DR_ERROR=script does not use resource $res\n";
        $l_rc = 1;
    }

    print DBG "-- end prerelease phase --\n";
    return $l_rc;
}

#======================================================================
#  Name:  process_undoprerelease
#
#  Description:  Invoked to undo any changes done by the prerelease
#                command.
#
#  Input: resource
#
#  Output: Any DR_ERROR=description or DR_LOG_DEBUG=description variables
#
#  Return Code:  0 = success
#                1 = failure
#======================================================================
sub process_undoprerelease {

    $l_rc = 0;
    $res = $_[0];
    # perform any actions here which were performed in the prerelease
    # command.

    print DBG "-- start undoprerelease phase --\n";
    foreach $key (sort keys %ENV) {
        if ($key =~ /^DR_/) {
            print DBG $key, '=', $ENV{$key}, "\n";
        }
    }

    # resource specific actions
    USE_SWITCH: {
        if ($res eq $REGISTER_DATA{CPU_RESOURCE}) {
            # undo cpu related changes done by the prerelease cmd
            last USE_SWITCH;
        }
        if ($res eq $REGISTER_DATA{MEM_RESOURCE}) {
            # undo mem related changes done by the prerelease cmd
            last USE_SWITCH;
        }
        print "DR_ERROR=script does not use resource $res\n";
        $l_rc = 1;
    }

    print DBG "-- end undoprerelease phase --\n";

    return $l_rc;
}
```

```
#======================================================================
#  Name:  process_postrelease
#
#  Description:  After the resource is removed, this command makes
#                necessary adjustments
#
#  Input: resource
#
#  Output: Any DR_ERROR=description or DR_LOG_DEBUG=description variables
#
#  Return Code:  0 = success
#                1 = failure
#======================================================================
sub process_postrelease {

    $l_rc = 0;
    $res = $_[0];

    # reacquire any resource released during prerelease
    # activate any applications quieced during prerelease.

    print DBG "-- start postrelease phase --\n";
    foreach $key (sort keys %ENV) {
        if ($key =~ /^DR_/) {
            print DBG $key, '=', $ENV{$key}, "\n";
        }
    }

    # resource specific actions
    USE_SWITCH: {
        if ($res eq $REGISTER_DATA{CPU_RESOURCE}) {
            # perform cpu related actions.
            last USE_SWITCH;
        }
        if ($res eq $REGISTER_DATA{MEM_RESOURCE}) {
            # perform mem related actions.
            last USE_SWITCH;
        }
        print "DR_ERROR=script does not use resource $res\n";
        $l_rc = 1;
    }

    print DBG "-- end postrelease phase --\n";
    return $l_rc;
}

#======================================================================
#  Name:  process_checkacquire
#
```

```
#  Description:  verifies a resource can be added withouth
#                compromising the application or system.
#
#  Input: resource
#
#  Output: Any DR_ERROR=description or DR_LOG_DEBUG=description variables
#
#  Return Code:  0 = success
#                1 = failure
#=====================================================================
sub process_checkacquire {

    $l_rc = 0;
    $res = $_[0];

    print DBG “-- start checkacquire phase --\n”;
    foreach $key (sort keys %ENV) {
        if ($key =~ /^DR_/) {
            print DBG $key, ‘=’, $ENV{$key}, “\n”;
        }
    }

    # resource specific actions
    USE_SWITCH: {
        if ($res eq $REGISTER_DATA{CPU_RESOURCE}) {
            # perform all cpu related checks here and determine
            # if resource addition can proceed.
            print DBG “cpu resources: logical $ENV{DR_LCPUID}, bind $ENV{DR_BCPUID}\n”;
            if ($END{DR_LCPUID} eq 2) {
                $l_rc = 1;
            }
            last USE_SWITCH;
        }
        if ($res eq $REGISTER_DATA{MEM_RESOURCE}) {
            # perform all mem related checks here and determine
            # if resource addition can proceed.
            print DBG “mem resources: $ENV{DR_MEM_SIZE_REQUEST}\n”;
            last USE_SWITCH;
        }
        print “DR_ERROR=script does not use resource $res\n”;
        $l_rc = 1;
    }

    print DBG “-- end checkacquire phase --\n”;
    return $l_rc;
}

#=====================================================================
#  Name:  process_preacquire
```

```
#
#  Description:  prepares application before the resource is added
#
#  Input: resource
#
#  Output: Any DR_ERROR=description or DR_LOG_DEBUG=description variables
#
#  Return Code:  0 = success
#                1 = failure
#=======================================================================
sub process_preacquire {

    $l_rc = 0;
    $res = $_[0];

    print DBG “-- start preacquire phase --\n”;
    foreach $key (sort keys %ENV) {
        if ($key =~ /^DR_/) {
            print DBG $key, ‘=’, $ENV{$key}, “\n”;
        }
    }

    # resource specific actions
    USE_SWITCH: {
        if ($res eq $REGISTER_DATA{CPU_RESOURCE}) {
            # Prepare application for cpu additions.
            last USE_SWITCH;
        }
        if ($res eq $REGISTER_DATA{MEM_RESOURCE}) {
             # Prepare application for memory additions.
            last USE_SWITCH;
        }
        print “DR_ERROR=script does not use resource $res\n”;
    $l_rc = 1;
    }

    print DBG “-- end preacquire phase --\n”;

    return $l_rc;
}

#=======================================================================
#  Name:  process_undopreacquire
#
#  Description:  If a failure occues, this will undo any changes made by
#                the preacquire command.
#
#  Input: resource
#
```

```
#  Output: Any DR_ERROR=description or DR_LOG_DEBUG=description variables
#
#  Return Code:  0 = success
#                1 = failure
#=======================================================================
sub process_undopreacquire {

    $l_rc = 0;
    $res = $_[0];

    print DBG "-- start undopreacquire phase --\n";
    foreach $key (sort keys %ENV) {
        if ($key =~ /^DR_/) {
            print DBG $key, '=', $ENV{$key}, "\n";
        }
    }

    # resource specific actions
    USE_SWITCH: {
        if ($res eq $REGISTER_DATA{CPU_RESOURCE}) {
            # undo cpu actions taken in the preacquire command.
            last USE_SWITCH;
        }
        if ($res eq $REGISTER_DATA{MEM_RESOURCE}) {
            # undo mem actions taken in the preacquire command.
            last USE_SWITCH;
        }
        print "DR_ERROR=script does not use resource $res\n";
        $l_rc = 1;
    }

    print DBG "-- end undopreacquire phase --\n";

    return $l_rc;
}

#=======================================================================
#  Name:  process_postacquire
#
#  Description:  After a resource has been added, this will perform any
#                necessary actions.
#
#  Input: resource
#
#  Output: Any DR_ERROR=description or DR_LOG_DEBUG=description variables
#
#  Return Code:  0 = success
#                1 = failure
#=======================================================================
sub process_postacquire {

    $l_rc = 0;
    $res = $_[0];

    print DBG "-- start undopreacquire phase --\n";
    foreach $key (sort keys %ENV) {
        if ($key =~ /^DR_/) {
            print DBG $key, '=', $ENV{$key}, "\n";
        }
    }

    # resource specific actions
    USE_SWITCH: {
        if ($res eq $REGISTER_DATA{CPU_RESOURCE}) {
            # Perform actions to allow the application to adjust to a
            # cpu addition such as adding more threads, etc.
            last USE_SWITCH;
        }
        if ($res eq $REGISTER_DATA{MEM_RESOURCE}) {
            # Perform actions to allow the application to adjust to a
            # memory addition such as increasing memory areas reserved
            # for application, etc.
            last USE_SWITCH;
        }
        print "DR_ERROR=script does not use resource $res\n";
        $l_rc = 1;
    }

    print DBG "-- end undopreacquire phase --\n";

    return $l_rc;
}

#---------------------------------------------------------------------
# Main Program
#---------------------------------------------------------------------

# because we should only write the specified name-value
# pairs in the DRAF, we should print debug information
# to a file.
open (DBG, ">>/tmp/dr_IBM_template.pl.dbg");

# This block processes the command line inputs.
ARG_SWITCH: {
    if ($#ARGV == -1) { $rc = -1; last ARG_SWITCH; }
    if ($#ARGV == 0) { $command_str = $ARGV[0] ; last ARG_SWITCH; }
    if ($#ARGV == 1) { $command_str = $ARGV[0]; $res_name = $ARGV[1]; last ARG_SWITCH; }
    $rc = -2;
}

# Convert the string to a command.
$command = str_to_cmd $command_str;

#This block invokes the proper function to handle the command
CMD_SWITCH: {
    if ($command == ‘’) {$rc = 10; print “DR_ERROR=command not supported\n”; last CMD_SWITCH }
    if ($command == 1) {$rc = process_scriptinfo; last CMD_SWITCH }
    if ($command == 2) {$rc = process_register; last CMD_SWITCH }
    if ($command == 3) {$rc = process_usage $res_name; last CMD_SWITCH }
    if ($command == 4) {$rc = process_checkrelease $res_name; last CMD_SWITCH }
    if ($command == 5) {$rc = process_prerelease $res_name; last CMD_SWITCH }
    if ($command == 6) {$rc = process_postrelease $res_name; last CMD_SWITCH }
    if ($command == 7) {$rc = process_undoprerelease $res_name; last CMD_SWITCH }
    if ($command == 8) {$rc = process_checkacquire $res_name; last CMD_SWITCH }
    if ($command == 9) {$rc = process_preacquire $res_name; last CMD_SWITCH }
    if ($command == 10) {$rc = process_postacquire $res_name; last CMD_SWITCH }
    if ($command == 11) {$rc = process_undopreacquire $res_name; last CMD_SWITCH }
}

# close the debug file handle
close(DBG);
# exit status generated from command processing
$rc;
```

Korn shell template

Example B-3 on page 422 provides the Korn shell version of the DLPAR script template. It can be used by system administrators as a starting point to integrate DLPAR operations into your applications.

The script sends debug information to the debug file, /tmp/<script_file_name>.dbg. The debug information is very helpful when you have to debug your DLPAR script, because the script should not print any undefined name-value pairs to the standard out.

To display the debug information sent to the file, type the following command:

```
$ tail -f /tmp/<script_file_name>.dbg
```

Example: B-3   DLPAR script template: Korn shell

```
#! /usr/bin/ksh
# (C) COPYRIGHT International Business Machines Corp. 2000, 2002
# All Rights Reserved
# Licensed Materials - Property of IBM
#
# US Government Users Restricted Rights - Use, duplication or
# disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
#
#
# DLPAR aware Application developers will modify this script to
# develop a DLPAR script to suit their application's needs of control
# vis-a-vis Dynamic Reconfiguration(DLPAR) Operations.
#


# FILE NAME: IBM_template.sh
#
# FILE DESCRIPTION:
#   This is an example template shell DLPAR script file for
#   Dynamic Reconfiguration Application Framework of AIX.
#   This template file just prints the various inputs
#   from drmgr.
#
#   Note that DLPAR script file should adher to the guildelines
#   related to AIX Dynamic Reconfiguration. Some of the
#   issues to be considered while chaagng this script file
#   are:
#   1. Output name=value pairs only to stdout
#      as per the DRAF guidelines. Refer to
#      Manuals related to DLPAR for more details.
#   2. Return 0 upon success, 10 if the command
#      is not implemented, else return any other
#      return code (1 to 9, 11 to 255)
#   3. Use DRAF defined environment variables and
#      input parameters for processing the
#      command.
#   4. To debug the script file, one can use
#      the method shown in this template file.
#
# RETURN VALUE DESCRIPTION:
#      0         Successful
#      10        Command not implemented
#      Else      Error
#
################################ dbg ######################################
#
# NAME: dbg()
```

```
#
# DESCRIPTION: Write the debug message to debug file
#
# INPUT:
# Message to write to debug file
#
# OUTPUT:
# Message echoed to the debug file.
#
# RETURN VALUE DESCRIPTION:
# None
#
##############################################################################

dbg()
{
    echo $1 >> ${DBG_FILE_NAME}
}

############################## process_scriptinfo ############################
#
# NAME: process_scriptinfo()
#
# DESCRIPTION: Process 'scriptinfo' command from drmgr
#
# INPUT:
# The various environment variables set by drmgr
#
# OUTPUT:
# Output name=value pairs to stdout
# Various pieces of information about the DLPAR script.
#
# RETURN VALUE DESCRIPTION:
# 0 : success
# Else failure.
#
##############################################################################

process_scriptinfo()
{
    echo "DR_SCRIPTINFO=AIX DR ksh example script"
    echo "DR_VERSION=1"
    echo "DR_DATE=18102002"
    echo "DR_VENDOR=IBM"
    echo "DR_TIMEOUT=10"
    return 0
}

############################## process_register   ############################
```

```
#
# NAME: process_register()
#
# DESCRIPTION: Process 'register' command from drmgr
#
# INPUT:
# The various environment variables set by drmgr
#
# OUTPUT:
# Output name=value pairs to stdout
# List of all the resources supported by this DLPAR script.
#
# RETURN VALUE DESCRIPTION:
# 0 : success
# Else failure.
#
##############################################################################

process_register()
{
    echo "DR_RESOURCE=cpu"
    echo "DR_RESOURCE=mem"
    return 0
}

############################## process_usage      ############################
#
# NAME: process_usage()
#
# DESCRIPTION: Process 'usage' command from drmgr
#
# INPUT:
# The various environment variables set by drmgr
# resource name input variable
#
# OUTPUT:
# Output name=value pairs to stdout
# Writes the how this resource is being used by the application
# associated with this DLPAR script.
#
# RETURN VALUE DESCRIPTION:
# 0 : success
# Else failure.
#
##############################################################################

process_usage()
{
    case "$1" in
```

```
        "cpu")
        echo "DR_USAGE=cpu binding for performance"
        ;;
        "mem")
        echo "DR_USAGE=Shared(Pinned) memory for app XYZ"
        ;;
        *)
        echo "DR_ERROR=Script does not use Resource $1"
        ;;
    esac
    return 0
}

############################## process_checkrelease ##########################
#
# NAME: process_checkrelease()
#
# DESCRIPTION: Process 'checkrelease' command from drmgr
#
# INPUT:
# The various environment variables set by drmgr
# resource name input variable
#
# OUTPUT:
# Any debug information using DR debug name=value pairs
# such as DR_LOG_DEBUG="..."
# Any error message in the form of DR_ERROR name=value pair
#
#
# RETURN VALUE DESCRIPTION:
# 0 : success
# 10 : Command not implemented for this resource.
# Else DLPAR operatin is not ok with the DLPAR script/associated app.
#
##############################################################################

process_checkrelease()
{
    case "$1" in
        "cpu")
        dbg "Resource : cpu"
        # Do all the cpu related checks here and determine
        # whether DLPAR remove can proceed.
        ;;
        "mem")
        dbg "Resource : mem"
        # Do all the memory  related checks here and determine
        # whether DLPAR remove can proceed.
        ;;
        *)
        echo "DR_ERROR=Script does not support Resource $1"
        ;;
    esac

    return 0
}

############################## process_prerelease ############################
#
# NAME: process_prerelease()
#
# DESCRIPTION: Process 'prerelease' command from drmgr
#
# INPUT:
# The various environment variables set by drmgr
# resource name input variable
#
# OUTPUT:
# Any debug information using DR debug name=value pairs
# such as DR_LOG_DEBUG="..."
# Any error message in the form of DR_ERROR name=value pair
#
#
# RETURN VALUE DESCRIPTION:
# 0 : success
# 10 : Command not implemented for this resource.
# Else DLPAR script/associated application could not release the resource
# for DLPAR operation.
#
##############################################################################
process_prerelease()
{
    # Do any pre release actions here. One could send a signal
    # from here and wait for application do the necessary.
    # Return from here only after the desired actions have
    # taken place.
    case "$1" in
        "cpu")
        bg "Resource : cpu"
        # Release any cpu bindings etc here if the
        # resource being released is used by the app.
        ;;
        "mem")
        dbg "Resource : mem"
        # Release application hold over any memory
        # that is being removed.
        ;;
        *)
        echo "DR_ERROR=Script does not support Resource $1"
        ;;
    esac

    return 0
}

############################## process_postrelease ############################
#
# NAME: process_postrelease()
#
# DESCRIPTION: Process 'postrelease' command from drmgr
#
# INPUT:
# The various environment variables set by drmgr
# resource name input variable
#
# OUTPUT:
# Any debug information using DR debug name=value pairs
# such as DR_LOG_DEBUG="..."
# Any error message in the form of DR_ERROR name=value pair
#
#
# RETURN VALUE DESCRIPTION:
# 0 : success
# 10 : Command not implemented for this resource.
# Else DLPAR script/associated application could not post DLPAR operations.
#
###############################################################################
process_postrelease()
{
    # Reacquire any resource release during prerelease.
    # activate any apps quieced during prerelease.

    case "$1" in
        "cpu")
        dbg "Resource : cpu"
        ;;
        "mem")
        dbg "Resource : mem"
        ;;
        *)
        echo "DR_ERROR=Script does not support Resource $1"
        ;;
    esac

    return 0
}
```

```
############################## process_undoprerelease ########################
#
# NAME: process_undoprerelease()
#
# DESCRIPTION: Process 'process_undoprerelease' command from drmgr
#
# INPUT:
# The various environment variables set by drmgr
# resource name input variable
#
# OUTPUT:
# Any debug information using DR debug name=value pairs
# such as DR_LOG_DEBUG="..."
# Any error message in the form of DR_ERROR name=value pair
#
#
# RETURN VALUE DESCRIPTION:
# 0 : success
# 10 : Command not implemented for this resource.
# Else DLPAR script/associated application failed undorelease
#
###############################################################################
process_undoprerelease()
{
    # DLPAR operation was aborted/failed. Hence undo any
    # changes done during prerelease for this resource
    # and the application associated with the DLPAR script.
    case "$1" in
        "cpu")
        dbg "Resource : cpu"
        ;;
        "mem")
        dbg "Resource : mem"
        ;;
        *)
        echo "DR_ERROR=Script does not support Resource $1"
        ;;
    esac
    return 0
}

############################## process_checkacquire   ########################
#
# NAME: process_checkacquire()
#
# DESCRIPTION: Process 'process_checkacquire' command from drmgr
#
# INPUT:
# The various environment variables set by drmgr
# resource name input variable
#
# OUTPUT:
# Any debug information using DR debug name=value pairs
# such as DR_LOG_DEBUG="..."
# Any error message in the form of DR_ERROR name=value pair
#
#
# RETURN VALUE DESCRIPTION:
# 0 : success
# 10 : Command not implemented for this resource.
# Else DLPAR script/associated application does want this resource.
#
##############################################################################
process_checkacquire()
{
    # Do any checks prior to resource addition.
    case "$1" in
        "cpu")
        dbg "Resource : cpu"
        ;;
        "mem")
        dbg "Resource : mem"
        ;;
        *)
        echo "DR_ERROR=Script does not support Resource $1"
        ;;
    esac
    return 0
}
############################## process_preacquire     ########################
#
# NAME: process_preacquire()
#
# DESCRIPTION: Process 'process_preacquire' command from drmgr
#
# INPUT:
# The various environment variables set by drmgr
# resource name input variable
#
# OUTPUT:
# Any debug information using DR debug name=value pairs
# such as DR_LOG_DEBUG="..."
# Any error message in the form of DR_ERROR name=value pair
#
#
# RETURN VALUE DESCRIPTION:
# 0 : success
# 10 : Command not implemented for this resource.
# Else DLPAR script/associated application preacquire failed.
#
##############################################################################
process_preacquire()
{
 # Do all the necessary work prior to resource addition.
 case "$1" in
  "cpu")
   dbg "Resource : cpu"
   ;;
  "mem")
   dbg "Resource : mem"
   ;;
  *)
   echo "DR_ERROR=Script does not support Resource $1"
   ;;
 esac
 return 0
}

############################## process_undopreacquire ########################
#
# NAME: process_undopreacquire()
#
# DESCRIPTION: Process 'process_undopreacquire' command from drmgr
#
# INPUT:
# The various environment variables set by drmgr
# resource name input variable
#
# OUTPUT:
# Any debug information using DR debug name=value pairs
# such as DR_LOG_DEBUG="..."
# Any error message in the form of DR_ERROR name=value pair
#
#
# RETURN VALUE DESCRIPTION:
# 0 : success
# 10 : Command not implemented for this resource.
# Else DLPAR script/associated application undopreacquire failed.
#
##############################################################################
process_undopreacquire()
{
    # DLPAR operation has failed. So undo any activities done during
    # preacquire
    case "$1" in
        "cpu")
        dbg "Resource : cpu"
        ;;
        "mem")
        dbg "Resource : mem"
        ;;
        *)
        echo "DR_ERROR=Script does not support Resource $1"
        ;;
    esac
    return 0
}
############################## process_postacquire  #########################
#
# NAME: process_postacquire()
#
# DESCRIPTION: Process 'process_postacquire' command from drmgr
#
# INPUT:
# The various environment variables set by drmgr
# resource name input variable
#
# OUTPUT:
# Any debug information using DR debug name=value pairs
# such as DR_LOG_DEBUG="..."
# Any error message in the form of DR_ERROR name=value pair
#
#
# RETURN VALUE DESCRIPTION:
# 0 : success
# 10 : Command not implemented for this resource.
# Else DLPAR script/associated application postacquire failed.
#
##############################################################################
process_postacquire()
{
    # execute any actions required after the DLPAR add operation.
    # Egs: Increase the number of threads for the application
    #      Increase memory areas reserved for application etc.
    case "$1" in
        "cpu")
        dbg "Resource : cpu"
        ;;
        "mem")
        dbg "Resource : mem"
        ;;
        *)
        echo "DR_ERROR=Script does not support Resource $1"
        ;;
    esac
    return 0
```

```
}

###############################################
#            MAIN SCRIPT STARTS HERE
###############################################

script_file_name=`basename $0`
DBG_FILE_NAME=/tmp/${script_file_name}.dbg

date_and_time=`date`
dbg "------ DLPAR Script start at $date_and_time -------"

if [ $# -eq 0 ]; then
    # Atleast the command must be part of the invocation
    dbg "No command passed to the DLPAR script"
    echo "DR_ERROR=Script Usage Error"
    exit 1
fi

# Note down the command
command=$1
ret_code=0

dbg "command issued: $1"
case "$1" in
    scriptinfo)
    process_scriptinfo
    ret_code=$?
    ;;
    register)
    process_register
    ret_code=$?
    ;;
    usage)
    process_usage $2
    ret_code=$?
    ;;
    checkrelease)
    process_checkrelease $2
    ret_code=$?
    ;;
    prerelease)
    process_prerelease $2
    ret_code=$?
    ;;
    postrelease)
    process_postrelease $2
    ret_code=$?
    ;;
    undoprerelease)
    process_undoprerelease $2
    ret_code=$?
    ;;
    checkacquire)
    process_checkacquire $2
    ret_code=$?
    ;;
    preacquire)
    process_preacquire $2
    ret_code=$?
    ;;
    undopreacquire)
    process_undopreacquire $2
    ret_code=$?
    ;;
    postacquire)
    process_postacquire $2
    ret_code=$?
    ;;
    *)
    dbg "unknown command: $1 issued"
    ret_code=10
    ;;
    esac

dbg "SCRIPT exiting with return code : $ret_code"

dbg "...............DLPAR Script end  ..............."

return $ret_code
```

B.1 Sample DLPAR-aware application using a signal handler

If you can access the source code of your application, you can modify your application by adding a signal handler that reacts to the SIGRECONFIG signal so that the application is made DLPAR-aware. The SIGRECONFIG signal will be delivered to the application process upon DLPAR events.

Example B-4 on page 434 can be used by application programmers as a starting point to add a signal handler to the application.

B.1.1 How to compile and run the application

Before running this application, you must compile the C source code by executing the following command[1]:

```
$ cc -o DLPAR_appl DLPAR_appl.c
```

Where:

DLPAR_appl	The application program name to be executed
DLPAR_appl.c	The file name of C program source code

To run this application, type **`DLPAR_appl`** at the command line prompt. To stop it, type **`Control-C`** at the command line prompt where you have invoked it.

The application process sends debug information to the /tmp/dr_api_template.C.dbg file. To display the debug information sent to the file, issue the following:

```
$ tail -f /tmp/dr_api_template.C.dbg
```

Example: B-4 C language application with a signal handler

```
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <errno.h>
#include <sys/dr.h>

/*=====================================================================*/
/*  Prototypes                                                         */
/*=====================================================================*/
void perror_msg(char *func_name, int errno_old, const int line_num);
void dr_func(int arg);
int checkAcquire_mem(void);
int checkAcquire_cpu(void);
int preAcquire_mem(void);
int preAcquire_cpu(void);
int postAcquire_mem(void);
int postAcquire_cpu(void);
int postAcquireError_mem(void);
int postAcquireError_cpu(void);
int checkRelease_mem(void);
int checkRelease_cpu(void);
int preRelease_mem(void);
int preRelease_cpu(void);
int postRelease_mem(void);
int postRelease_cpu(void);
```

[1] This example requires that the C compiler is installed and available on your AIX system.

```
int postReleaseError_mem(void);
int postReleaseError_cpu(void);

/*====================================================================*/
/*  Globals                                                           */
/*====================================================================*/
extern int errno;
dr_info_t dr_info;
char msg_buf[BUFSIZ];
FILE * l_dbgFd;

typedef struct {
    int (*mem_ptr)(void);
    int (*cpu_ptr)(void);
} phases_t;

phases_t definedPhase[] = {                                        /* #J */
    { &checkAcquire_mem,      &checkAcquire_cpu     },
    { &preAcquire_mem,        &preAcquire_cpu       },
    { &postAcquire_mem,       &postAcquire_cpu      },
    { &postAcquireError_mem,  &postAcquireError_cpu },
    { &checkRelease_mem,      &checkRelease_cpu     },
    { &preRelease_mem,        &preRelease_cpu       },
    { &postRelease_mem,       &postRelease_cpu      },
    { &postReleaseError_mem,  &postReleaseError_cpu }
};

#define CHECK_ACQUIRE         &definedPhase[0];
#define PRE_ACQUIRE           &definedPhase[1];
#define POST_ACQUIRE          &definedPhase[2];
#define POST_ACQUIRE_ERROR    &definedPhase[3];
#define CHECK_RELEASE         &definedPhase[4];
#define PRE_RELEASE           &definedPhase[5];
#define POST_RELEASE          &definedPhase[6];
#define POST_RELEASE_ERROR    &definedPhase[7];

#define DR_API_SUCCESS        0
#define DR_API_FAIL           1

/*====================================================================*/
/*  Helper Functions                                                  */
/*====================================================================*/

/*====================================================================*/
/* Description:  Handles any unexected errors                         */
/*                                                                    */
/* Output: none                                                       */
/* Inputs: function name                                              */
/*         errno                                                      */
/*        linenumber                                                 */
/*===================================================================*/
void
perror_msg(char *func_name, int errno_old, const int line_num)
{
    sprintf(msg_buf
        , "%s failed with errno = %d at line (%d).\n"
        , func_name, errno_old, line_num);
    perror(msg_buf);

    return;
}

/*===================================================================*/
/* Description: Memory Check Acquire phase                           */
/*                                                                   */
/*       This function should ensure that a memory addition will not   */
/*       disrupt the application, before the actual DR event occurs.   */
/*                                                                   */
/*  Output:  integer value indicating status of DR operation          */
/*     Values: DR_API_SUCCESS                                        */
/*  Input:  None                                                     */
/*===================================================================*/
int  checkAcquire_mem(void)
{
    int l_rc = DR_API_SUCCESS;

    fprintf(l_dbgFd, "*****Entered CheckedAcquire_mem*****\n");

    /* Check for plock'd memory */
    if (dr_info.plock) {
        fprintf(l_dbgFd, "\t-- process has plock()'ed memory --\n");
        l_rc = DR_API_FAIL;
    }
    /* check for pinned memory */
    if (dr_info.pshm) {
        fprintf(l_dbgFd, "\t-- process has pinned shared memory --\n");
        l_rc = DR_API_FAIL;
    }

    return l_rc;
}

/*===================================================================*/
/* Description:  CPU check acquire phase                             */
/*      This function should ensure that a cpu addition will not       */
/*      disrupt the application, before the actual DR event takes place*/
/*                                                                   */
/* Output: integer value indicating status of DR operation            */
/*     Values:  DR_API_SUCCESS                                          */
/*              DR_API_FAIL                                             */
/* Inputs: None                                                         */
/*====================================================================*/
int checkAcquire_cpu(void)
{
    int l_rc = DR_API_SUCCESS;

    fprintf(l_dbgFd, "*****Entered CheckedAcquire_cpu*****\n");

    /* check for processor dependencies */
    if (dr_info.bindproc) {
        fprintf(l_dbgFd, "\t-- process has bindprocessor() dependency --\n");
        l_rc = DR_API_FAIL;
    }
    if (dr_info.softpset) {
        fprintf(l_dbgFd, "\t-- process has soft pset() dependency --\n");
        l_rc = DR_API_FAIL;
    }
    if (dr_info.hardpset) {
        fprintf(l_dbgFd, "\t-- process has hard pset() dependency --\n");
        l_rc = DR_API_FAIL;
    }

    return l_rc;
}

/*====================================================================*/
/*  Mem pre acquire phase                                               */
/*                                                                      */
/*  Detailed Description:                                               */
/*      This function should ensure that the necessary steps are taken */
/*      to prepare the application for a memory addition.  If need be, */
/*      the application should be halted.                               */
/*                                                                      */
/* Output: integer value indicating status of DR operation             */
/*    Values: DR_API_SUCCESS                                            */
/*            DR_API_FAIL                                               */
/* Inputs: None                                                         */
/*====================================================================*/
int  preAcquire_mem(void)
{
    int l_rc = DR_API_SUCCESS;

    fprintf(l_dbgFd, "*****Entered PreeAcquire_mem*****\n");

    /* Perform actions here. */

    return l_rc;
```

```
}

/*==================================================================*/
/* CPU pre acquire phase                                            */
/*                                                                  */
/* Detailed Description:                                            */
/*      This function should ensure that the necessary steps are taken */
/*      to prepare the application for a cpu addition. If need be,     */
/*      the application should be stopped.                          */
/*                                                                  */
/* Output: integer value indicating status of DR operation          */
/*     Values:  DR_API_SUCCESS                                      */
/*              DR_API_FAIL                                         */
/* Inputs: None                                                     */
/*==================================================================*/
int preAcquire_cpu(void)
{
    int l_rc = DR_API_SUCCESS;

    fprintf(l_dbgFd, “*****Entered PreAcquire_cpu*****\n”);

    /* Perform actions here */

    return l_rc;
}

/*==================================================================*/
/* Mem post acquire phase                                           */
/*                                                                  */
/* Detailed Description:                                            */
/*      After a memory addition has taken place, this function should  */
/*      perform any actions to clean up the DR operation and allow the */
/*      application to use the new resources.  If the application was  */
/*      stopped, it should be restarted here.                       */
/*                                                                  */
/*  Output: integer value indicating status of DR operation         */
/*     Values: DR_API_SUCCESS                                       */
/*             DR_API_FAIL                                          */
/* Inputs: None                                                     */
/*==================================================================*/
int postAcquire_mem(void)
{
    int l_rc = DR_API_SUCCESS;

    fprintf(l_dbgFd, “*****Entered PostAcquire_mem*****\n”);

    /* Perform actions here */

    return l_rc;
```

```
}

/*==================================================================*/
/* CPU post acquire phase                                           */
/*                                                                  */
/* Detailed Description:                                            */
/*      After a cpu addition, this function allows the application */
/*      access to the new resources.  If the application was stopped,  */
/*      iy should be restarted here.                                */
/*                                                                  */
/*  Output: integer value indicating status of DR operation         */
/*     Values: DR_API_SUCCESS                                       */
/*             DR_API_FAIL                                          */
/* Inputs: None                                                     */
/*==================================================================*/
int postAcquire_cpu(void)
{
    int l_rc = DR_API_SUCCESS;

    fprintf(l_dbgFd, "*****Entered PostAcquire_cpu*****\n");

    /* Perform actions here */

    return l_rc;
}

/*==================================================================*/
/* Handles post acquire Error phase for mem                         */
/*                                                                  */
/* Detailed Description:                                            */
/*      If an error should occur, this phase should handle undoing the */
/*      preacquire actions taken for mem rmovals.                   */
/*                                                                  */
/*  Output: integer value indicating status of DR operation         */
/*     Values: DR_API_SUCCESS                                       */
/*             DR_API_FAIL                                          */
/* Inputs: None                                                     */
/*==================================================================*/
int postAcquireError_mem(void)
{
    int l_rc = DR_API_SUCCESS;

    fprintf(l_dbgFd, "*****Entered PostAcquireError_mem*****\n");

    /* Perform actions here */

    return l_rc;
}
```

```
/*====================================================================*/
/* Handles post acquire Error phase for cpu                           */
/*                                                                    */
/* Detailed Description:                                              */
/*      If an error should occur, this phase should handle undoing the */
/*      preacquire actions taken for cpu rmovals.                     */
/*                                                                    */
/* Output: integer value indicating status of DR operation            */
/*     Values: DR_API_SUCCESS                                         */
/*             DR_API_FAIL                                            */
/* Inputs: None                                                       */
/*====================================================================*/
int postAcquireError_cpu(void)
{
    int l_rc = DR_API_SUCCESS;

    fprintf(l_dbgFd, "*****Entered PostAcquireError_cpu*****\n");

    /* Perform actions here */

    return l_rc;
}

/*====================================================================*/
/* Mem check release phase                                            */
/*                                                                    */
/* Detailed Description:                                              */
/*      This should check to make sure the application can tolerate a  */
/*      memory removal.  If not, this should terminate the DR operation*/
/*                                                                    */
/* Output: integer value indicating status of DR operation            */
/*     Values: DR_API_SUCCESS                                         */
/*             DR_API_FAIL                                            */
/* Inputs: None                                                       */
/*====================================================================*/
int checkRelease_mem(void)
{
    int l_rc = DR_API_SUCCESS;

    fprintf(l_dbgFd, "*****Entered CheckeRelease_mem*****\n");

    /* Check for memory issues */
    if (dr_info.plock) {
        fprintf(l_dbgFd, "\t-- process has plock()'ed memory --\n");
        l_rc = DR_API_FAIL;
    }
    if (dr_info.pshm) {
        fprintf(l_dbgFd, "\t-- process has pinned shared memory --\n");
        l_rc = DR_API_FAIL;
    }
    return l_rc;

}

/*==================================================================*/
/* Handles post release Error phase for cpu                         */
/*                                                                  */
/* Detailed Description:                                            */
/*      If an error should occur, this phase should handle undoing the */
/*      prerelease actions taken for cpu rmovals.                   */
/*                                                                  */
/* Output:  integer value indicating status of DR operation         */
/*     Values: DR_API_SUCCESS                                       */
/*           l DR_API_FAIL                                          */
/* Inputs: None                                                     */
/*==================================================================*/
int checkRelease_cpu(void)
{
    int l_rc = DR_API_SUCCESS;

    fprintf(l_dbgFd, "*****Entered CheckRelease_cpu*****\n");

    /* Check for processor dependencies */
    if (dr_info.bindproc) {
        fprintf(l_dbgFd, "\t-- process has bindprocessor() dependency --\n");
        l_rc = DR_API_FAIL;
    }
    if (dr_info.softpset) {
        fprintf(l_dbgFd, "\t-- process has soft pset() dependency --\n");
        l_rc = DR_API_FAIL;
    }
    if (dr_info.hardpset) {
        fprintf(l_dbgFd, "\t-- process has hard pset() dependency --\n");
        l_rc = DR_API_FAIL;
    }
    return l_rc;
}

/*==================================================================*/
/* Mem pre release phase                                            */
/*                                                                  */
/* Detailed Description:                                            */
/*      This function should prepare the application for memory    */
/*      resources being removed.                                    */
/*                                                                  */
/* Output: integer value indicating status of DR operation          */
/*     Values: DR_API_SUCCESS                                       */
/*             DR_API_FAIL                                          */
/* Inputs: None                                                         */
/*======================================================================*/
int preRelease_mem(void)
{
    int l_rc = DR_API_SUCCESS;

    fprintf(l_dbgFd, "*****Entered PreRelease_mem*****\n");

    /* Perform actions here */

    return l_rc;
}

/*======================================================================*/
/* Cpu pre release phase                                                */
/*                                                                      */
/* Detailed Description:                                                */
/*      This should prepare the application for cpu resources being    */
/*      removed.                                                        */
/*                                                                      */
/* Output: integer value indicating status of DR operation              */
/*     Values: DR_API_SUCCESS                                           */
/*             DR_API_FAIL                                              */
/* Inputs: None                                                         */
/*======================================================================*/
int preRelease_cpu(void)
{
    int l_rc = DR_API_SUCCESS;

    fprintf(l_dbgFd, "*****Entered PreRelease_cpu*****\n");

    /* Perform actions here */

    return l_rc;
}

/*======================================================================*/
/* Mem post release phase                                               */
/*                                                                      */
/* Detailed Description:                                                */
/*      After the memory resources are removed, this function should    */
/*      take care of cleaning up any DR modifications made and allow    */
/*      the application to continue running.                            */
/*                                                                      */
/* Output: integer value indicating status of DR operation              */
/*     Values: DR_API_SUCCESS                                           */
/*             DR_API_FAIL                                              */
/* Inputs: None                                                         */
/*======================================================================*/
```

```
int  postRelease_mem(void)
{
    int l_rc = DR_API_SUCCESS;

    fprintf(l_dbgFd, "*****Entered PostReleasee_mem*****\n");

    /* Perform actions here */

    return l_rc;
}

/*==================================================================*/
/* Cpu post release phase                                           */
/*                                                                  */
/* Detailed Description:                                            */
/*      After cpu resources are  removed, this function should handle  */
/*      the application so that it can continue after the DR operation.*/
/*                                                                  */
/* Output: integer value indicating status of DR operation          */
/*     Values: DR_API_SUCCESS                                       */
/*     Values: DR_API_FAIL                                          */
/* Inputs: None                                                     */
/*==================================================================*/
int  postRelease_cpu(void)
{
    int l_rc = DR_API_SUCCESS;

    fprintf(l_dbgFd, "*****Entered PostRelease_cpu*****\n");

    /* Perform actions here */

    return l_rc;
}

/*==================================================================*/
/* Handles post release Error phase for mem                         */
/*                                                                  */
/* Detailed Description:                                            */
/*      If an error should occur, this phase should handle undoing the */
/*      prerelease actions taken for mem rmovals.                   */
/*                                                                  */
/* Output: integer value indicating status of DR operation          */
/*     Values: DR_API_SUCCESS                                       */
/*             DR_API_FAIL                                          */
/* Inputs: None                                                     */
/*==================================================================*/
int  postReleaseError_mem(void)
{
    int l_rc = DR_API_SUCCESS;
    fprintf(l_dbgFd, "*****Entered PostReleaseError_mem*****\n");

    /* Perform actions here */

    return l_rc;
}

/*=====================================================================*/
/* Handles post release Error phase for cpu                            */
/*                                                                     */
/* Detailed Description:                                               */
/*      If an error should occur, this phase should handle undoing the */
/*      prerelease actions taken for cpu rmovals.                      */
/*                                                                     */
/* Output: integer value indicating status of DR operation             */
/*     Values: DR_API_SUCCESS                                          */
/*             DR_API_FAIL                                             */
/* Inputs: None                                                        */
/*=====================================================================*/
 int postReleaseError_cpu(void)
{
    int l_rc = DR_API_SUCCESS;

    fprintf(l_dbgFd, "*****Entered PostReleaseError_cpu*****\n");

    /* Perform actions here */

    return l_rc;
}

/*=====================================================================*/
/*  Functions                                                          */
/*=====================================================================*/

/*=====================================================================*/
/* SIGRECONFIG signal handler                                          */
/*                                                                     */
/* Detailed Description:                                               */
/*      This will handle the signal, when a DR event occurs.  The main */
/*      information is communicated through a dr_info_t data structure.*/
/*      This will take care of parsing the data structure and taking   */
/*      appropriate actions.                                           */
/*                                                                     */
/* Output: None                                                        */
/* Input: argument, not used for compatability                         */
/*=====================================================================*/
void
dr_func(int arg)
```

```
{
    int l_rc= DR_API_SUCCESS;
    phases_t *l_currentPhase;

/* #1 create debug output file stream */
    l_dbgFd = fopen("/tmp/dr_api_template.C.dbg", "a"); /* open for appending */
    if (l_dbgFd == NULL) {
        perror_msg("NULL file descriptor", errno, __LINE__);
        exit(1);
    }

    fprintf (l_dbgFd, "---Start of Signal Handler---\n");

    l_rc = dr_reconfig(DR_QUERY, &dr_info);              /* #C */
    if (l_rc  != 0) {
        perror_msg("dr_reconfig()", errno, __LINE__);
        exit(1);
    }

/* #2 determine type of operation and phase. */

    /* addition operations */
    if (dr_info.add) {                                   /* #D */
        fprintf(l_dbgFd, "An add request\n ");

        /* now determine which acquire phase we are in. */
        if (dr_info.check) {
            fprintf(l_dbgFd, "\t** check phase **\n");
            l_currentPhase = CHECK_ACQUIRE;
        } else if (dr_info.pre) {
            fprintf(l_dbgFd, "\t** pre  phase **\n");
            l_currentPhase = PRE_ACQUIRE;
        } else if (dr_info.post) {
            fprintf(l_dbgFd, "\t** post phase **\n");
            l_currentPhase = POST_ACQUIRE;
        } else if (dr_info.posterror) {
            fprintf(l_dbgFd, "\t** error phase **\n");
            l_currentPhase = POST_ACQUIRE_ERROR;
        }
    }
    /* remove operations. */
    if (dr_info.rem) {                                   /* #E */
        fprintf(l_dbgFd, "A remove request\n ");

        /* now determine which remove phase we are in. */
        if (dr_info.check) {
            fprintf(l_dbgFd, "\t** check phase **\n");
            l_currentPhase = CHECK_RELEASE;
        } else if (dr_info.pre) {
            fprintf(l_dbgFd, "\t** pre phase **\n");
            l_currentPhase = PRE_RELEASE;
        } else if (dr_info.post) {
            fprintf(l_dbgFd, "\t** post phase **\n");
        l_currentPhase = POST_RELEASE;
        } else if (dr_info.posterror) {
            fprintf(l_dbgFd, "\t** error phase **\n");
            l_currentPhase = POST_RELEASE_ERROR;
        }
    }

/* #3 invoke the command associated with the resource. */

    /* cpu resource. */
    if (dr_info.cpu) {                                   /* #F */
        fprintf(l_dbgFd, "Resource is CPU .\n");
        fprintf(l_dbgFd, "\tlogical CPU ID = %d\n\tBind CPU ID = %d\n"
            , dr_info.lcpu, dr_info.bcpu);

        /* invoke the command to process a cpu DR event */
        l_rc = l_currentPhase->cpu_ptr();                /* #H */

    /* memory resource. */
    } else if (dr_info.mem) {                            /* #G */
        fprintf(l_dbgFd, "Resource is Memory.\n");
        fprintf(l_dbgFd, "\trequested memory size (in bytes) = %lld\n"
            , dr_info.req_memsz_change);
        fprintf(l_dbgFd, "\tsystem memory size = %lld\n", dr_info.sys_memsz);
        fprintf(l_dbgFd, "\tnumber of free frames in system = %lld\n"
            , dr_info.sys_free_frames);
        fprintf(l_dbgFd, "\tnumber of pinnable frams in system = %lld\n"
            , dr_info.sys_pinnable_frames);
        fprintf(l_dbgFd, "\ttotal number of frames in system = %lld\n"
            , dr_info.sys_total_frames);

        /* invoke the command to process a mem DR event */
        l_rc = l_currentPhase->mem_ptr();                /* #I */

    /* unknown resource. */
    } else {
        fprintf(l_dbgFd, "Unknown resource type.\n");
    }

    /* Check the return code of the DLPAR operation handler. */
    if (l_rc == DR_API_FAIL) {
        fprintf(l_dbgFd, "DLPAR OPERATION failed!\n");

        /* Let the DR manager know we have to fail the DLPAR operation. */
        l_rc = dr_reconfig(DR_EVENT_FAIL, &dr_info);
        if (l_rc  != 0) {
            perror_msg("dr_reconfig()", errno, __LINE__);
            exit(1);
        }
    }

    fprintf(l_dbgFd, "---end of signal handler.---\n\n");
    fclose(l_dbgFd);
}

/*=====================================================================*/
/*  Main application                                                   */
/*=====================================================================*/

/*=====================================================================*/
/* Some Applicaiton that is registered for DR signals                  */
/*                                                                     */
/* Detailed Description:                                               */
/*     This is a sample program that registers the signal handler      */
/*     function that will response to the SIGRECONFIG signal.          */
/*                                                                     */
/* Output: None                                                        */
/* Inputs: None                                                        */
/*=====================================================================*/
int
main(int argc, char *argv[], char *envp[])
{
    int                 rc;
    struct sigaction    sigact_save, sigact;

/* Start: register this application for a DR signal. */
    if ((rc = sigemptyset(&sigact.sa_mask)) != 0) {
        perror_msg("sigemptyset()", errno, __LINE__);
        exit(1);
    }
    if ((rc = sigemptyset(&sigact_save.sa_mask)) != 0) {
        perror_msg("sigemptyset()", errno, __LINE__);
        exit(1);
    }

    /* register the signal handler function dr_func. */
    sigact.sa_handler = dr_func;
    sigact.sa_flags |= SA_SIGINFO;


    if ((rc = sigaction(SIGRECONFIG, &sigact, &sigact_save)) != 0) { /* #A */
        perror_msg("sigaction()", errno, __LINE__);
        exit(1);
    }
```

```
/* Finish:  registered the signal handler. */
    while (1) { /* #B */
        ;
        /* your applicaiton logic goes here. */
    }

    exit(0);

}
```

C

Dynamic logical partitioning output samples

This appendix provides sample outputs from several debug facilities to be used with dynamic logical partitioning (DLPAR) events explained in 3.10, “Error handling of DLPAR operations” on page 115. There are three debug facilities described in the following:

- Using the syslog facility
- Using the AIX system trace facility
- Using the AIX error log facility

These facilities are to be used to analyze the problem if a DLPAR operation request fails. When writing a DLPAR script or DLPAR-aware application, if the problem persists, then testing the application in a test partition can help isolate the problem.

By familiarizing yourself with these facilities, you can quickly determine the root cause of a DLPAR operation failure.

Using the syslog facility

The syslog facility records the activity of DLPAR operations when it is configured correctly, as explained in “The syslog facility” on page 118.

CPU addition

Example C-1 shows a sample syslog output when a CPU addition DLPAR operation is successfully performed.

Example: C-1 Sample syslog output for a CPU addition request

```
Jul 27 16:49:51 thimblelp4 syslogd: restart
Jul 27 16:50:25 thimblelp4 DRMGR:  ==== Start: CPU addition operation ====
Jul 27 16:50:25 thimblelp4 DRMGR: Cpu: 0x1002 has been unisolated and allocated
Jul 27 16:50:25 thimblelp4 DRMGR: Starting CHECK phase for cpu Add operation.
Jul 27 16:50:25 thimblelp4 DRMGR: Phase CHECK started for scripts,kernel extensions and
applications.
Jul 27 16:50:25 thimblelp4 DRMGR: Starting CHECK phase for Scripts.
Jul 27 16:50:25 thimblelp4 DRMGR: Completed the phase for Scripts.
Jul 27 16:50:26 thimblelp4 DRMGR: Starting the phase for kernel extensions.
Jul 27 16:50:26 thimblelp4 DRMGR: Completed the phase for kernel extensions.
Jul 27 16:50:26 thimblelp4 DRMGR: Starting the phase for application signal handlers.
Jul 27 16:50:26 thimblelp4 DRMGR: Completed the phase for kernel extensions.
Jul 27 16:50:26 thimblelp4 DRMGR: Starting PRE phase.
Jul 27 16:50:26 thimblelp4 DRMGR: Phase PRE started for scripts,kernel extensions and
applications.
Jul 27 16:50:26 thimblelp4 DRMGR: Starting PRE phase for scripts.
Jul 27 16:50:26 thimblelp4 DRMGR: Completed the phase for Scripts.
Jul 27 16:50:26 thimblelp4 DRMGR: Starting the phase for application signal handlers.
Jul 27 16:50:26 thimblelp4 DRMGR: Completed the phase for kernel extensions.
Jul 27 16:50:27 thimblelp4 DRMGR: kernel operations complete
Jul 27 16:50:27 thimblelp4 DRMGR: firmware operations complete
Jul 27 16:50:27 thimblelp4 DRMGR: ODM update complete
Jul 27 16:50:27 thimblelp4 DRMGR: Starting POST phase.
Jul 27 16:50:27 thimblelp4 DRMGR: Phase POST started for scripts,kernel extensions and
applications.
Jul 27 16:50:27 thimblelp4 DRMGR: Starting the phase for application signal handlers.
Jul 27 16:50:27 thimblelp4 DRMGR: Completed the phase for kernel extensions.
Jul 27 16:50:27 thimblelp4 DRMGR: Starting POST phase for scripts.
Jul 27 16:50:27 thimblelp4 DRMGR: Completed the phase for Scripts.
Jul 27 16:50:27 thimblelp4 DRMGR:  ~~~~ End: CPU addition operation ~~~~
```

CPU removal

Example C-2 shows a sample syslog output when a CPU removal DLPAR operation is successfully performed.

Example: C-2 Sample syslog output for a CPU removal request

```
Jul 27 16:47:58 thimblelp4 syslogd: restart
Jul 27 16:48:08 thimblelp4 DRMGR:  ==== Start: CPU Removal operation ====
Jul 27 16:48:08 thimblelp4 DRMGR: Starting CHECK phase for cpu Remove operation.
Jul 27 16:48:08 thimblelp4 DRMGR: Phase CHECK started for scripts,kernel extensions and
applications.
Jul 27 16:48:08 thimblelp4 DRMGR: Starting CHECK phase for Scripts.
Jul 27 16:48:08 thimblelp4 DRMGR: Completed the phase for Scripts.
Jul 27 16:48:08 thimblelp4 DRMGR: Starting the phase for kernel extensions.
Jul 27 16:48:08 thimblelp4 DRMGR: Completed the phase for kernel extensions.
Jul 27 16:48:08 thimblelp4 DRMGR: Starting the phase for application signal handlers.
Jul 27 16:48:08 thimblelp4 DRMGR: Completed the phase for kernel extensions.
Jul 27 16:48:08 thimblelp4 DRMGR: Starting PRE phase.
Jul 27 16:48:08 thimblelp4 DRMGR: Phase PRE started for scripts,kernel extensions and
applications.
Jul 27 16:48:08 thimblelp4 DRMGR: Starting PRE phase for scripts.
Jul 27 16:48:08 thimblelp4 DRMGR: Completed the phase for Scripts.
Jul 27 16:48:08 thimblelp4 DRMGR: Starting the phase for application signal handlers.
Jul 27 16:48:08 thimblelp4 DRMGR: Completed the phase for kernel extensions.
Jul 27 16:48:08 thimblelp4 DRMGR: kernel operations complete
Jul 27 16:48:08 thimblelp4 DRMGR: Cpu: 0x1002 has been isolated and unallocated
Jul 27 16:48:08 thimblelp4 DRMGR: Firmware operations complete
Jul 27 16:48:09 thimblelp4 DRMGR: ODM update complete
Jul 27 16:48:09 thimblelp4 DRMGR: Starting POST phase.
Jul 27 16:48:09 thimblelp4 DRMGR: Phase POST started for scripts,kernel extensions and
applications.
Jul 27 16:48:09 thimblelp4 DRMGR: Starting the phase for application signal handlers.
Jul 27 16:48:09 thimblelp4 DRMGR: Completed the phase for kernel extensions.
Jul 27 16:48:09 thimblelp4 DRMGR: Starting POST phase for scripts.
Jul 27 16:48:09 thimblelp4 DRMGR: Completed the phase for Scripts.
Jul 27 16:48:09 thimblelp4 DRMGR:  ~~~~ End: CPU Removal operation ~~~~
```

Memory addition

Example C-3 shows a sample syslog output when a memory addition DLPAR operation is successfully performed.

Example: C-3 Sample syslog output for a memory addition request

```
Jul 27 16:49:51 thimblelp4 syslogd: restart
Jul 27 16:51:34 thimblelp4 DRMGR:  ==== Start: MEM Addition operation ====
Jul 27 16:51:34 thimblelp4 DRMGR: Configured LMB addr: 0x0
Jul 27 16:51:34 thimblelp4 DRMGR: Total Megabytes to add is 0
Jul 27 16:51:34 thimblelp4 DRMGR: Starting CHECK phase for mem Add operation.
Jul 27 16:51:34 thimblelp4 DRMGR: Phase CHECK started for scripts,kernel extensions and
applications.
Jul 27 16:51:34 thimblelp4 DRMGR: Starting CHECK phase for Scripts.
Jul 27 16:51:34 thimblelp4 DRMGR: Completed the phase for Scripts.
Jul 27 16:51:34 thimblelp4 DRMGR: Starting the phase for kernel extensions.
Jul 27 16:51:34 thimblelp4 DRMGR: Completed the phase for kernel extensions.
Jul 27 16:51:34 thimblelp4 DRMGR: Starting the phase for application signal handlers.
Jul 27 16:51:34 thimblelp4 DRMGR: Completed the phase for kernel extensions.
Jul 27 16:51:34 thimblelp4 DRMGR: Starting PRE phase.
Jul 27 16:51:34 thimblelp4 DRMGR: Phase PRE started for scripts,kernel extensions and
applications.
Jul 27 16:51:34 thimblelp4 DRMGR: Starting PRE phase for scripts.
Jul 27 16:51:34 thimblelp4 DRMGR: Completed the phase for Scripts.
Jul 27 16:51:34 thimblelp4 DRMGR: Starting the phase for application signal handlers.
Jul 27 16:51:34 thimblelp4 DRMGR: Completed the phase for kernel extensions.
Jul 27 16:51:35 thimblelp4 DRMGR: ODM operations complete
Jul 27 16:51:35 thimblelp4 DRMGR: Starting POST phase.
Jul 27 16:51:35 thimblelp4 DRMGR: Phase POST started for scripts,kernel extensions and
applications.
Jul 27 16:51:35 thimblelp4 DRMGR: Starting the phase for application signal handlers.
Jul 27 16:51:35 thimblelp4 DRMGR: Completed the phase for kernel extensions.
Jul 27 16:51:35 thimblelp4 DRMGR: Starting POST phase for scripts.
Jul 27 16:51:35 thimblelp4 DRMGR: Completed the phase for Scripts.
Jul 27 16:51:35 thimblelp4 DRMGR:  ~~~~ End: DR operation ~~~~
```

Memory removal

Example C-4 shows a sample syslog output when a memory removal DLPAR operation is successfully performed.

Example: C-4 Sample syslog output for a memory removal request

```
Jul 27 16:49:51 thimblelp4 syslogd: restart
Jul 27 16:51:07 thimblelp4 DRMGR:  ==== Start: MEM Removal operation ====
Jul 27 16:51:07 thimblelp4 DRMGR: Starting CHECK phase for mem Remove operation.
Jul 27 16:51:07 thimblelp4 DRMGR: Phase CHECK started for scripts,kernel extensions and
applications.
Jul 27 16:51:07 thimblelp4 DRMGR: Starting CHECK phase for Scripts.
Jul 27 16:51:07 thimblelp4 DRMGR: Completed the phase for Scripts.
Jul 27 16:51:07 thimblelp4 DRMGR: Starting the phase for kernel extensions.
Jul 27 16:51:07 thimblelp4 DRMGR: Completed the phase for kernel extensions.
Jul 27 16:51:07 thimblelp4 DRMGR: Starting the phase for application signal handlers.
Jul 27 16:51:07 thimblelp4 DRMGR: Completed the phase for kernel extensions.
Jul 27 16:51:07 thimblelp4 DRMGR: Starting PRE phase.
Jul 27 16:51:07 thimblelp4 DRMGR: Phase PRE started for scripts,kernel extensions and
applications.
Jul 27 16:51:07 thimblelp4 DRMGR: Starting PRE phase for scripts.
Jul 27 16:51:07 thimblelp4 DRMGR: Completed the phase for Scripts.
Jul 27 16:51:07 thimblelp4 DRMGR: Starting the phase for application signal handlers.
Jul 27 16:51:07 thimblelp4 DRMGR: Completed the phase for kernel extensions.
Jul 27 16:51:08 thimblelp4 DRMGR: LMB index:0xf has been sucessfully removed
Jul 27 16:51:08 thimblelp4 DRMGR: Firmware operations complete
Jul 27 16:51:08 thimblelp4 DRMGR: ODM operations complete
Jul 27 16:51:08 thimblelp4 DRMGR: Starting POST phase.
Jul 27 16:51:08 thimblelp4 DRMGR: Phase POST started for scripts,kernel extensions and
applications.
Jul 27 16:51:08 thimblelp4 DRMGR: Starting the phase for application signal handlers.
Jul 27 16:51:08 thimblelp4 DRMGR: Completed the phase for kernel extensions.
Jul 27 16:51:08 thimblelp4 DRMGR: Starting POST phase for scripts.
Jul 27 16:51:08 thimblelp4 DRMGR: Completed the phase for Scripts.
Jul 27 16:51:08 thimblelp4 DRMGR:  ~~~~ End: DR operation ~~~~
```

Using the AIX system trace facility

The AIX system trace facility records the detailed activity of DLPAR operations when it is configured correctly, as explained in “AIX system trace facility” on page 119.

Note: The trace hook ID for DLPAR operations is 38F.

CPU addition trace output

Example C-5 shows a sample system trace output when a CPU addition DLPAR operation is successfully performed.

Example: C-5 Sample system trace output for a CPU addition request

```
ID     ELAPSED_SEC     DELTA_MSEC   APPL    SYSCALL KERNEL  INTERRUPT

001    0.000000000       0.000000                   TRACE ON channel 0
                                                    Mon Sep 23 11:40:52 2002
38F   73.305720033   73305.720033                   DYNAMIC RECONFIG: Dr_register: DR
Operation: 0000000000000008 FORCE Option: 0000
38F   73.305723429       0.003396                   DYNAMIC RECONFIG: get_user_data: DR
Operation: 0000000000000008 input data: 000000002FF21EF8
38F   73.305725392       0.001963                   DYNAMIC RECONFIG: Addcpu_validate: DR
Phase: 0000 Input: F00000002FF3A4D8
38F   73.305727203       0.001811                   DYNAMIC RECONFIG: Register_dr_event: DR
Operation: 40000000
38F   73.305729377       0.002174                   DYNAMIC RECONFIG: dr_callout: DR Callout
index: 0003 DR Phase: 0001
38F   73.305729602       0.000225                   DYNAMIC RECONFIG: Addcpu_validate: DR
Phase: 0001 Input: 0000000000FB73C0
38F   73.306332726      0.603124                   DYNAMIC RECONFIG: Dr_notify: DR Phase: 0002
Flags: 0001 Timeout in secs: 0000 Input: 0000000000000000
38F   73.306334115       0.001389                   DYNAMIC RECONFIG: Validate_notify: DR
Phase: 0002 Flags: 0001
38F   73.306334355       0.000240                   DYNAMIC RECONFIG: validate_phase: Current
Phase: 0001 Requested Phase: 0002 Flags: 0001
38F   73.306336885       0.002530                   DYNAMIC RECONFIG: Run_notify: Perform DR
Check/Pre/Post/Posterror Phases
38F   73.306337096       0.000211                   DYNAMIC RECONFIG: dr_callout: DR Callout
index: 0003 DR Phase: 0002
38F   73.306538971      0.201875                   DYNAMIC RECONFIG: Dr_notify: DR Phase: 0002
Flags: 0004 Timeout in secs: 003C Input: 0000000000000000
38F   73.306539560       0.000589                   DYNAMIC RECONFIG: Validate_notify: DR
Phase: 0002 Flags: 0004
38F   73.306539800       0.000240                   DYNAMIC RECONFIG: validate_phase: Current
Phase: 0002 Requested Phase: 0002 Flags: 0004
```

```
38F   73.306545297       0.005497                  DYNAMIC RECONFIG: Run_notify: Perform DR
Check/Pre/Post/Posterror Phases
38F   73.306546257       0.000960                  DYNAMIC RECONFIG: Invoke Reconfig Handlers:
action: 0001
38F   73.308395576       1.849319                  DYNAMIC RECONFIG: Invoke Reconfig Handlers:
Number of reconfig handlers waiting for: 0000
38F   73.308396070       0.000494                  DYNAMIC RECONFIG: Invoke Reconfig Handlers:
All reconfig handlers completed, Status: 0000
38F   73.308536626       0.140556                  DYNAMIC RECONFIG: Dr_notify: DR Phase: 0002
Flags: 0002 Timeout in secs: 0000 Input: 0000000000000000
38F   73.308537287       0.000661                  DYNAMIC RECONFIG: Validate_notify: DR
Phase: 0002 Flags: 0002
38F   73.308537527       0.000240                  DYNAMIC RECONFIG: validate_phase: Current
Phase: 0002 Requested Phase: 0002 Flags: 0002
38F   73.308538189       0.000662                  DYNAMIC RECONFIG: Run_notify: Perform DR
Check/Pre/Post/Posterror Phases
38F   73.308538414       0.000225                  DYNAMIC RECONFIG: dr_send_signal: Posting
signal (003A) to all processes catching
38F   73.308730538       0.192124                  DYNAMIC RECONFIG: Dr_reconfig: Flags: 0001
DR Info: 000000002FF22610 DR Operation: 0008 DR Phase: 0002
38F   73.308737235       0.006697                  DYNAMIC RECONFIG: dr_send_signal: Number of
processes posted: 0001
38F   83.309400202   10000.662967                  DYNAMIC RECONFIG: Dr_notify: DR Phase: 0003
Flags: 0001 Timeout in secs: 0000 Input: 0000000000000000
38F   83.309402034       0.001832                  DYNAMIC RECONFIG: Validate_notify: DR
Phase: 0003 Flags: 0001
38F   83.309402260       0.000226                  DYNAMIC RECONFIG: validate_phase: Current
Phase: 0002 Requested Phase: 0003 Flags: 0001
38F   83.309404165       0.001905                  DYNAMIC RECONFIG: Run_notify: Perform DR
Check/Pre/Post/Posterror Phases
38F   83.309404703       0.000538                  DYNAMIC RECONFIG: dr_callout: DR Callout
index: 0003 DR Phase: 0003
38F   83.309406957       0.002254                  DYNAMIC RECONFIG: Addcpu_pre: Logical CPU
coming online: 0002
38F   83.309607727       0.200770                  DYNAMIC RECONFIG: Dr_notify: DR Phase: 0003
Flags: 0002 Timeout in secs: 0000 Input: 0000000000000000
38F   83.309608258       0.000531                  DYNAMIC RECONFIG: Validate_notify: DR
Phase: 0003 Flags: 0002
38F   83.309608483       0.000225                  DYNAMIC RECONFIG: validate_phase: Current
Phase: 0003 Requested Phase: 0003 Flags: 0002
38F   83.309608825       0.000342                  DYNAMIC RECONFIG: Run_notify: Perform DR
Check/Pre/Post/Posterror Phases
38F   83.309610301       0.001476                  DYNAMIC RECONFIG: dr_send_signal: Posting
signal (003A) to all processes catching
38F   83.309738306       0.128005                  DYNAMIC RECONFIG: Dr_reconfig: Flags: 0001
DR Info: 000000002FF22610 DR Operation: 0008 DR Phase: 0003
38F   83.309751780       0.013474                  DYNAMIC RECONFIG: dr_send_signal: Number of
processes posted: 0001
```

```
38F   93.310361840   10000.610060                  DYNAMIC RECONFIG: Dr_notify: DR Phase: 0005
Flags: 0001 Timeout in secs: 0000 Input: 0000000000000000
38F   93.310363963       0.002123                  DYNAMIC RECONFIG: Validate_notify: DR
Phase: 0005 Flags: 0001
38F   93.310364254       0.000291                  DYNAMIC RECONFIG: validate_phase: Current
Phase: 0003 Requested Phase: 0005 Flags: 0001
38F   93.310365163       0.000909                  DYNAMIC RECONFIG: Kernel_notify: Perform DR
Kernel Phase
38F   93.310365454       0.000291                  DYNAMIC RECONFIG: dr_callout: DR Callout
index: 0003 DR Phase: 0005
38F   93.310367432       0.001978                  DYNAMIC RECONFIG: Addcpu_doit: Logical CPU:
0002 Physical ID: 0015
38F   93.310371860       0.004428                  DYNAMIC RECONFIG: Register_dr_event: DR
Operation: 80000008
38F   93.310372747       0.000887                  DYNAMIC RECONFIG: Invoke Reconfig Handlers:
action: 0002
38F   93.310394779       0.022032                  DYNAMIC RECONFIG: Invoke Reconfig Handlers:
Number of reconfig handlers waiting for: 0000
38F   93.310395245       0.000466                  DYNAMIC RECONFIG: Invoke Reconfig Handlers:
All reconfig handlers completed, Status: 0000
38F   93.310416484       0.021239                  DYNAMIC RECONFIG: Call MPC freeze handler
01
38F   93.310445504       0.029020                  DYNAMIC RECONFIG: Start_bs_proc: Starting a
new cpu: Physical ID: 0015 Gserver: 00000000000000FF Server: 0000000000000015
38F   93.991532779     681.087275                  DYNAMIC RECONFIG: Invoke Reconfig Handlers:
action: 0004
38F   93.991536952       0.004173                  DYNAMIC RECONFIG: Invoke Reconfig Handlers:
Number of reconfig handlers waiting for: 0000
38F   93.991537709       0.000757                  DYNAMIC RECONFIG: Invoke Reconfig Handlers:
All reconfig handlers completed, Status: 0000
38F   93.991556098       0.018389                  DYNAMIC RECONFIG: Unregister_dr_event: DR
Operation: 80000008
38F   94.015313247      23.757149                  DYNAMIC RECONFIG: Dr_notify: DR Phase: 0006
Flags: 0001 Timeout in secs: 0000 Input: 0000000000000000
38F   94.015314549       0.001302                  DYNAMIC RECONFIG: Validate_notify: DR
Phase: 0006 Flags: 0001
38F   94.015314905       0.000356                  DYNAMIC RECONFIG: validate_phase: Current
Phase: 0005 Requested Phase: 0006 Flags: 0001
38F   94.015315930       0.001025                  DYNAMIC RECONFIG: Run_notify: Perform DR
Check/Pre/Post/Posterror Phases
38F   94.015316366       0.000436                  DYNAMIC RECONFIG: dr_callout: DR Callout
index: 0003 DR Phase: 0006
38F   94.015382834       0.066468                  DYNAMIC RECONFIG: Dr_notify: DR Phase: 0006
Flags: 0002 Timeout in secs: 0000 Input: 0000000000000000
38F   94.015383336       0.000502                  DYNAMIC RECONFIG: Validate_notify: DR
Phase: 0006 Flags: 0002
38F   94.015383561       0.000225                  DYNAMIC RECONFIG: validate_phase: Current
Phase: 0006 Requested Phase: 0006 Flags: 0002
```

```
38F   94.015383903       0.000342                   DYNAMIC RECONFIG: Run_notify: Perform DR
Check/Pre/Post/Posterror Phases
38F   94.015384768       0.000865                   DYNAMIC RECONFIG: dr_send_signal: Posting
signal (003A) to all processes catching
38F   94.015521608       0.136840                   DYNAMIC RECONFIG: Dr_reconfig: Flags: 0001
DR Info: 000000002FF22610 DR Operation: 0008 DR Phase: 0006
38F   94.015556140       0.034532                   DYNAMIC RECONFIG: dr_send_signal: Number of
processes posted: 0001
38F  104.016466326   10000.910186                   DYNAMIC RECONFIG: Dr_unregister:
Unregistering DR operation
38F  104.016468027       0.001701                   DYNAMIC RECONFIG: dr_callout: DR Callout
index: 0003 DR Phase: 0009
38F  104.016468507       0.000480                   DYNAMIC RECONFIG: Clearing DR Kernel
Data...
38F  104.016470936       0.002429                   DYNAMIC RECONFIG: Unregister_dr_event: DR
Operation: 40000000
002  322.179517597  218163.046661                   TRACE OFF channel 0000 Mon Sep 23 11:46:14
2002
```

CPU removal trace output

Example C-6 shows a sample system trace output when a CPU removal DLPAR operation is successfully performed.

Example: C-6 Sample system trace output for a CPU removal request

```
ID     ELAPSED_SEC     DELTA_MSEC   APPL    SYSCALL KERNEL  INTERRUPT

001    0.000000000       0.000000                   TRACE ON channel 0
                                                    Sat Jul 27 17:22:09 2002
38F    8.210889322    8210.889322                   DYNAMIC RECONFIG: Dr_register: DR
Operation: 0000000000000004 FORCE Option: 0000
38F    8.210890417       0.001095                   DYNAMIC RECONFIG: get_user_data: DR
Operation: 0000000000000004 input data: 000000002FF22AF0
38F    8.210892128       0.001711                   DYNAMIC RECONFIG: Rmcpu_validate: DR Phase:
0000 CPU id: 0001 CPU Type: 0002
38F    8.210892971       0.000843                   DYNAMIC RECONFIG: Register_dr_event: DR
Operation: 40000000
38F    8.210896029       0.003058                   DYNAMIC RECONFIG: dr_callout: DR Callout
index: 0002 DR Phase: 0001
38F    8.210896238       0.000209                   DYNAMIC RECONFIG: Rmcpu_validate: DR Phase:
0001 CPU id: 0001 CPU Type: 0002
38F    8.212724871       1.828633                   DYNAMIC RECONFIG: Dr_notify: DR Phase: 0002
Flags: 0001 Timeout in secs: 0000 Input: 0000000000000000
38F    8.212725498       0.000627                   DYNAMIC RECONFIG: Validate_notify: DR
Phase: 0002 Flags: 0001
38F    8.212726175       0.000677                   DYNAMIC RECONFIG: validate_phase: Current
Phase: 0001 Requested Phase: 0002 Flags: 0001
```

```
38F    8.212728618       0.002443                   DYNAMIC RECONFIG: Run_notify: Perform DR
Check/Pre/Post/Posterror Phases
38F    8.212728821       0.000203                   DYNAMIC RECONFIG: dr_callout: DR Callout
index: 0002 DR Phase: 0002
38F    8.212836663       0.107842                  DYNAMIC RECONFIG: Dr_notify: DR Phase: 0002
Flags: 0004 Timeout in secs: 003C Input: 0000000000000000
38F    8.212837119       0.000456                   DYNAMIC RECONFIG: Validate_notify: DR
Phase: 0002 Flags: 0004
38F    8.212837322       0.000203                   DYNAMIC RECONFIG: validate_phase: Current
Phase: 0002 Requested Phase: 0002 Flags: 0004
38F    8.212842109       0.004787                   DYNAMIC RECONFIG: Run_notify: Perform DR
Check/Pre/Post/Posterror Phases
38F    8.212842392       0.000283                  DYNAMIC RECONFIG: Invoke Reconfig Handlers:
action: 0010
38F    8.212843444       0.001052                  DYNAMIC RECONFIG: Invoke Reconfig Handlers:
Number of reconfig handlers waiting for: 0000
38F    8.212844231       0.000787                  DYNAMIC RECONFIG: Invoke Reconfig Handlers:
All reconfig handlers completed, Status: 0000
38F    8.212914941       0.070710                  DYNAMIC RECONFIG: Dr_notify: DR Phase: 0002
Flags: 0002 Timeout in secs: 0000 Input: 0000000000000000
38F    8.212915409       0.000468                   DYNAMIC RECONFIG: Validate_notify: DR
Phase: 0002 Flags: 0002
38F    8.212915605       0.000196                   DYNAMIC RECONFIG: validate_phase: Current
Phase: 0002 Requested Phase: 0002 Flags: 0002
38F    8.212915858       0.000253                   DYNAMIC RECONFIG: Run_notify: Perform DR
Check/Pre/Post/Posterror Phases
38F    8.212916116       0.000258                   DYNAMIC RECONFIG: dr_send_signal: Posting
signal (003A) to all processes catching
38F    8.212995181       0.079065                  DYNAMIC RECONFIG: dr_send_signal: Number of
processes posted: 0000
38F    8.213781181       0.786000                  DYNAMIC RECONFIG: Dr_notify: DR Phase: 0003
Flags: 0001 Timeout in secs: 0000 Input: 0000000000000000
38F    8.213781637       0.000456                   DYNAMIC RECONFIG: Validate_notify: DR
Phase: 0003 Flags: 0001
38F    8.213781858       0.000221                   DYNAMIC RECONFIG: validate_phase: Current
Phase: 0002 Requested Phase: 0003 Flags: 0001
38F    8.213782153       0.000295                   DYNAMIC RECONFIG: Run_notify: Perform DR
Check/Pre/Post/Posterror Phases
38F    8.213782332       0.000179                   DYNAMIC RECONFIG: dr_callout: DR Callout
index: 0002 DR Phase: 0003
38F    8.213783243       0.000911                  DYNAMIC RECONFIG: Rmcpu_pre: DR Phase: 0003
Logical CPU id: 0001
38F    8.213980321       0.197078                  DYNAMIC RECONFIG: Dr_notify: DR Phase: 0003
Flags: 0002 Timeout in secs: 0000 Input: 0000000000000000
38F    8.213980795       0.000474                   DYNAMIC RECONFIG: Validate_notify: DR
Phase: 0003 Flags: 0002
38F    8.213980992       0.000197                   DYNAMIC RECONFIG: validate_phase: Current
Phase: 0003 Requested Phase: 0003 Flags: 0002
```

```
38F    8.213981262       0.000270                  DYNAMIC RECONFIG: Run_notify: Perform DR
Check/Pre/Post/Posterror Phases
38F    8.213981515       0.000253                  DYNAMIC RECONFIG: dr_send_signal: Posting
signal (003A) to all processes catching
38F    8.214019626       0.038111                 DYNAMIC RECONFIG: dr_send_signal: Number of
processes posted: 0000
38F    8.214186223       0.166597                 DYNAMIC RECONFIG: Dr_notify: DR Phase: 0005
Flags: 0001 Timeout in secs: 0000 Input: 0000000000000000
38F    8.214186709       0.000486                  DYNAMIC RECONFIG: Validate_notify: DR
Phase: 0005 Flags: 0001
38F    8.214187109       0.000400                  DYNAMIC RECONFIG: validate_phase: Current
Phase: 0003 Requested Phase: 0005 Flags: 0001
38F    8.214187429       0.000320                 DYNAMIC RECONFIG: Kernel_notify: Perform DR
Kernel Phase
38F    8.214187601       0.000172                  DYNAMIC RECONFIG: dr_callout: DR Callout
index: 0002 DR Phase: 0005
38F    8.214189318       0.001717                  DYNAMIC RECONFIG: Rmcpu_doit: DR Phase:
0005 CPU Guard Operation: 0000
38F    8.214192929       0.003611                  DYNAMIC RECONFIG: Register_dr_event: DR
Operation: 80000004
38F    8.214193618       0.000689                  DYNAMIC RECONFIG: Rmcpu_doit: Controlling
LCPU: 0000 Highest Bind cpuid: 0001
38F    8.214213369       0.019751                  DYNAMIC RECONFIG: Rmcpu_doit: Invoke HA
Handlers...
38F    8.218559640       4.346271                 DYNAMIC RECONFIG: Invoke Reconfig Handlers:
action: 0020
38F    8.218561874       0.002234                 DYNAMIC RECONFIG: Invoke Reconfig Handlers:
Number of reconfig handlers waiting for: 0000
38F    8.218562286       0.000412                 DYNAMIC RECONFIG: Invoke Reconfig Handlers:
All reconfig handlers completed, Status: 0000
38F    8.218562489       0.000203                  DYNAMIC RECONFIG: Migrating all
PROCESSOR_CLASS_ANY work from the cpu being removed
38F    8.218778851       0.216362                  DYNAMIC RECONFIG: Initializing/Rerouting
the Interrupts... From Physical CPU: 0013 To Physical CPU: 0011 Phase: 0001 Flags: 0001
38F    8.218966349       0.187498                  DYNAMIC RECONFIG: Rmcpu_doit: Disable
Decrementer...
38F    8.218980956       0.014607                  DYNAMIC RECONFIG: Call MPC remove handler
01
38F    8.218982174       0.001218                  DYNAMIC RECONFIG: Initializing/Rerouting
the Interrupts... From Physical CPU: 0013 To Physical CPU: 0011 Phase: 0002 Flags: 0002
38F    8.218982470       0.000296                  DYNAMIC RECONFIG: Rmcpu_doit: Enable
Decrementer...
38F    8.225044879       6.062409                 DYNAMIC RECONFIG: DR: Stopping logical CPU:
0001
38F    8.226081506       1.036627                  DYNAMIC RECONFIG: Updating System
Topology...
38F    8.226114591       0.033085                  DYNAMIC RECONFIG: Move_threads: Moving
threads from logical cpu 0001 to 0000
```

```
38F    8.226234647       0.120056                  DYNAMIC RECONFIG: migrate_watchdogs: From
LCPU: 0001 To LCPU: 0000
38F    8.226243790       0.009143                  DYNAMIC RECONFIG: Invoke Reconfig Handlers:
action: 0040
38F    8.226244399       0.000609                  DYNAMIC RECONFIG: Invoke Reconfig Handlers:
Number of reconfig handlers waiting for: 0000
38F    8.226244916       0.000517                  DYNAMIC RECONFIG: Invoke Reconfig Handlers:
All reconfig handlers completed, Status: 0000
38F    8.226245137       0.000221                  DYNAMIC RECONFIG: Rmcpu_doit: DR CPU
Removal: CPU Guard: 0000 Status: 0000
38F    8.226245839       0.000702                  DYNAMIC RECONFIG: Unregister_dr_event: DR
Operation: 80000004
38F    8.407728629     181.482790                  DYNAMIC RECONFIG: Dr_notify: DR Phase: 0006
Flags: 0001 Timeout in secs: 0000 Input: 0000000000000000
38F    8.407729373       0.000744                  DYNAMIC RECONFIG: Validate_notify: DR
Phase: 0006 Flags: 0001
38F    8.407729650       0.000277                  DYNAMIC RECONFIG: validate_phase: Current
Phase: 0005 Requested Phase: 0006 Flags: 0001
38F    8.407731693       0.002043                  DYNAMIC RECONFIG: Run_notify: Perform DR
Check/Pre/Post/Posterror Phases
38F    8.407731878       0.000185                  DYNAMIC RECONFIG: dr_callout: DR Callout
index: 0002 DR Phase: 0006
38F    8.407907882       0.176004                  DYNAMIC RECONFIG: Dr_notify: DR Phase: 0006
Flags: 0002 Timeout in secs: 0000 Input: 0000000000000000
38F    8.407908350       0.000468                  DYNAMIC RECONFIG: Validate_notify: DR
Phase: 0006 Flags: 0002
38F    8.407908553       0.000203                  DYNAMIC RECONFIG: validate_phase: Current
Phase: 0006 Requested Phase: 0006 Flags: 0002
38F    8.407908848       0.000295                  DYNAMIC RECONFIG: Run_notify: Perform DR
Check/Pre/Post/Posterror Phases
38F    8.407909297       0.000449                  DYNAMIC RECONFIG: dr_send_signal: Posting
signal (003A) to all processes catching
38F    8.407989156       0.079859                  DYNAMIC RECONFIG: dr_send_signal: Number of
processes posted: 0000
38F    8.409821659       1.832503                  DYNAMIC RECONFIG: Dr_unregister:
Unregistering DR operation
38F    8.409823339       0.001680                  DYNAMIC RECONFIG: dr_callout: DR Callout
index: 0002 DR Phase: 0009
38F    8.409823727       0.000388                  DYNAMIC RECONFIG: Clearing DR Kernel
Data...
38F    8.409826052       0.002325                  DYNAMIC RECONFIG: Unregister_dr_event: DR
Operation: 40000000
002   10.460418340    2050.592288                  TRACE OFF channel 0000 Sat Jul 27 17:22:19
2002
```

Memory addition trace output

Example C-7 shows a sample system trace output when a memory addition DLPAR operation is successfully performed.

Example: C-7 Sample system trace output for a memory addition request

```
ID     ELAPSED_SEC     DELTA_MSEC   APPL    SYSCALL KERNEL  INTERRUPT

001    0.000000000       0.000000                   TRACE ON channel 0
                                                    Sat Jul 27 17:21:13 2002
38F    5.368745028    5368.745028                   DYNAMIC RECONFIG: Dr_register: DR
Operation: 0000000000000002 FORCE Option: 0000
38F    5.368746271       0.001243                   DYNAMIC RECONFIG: get_user_data: DR
Operation: 0000000000000002 input data: 000000002FF22988
38F    5.368748763       0.002492                   DYNAMIC RECONFIG: Register_dr_event: DR
Operation: 40000000
38F    5.368751162       0.002399                   DYNAMIC RECONFIG: dr_callout: DR Callout
index: 0001 DR Phase: 0001
38F    5.370106721       1.355559                  DYNAMIC RECONFIG: Dr_notify: DR Phase: 0002
Flags: 0001 Timeout in secs: 0000 Input: 0000000000000000
38F    5.370107638       0.000917                   DYNAMIC RECONFIG: Validate_notify: DR
Phase: 0002 Flags: 0001
38F    5.370108136       0.000498                   DYNAMIC RECONFIG: validate_phase: Current
Phase: 0001 Requested Phase: 0002 Flags: 0001
38F    5.370110819       0.002683                   DYNAMIC RECONFIG: Run_notify: Perform DR
Check/Pre/Post/Posterror Phases
38F    5.370111003       0.000184                   DYNAMIC RECONFIG: dr_callout: DR Callout
index: 0001 DR Phase: 0002
38F    5.370229053       0.118050                  DYNAMIC RECONFIG: Dr_notify: DR Phase: 0002
Flags: 0004 Timeout in secs: 003C Input: 0000000000000000
38F    5.370229515       0.000462                   DYNAMIC RECONFIG: Validate_notify: DR
Phase: 0002 Flags: 0004
38F    5.370229724       0.000209                   DYNAMIC RECONFIG: validate_phase: Current
Phase: 0002 Requested Phase: 0002 Flags: 0004
38F    5.370232068       0.002344                   DYNAMIC RECONFIG: Run_notify: Perform DR
Check/Pre/Post/Posterror Phases
38F    5.370232345       0.000277                  DYNAMIC RECONFIG: Invoke Reconfig Handlers:
action: 0100
38F    5.370234357       0.002012                  DYNAMIC RECONFIG: Invoke Reconfig Handlers:
Number of reconfig handlers waiting for: 0000
38F    5.370234806       0.000449                  DYNAMIC RECONFIG: Invoke Reconfig Handlers:
All reconfig handlers completed, Status: 0000
38F    5.370313004       0.078198                  DYNAMIC RECONFIG: Dr_notify: DR Phase: 0002
Flags: 0002 Timeout in secs: 0000 Input: 0000000000000000
38F    5.370313490       0.000486                   DYNAMIC RECONFIG: Validate_notify: DR
Phase: 0002 Flags: 0002
38F    5.370313705       0.000215                   DYNAMIC RECONFIG: validate_phase: Current
Phase: 0002 Requested Phase: 0002 Flags: 0002
```

```
38F    5.370314013       0.000308                   DYNAMIC RECONFIG: Run_notify: Perform DR
Check/Pre/Post/Posterror Phases
38F    5.370314241       0.000228                   DYNAMIC RECONFIG: dr_send_signal: Posting
signal (003A) to all processes catching
38F    5.370385443       0.071202                  DYNAMIC RECONFIG: dr_send_signal: Number of
processes posted: 0000
38F    5.371572355       1.186912                  DYNAMIC RECONFIG: Dr_notify: DR Phase: 0003
Flags: 0001 Timeout in secs: 0000 Input: 0000000000000000
38F    5.371572823       0.000468                   DYNAMIC RECONFIG: Validate_notify: DR
Phase: 0003 Flags: 0001
38F    5.371573063       0.000240                   DYNAMIC RECONFIG: validate_phase: Current
Phase: 0002 Requested Phase: 0003 Flags: 0001
38F    5.371573352       0.000289                   DYNAMIC RECONFIG: Run_notify: Perform DR
Check/Pre/Post/Posterror Phases
38F    5.371573555       0.000203                   DYNAMIC RECONFIG: dr_callout: DR Callout
index: 0001 DR Phase: 0003
38F    5.371688418       0.114863                  DYNAMIC RECONFIG: Dr_notify: DR Phase: 0003
Flags: 0002 Timeout in secs: 0000 Input: 0000000000000000
38F    5.371688892       0.000474                   DYNAMIC RECONFIG: Validate_notify: DR
Phase: 0003 Flags: 0002
38F    5.371689199       0.000307                   DYNAMIC RECONFIG: validate_phase: Current
Phase: 0003 Requested Phase: 0003 Flags: 0002
38F    5.371689446       0.000247                   DYNAMIC RECONFIG: Run_notify: Perform DR
Check/Pre/Post/Posterror Phases
38F    5.371689679       0.000233                   DYNAMIC RECONFIG: dr_send_signal: Posting
signal (003A) to all processes catching
38F    5.371726129       0.036450                  DYNAMIC RECONFIG: dr_send_signal: Number of
processes posted: 0000
38F    5.373202390       1.476261                  DYNAMIC RECONFIG: Dr_notify: DR Phase: 0005
Flags: 0009 Timeout in secs: 0000 Input: 000000002FF22988
38F    5.373202857       0.000467                   DYNAMIC RECONFIG: Validate_notify: DR
Phase: 0005 Flags: 0009
38F    5.373203134       0.000277                   DYNAMIC RECONFIG: validate_phase: Current
Phase: 0003 Requested Phase: 0005 Flags: 0009
38F    5.373203430       0.000296                   DYNAMIC RECONFIG: get_user_data: DR
Operation: 0000000000000002 input data: 000000002FF22988
38F    5.373203774       0.000344                   DYNAMIC RECONFIG: dr_callout: DR Callout
index: 0001 DR Phase: 0004
38F    5.373204199       0.000425                  DYNAMIC RECONFIG: Kernel_notify: Perform DR
Kernel Phase
38F    5.373204359       0.000160                   DYNAMIC RECONFIG: dr_callout: DR Callout
index: 0001 DR Phase: 0005
38F    5.373205799       0.001440                  DYNAMIC RECONFIG: Invoke Reconfig Handlers:
action: 0200
38F    5.373206199       0.000400                  DYNAMIC RECONFIG: Invoke Reconfig Handlers:
Number of reconfig handlers waiting for: 0000
38F    5.373206666       0.000467                  DYNAMIC RECONFIG: Invoke Reconfig Handlers:
All reconfig handlers completed, Status: 0000
```

```
38F    5.373212290       0.005624                   DYNAMIC RECONFIG: Register_dr_event: DR
Operation: 80000002
38F    5.432293377      59.081087                   DYNAMIC RECONFIG: Unregister_dr_event: DR
Operation: 80000002
38F    5.432298927       0.005550                  DYNAMIC RECONFIG: Invoke Reconfig Handlers:
action: 0400
38F    5.432309307       0.010380                  DYNAMIC RECONFIG: Invoke Reconfig Handlers:
Number of reconfig handlers waiting for: 0000
38F    5.432309707       0.000400                  DYNAMIC RECONFIG: Invoke Reconfig Handlers:
All reconfig handlers completed, Status: 0000
38F    5.533246164     100.936457                  DYNAMIC RECONFIG: Dr_notify: DR Phase: 0006
Flags: 0001 Timeout in secs: 0000 Input: 0000000000000000
38F    5.533247161       0.000997                   DYNAMIC RECONFIG: Validate_notify: DR
Phase: 0006 Flags: 0001
38F    5.533248219       0.001058                   DYNAMIC RECONFIG: validate_phase: Current
Phase: 0005 Requested Phase: 0006 Flags: 0001
38F    5.533250176       0.001957                   DYNAMIC RECONFIG: Run_notify: Perform DR
Check/Pre/Post/Posterror Phases
38F    5.533250348       0.000172                   DYNAMIC RECONFIG: dr_callout: DR Callout
index: 0001 DR Phase: 0006
38F    5.533429817       0.179469                  DYNAMIC RECONFIG: Dr_notify: DR Phase: 0006
Flags: 0002 Timeout in secs: 0000 Input: 0000000000000000
38F    5.533430291       0.000474                   DYNAMIC RECONFIG: Validate_notify: DR
Phase: 0006 Flags: 0002
38F    5.533430518       0.000227                   DYNAMIC RECONFIG: validate_phase: Current
Phase: 0006 Requested Phase: 0006 Flags: 0002
38F    5.533430789       0.000271                   DYNAMIC RECONFIG: Run_notify: Perform DR
Check/Pre/Post/Posterror Phases
38F    5.533431349       0.000560                   DYNAMIC RECONFIG: dr_send_signal: Posting
signal (003A) to all processes catching
38F    5.533512937       0.081588                  DYNAMIC RECONFIG: dr_send_signal: Number of
processes posted: 0000
38F    5.535515956       2.003019                   DYNAMIC RECONFIG: Dr_unregister:
Unregistering DR operation
38F    5.535517217       0.001261                   DYNAMIC RECONFIG: dr_callout: DR Callout
index: 0001 DR Phase: 0009
38F    5.535517777       0.000560                   DYNAMIC RECONFIG: Clearing DR Kernel
Data...
38F    5.535520023       0.002246                   DYNAMIC RECONFIG: Unregister_dr_event: DR
Operation: 40000000
38F    5.548827288      13.307265                   DYNAMIC RECONFIG: HA_proc: Checking with
Kernel for BAD CPU: Input: 0001 Event: 0000000000000001 Retry: 0000000000000000
002    7.719713425    2170.886137                   TRACE OFF channel 0000 Sat Jul 27 17:21:21
2002
```

Memory removal trace output

Example C-8 shows a sample system trace output when a memory removal DLPAR operation is successfully performed.

Example: C-8 Sample system trace output for a memory removal request

```
ID     ELAPSED_SEC     DELTA_MSEC   APPL    SYSCALL KERNEL  INTERRUPT

001    0.000000000       0.000000                   TRACE ON channel 0
                                                    Sat Jul 27 17:20:16 2002
38F    7.821123474    7821.123474                   DYNAMIC RECONFIG: Dr_register: DR
Operation: 0000000000000001 FORCE Option: 0000
38F    7.821125437       0.001963                   DYNAMIC RECONFIG: get_user_data: DR
Operation: 0000000000000001 input data: 000000002FF22970
38F    7.821127517       0.002080                   DYNAMIC RECONFIG: Register_dr_event: DR
Operation: 40000000
38F    7.821128637       0.001120                   DYNAMIC RECONFIG: dr_callout: DR Callout
index: 0000 DR Phase: 0001
38F    7.822487468       1.358831                  DYNAMIC RECONFIG: Dr_notify: DR Phase: 0002
Flags: 0001 Timeout in secs: 0000 Input: 0000000000000000
38F    7.822488219       0.000751                   DYNAMIC RECONFIG: Validate_notify: DR
Phase: 0002 Flags: 0001
38F    7.822488601       0.000382                   DYNAMIC RECONFIG: validate_phase: Current
Phase: 0001 Requested Phase: 0002 Flags: 0001
38F    7.822489610       0.001009                   DYNAMIC RECONFIG: Run_notify: Perform DR
Check/Pre/Post/Posterror Phases
38F    7.822489813       0.000203                   DYNAMIC RECONFIG: dr_callout: DR Callout
index: 0000 DR Phase: 0002
38F    7.822603894       0.114081                  DYNAMIC RECONFIG: Dr_notify: DR Phase: 0002
Flags: 0004 Timeout in secs: 003C Input: 0000000000000000
38F    7.822604356       0.000462                   DYNAMIC RECONFIG: Validate_notify: DR
Phase: 0002 Flags: 0004
38F    7.822604534       0.000178                   DYNAMIC RECONFIG: validate_phase: Current
Phase: 0002 Requested Phase: 0002 Flags: 0004
38F    7.822608060       0.003526                   DYNAMIC RECONFIG: Run_notify: Perform DR
Check/Pre/Post/Posterror Phases
38F    7.822608417       0.000357                  DYNAMIC RECONFIG: Invoke Reconfig Handlers:
action: 1000
38F    7.822610244       0.001827                  DYNAMIC RECONFIG: Invoke Reconfig Handlers:
Number of reconfig handlers waiting for: 0000
38F    7.822610736       0.000492                  DYNAMIC RECONFIG: Invoke Reconfig Handlers:
All reconfig handlers completed, Status: 0000
38F    7.822686190       0.075454                  DYNAMIC RECONFIG: Dr_notify: DR Phase: 0002
Flags: 0002 Timeout in secs: 0000 Input: 0000000000000000
38F    7.822687070       0.000880                   DYNAMIC RECONFIG: Validate_notify: DR
Phase: 0002 Flags: 0002
38F    7.822687316       0.000246                   DYNAMIC RECONFIG: validate_phase: Current
Phase: 0002 Requested Phase: 0002 Flags: 0002
```

```
38F    7.822687629       0.000313                   DYNAMIC RECONFIG: Run_notify: Perform DR
Check/Pre/Post/Posterror Phases
38F    7.822687863       0.000234                   DYNAMIC RECONFIG: dr_send_signal: Posting
signal (003A) to all processes catching
38F    7.822772761       0.084898                   DYNAMIC RECONFIG: dr_send_signal: Number of
processes posted: 0000
38F    7.824002622       1.229861                   DYNAMIC RECONFIG: Dr_notify: DR Phase: 0003
Flags: 0001 Timeout in secs: 0000 Input: 0000000000000000
38F    7.824003040       0.000418                   DYNAMIC RECONFIG: Validate_notify: DR
Phase: 0003 Flags: 0001
38F    7.824003268       0.000228                   DYNAMIC RECONFIG: validate_phase: Current
Phase: 0002 Requested Phase: 0003 Flags: 0001
38F    7.824003612       0.000344                   DYNAMIC RECONFIG: Run_notify: Perform DR
Check/Pre/Post/Posterror Phases
38F    7.824003858       0.000246                   DYNAMIC RECONFIG: dr_callout: DR Callout
index: 0000 DR Phase: 0003
38F    7.824117669       0.113811                   DYNAMIC RECONFIG: Dr_notify: DR Phase: 0003
Flags: 0002 Timeout in secs: 0000 Input: 0000000000000000
38F    7.824118088       0.000419                   DYNAMIC RECONFIG: Validate_notify: DR
Phase: 0003 Flags: 0002
38F    7.824118328       0.000240                   DYNAMIC RECONFIG: validate_phase: Current
Phase: 0003 Requested Phase: 0003 Flags: 0002
38F    7.824118592       0.000264                   DYNAMIC RECONFIG: Run_notify: Perform DR
Check/Pre/Post/Posterror Phases
38F    7.824118832       0.000240                   DYNAMIC RECONFIG: dr_send_signal: Posting
signal (003A) to all processes catching
38F    7.824154322       0.035490                   DYNAMIC RECONFIG: dr_send_signal: Number of
processes posted: 0000
38F    7.825608291       1.453969                   DYNAMIC RECONFIG: Dr_notify: DR Phase: 0005
Flags: 0009 Timeout in secs: 0000 Input: 000000002FF22970
38F    7.825608752       0.000461                   DYNAMIC RECONFIG: Validate_notify: DR
Phase: 0005 Flags: 0009
38F    7.825608961       0.000209                   DYNAMIC RECONFIG: validate_phase: Current
Phase: 0003 Requested Phase: 0005 Flags: 0009
38F    7.825609287       0.000326                   DYNAMIC RECONFIG: get_user_data: DR
Operation: 0000000000000001 input data: 000000002FF22970
38F    7.825609632       0.000345                   DYNAMIC RECONFIG: dr_callout: DR Callout
index: 0000 DR Phase: 0004
38F    7.825610013       0.000381                   DYNAMIC RECONFIG: Kernel_notify: Perform DR
Kernel Phase
38F    7.825610167       0.000154                   DYNAMIC RECONFIG: dr_callout: DR Callout
index: 0000 DR Phase: 0005
38F    7.825611625       0.001458                   DYNAMIC RECONFIG: Invoke Reconfig Handlers:
action: 2000
38F    7.825617182       0.005557                   DYNAMIC RECONFIG: Invoke Reconfig Handlers:
Number of reconfig handlers waiting for: 0000
38F    7.825617643       0.000461                   DYNAMIC RECONFIG: Invoke Reconfig Handlers:
All reconfig handlers completed, Status: 0000
```

```
38F    7.825622836       0.005193                   DYNAMIC RECONFIG: Register_dr_event: DR
Operation: 80000001
38F    7.909427355      83.804519                   DYNAMIC RECONFIG: Unregister_dr_event: DR
Operation: 80000001
38F    7.909764006       0.336651                  DYNAMIC RECONFIG: Invoke Reconfig Handlers:
action: 4000
38F    7.909765157       0.001151                  DYNAMIC RECONFIG: Invoke Reconfig Handlers:
Number of reconfig handlers waiting for: 0000
38F    7.909765551       0.000394                  DYNAMIC RECONFIG: Invoke Reconfig Handlers:
All reconfig handlers completed, Status: 0000
38F    9.184001504    1274.235953                  DYNAMIC RECONFIG: Dr_notify: DR Phase: 0006
Flags: 0001 Timeout in secs: 0000 Input: 0000000000000000
38F    9.184002919       0.001415                   DYNAMIC RECONFIG: Validate_notify: DR
Phase: 0006 Flags: 0001
38F    9.184003412       0.000493                   DYNAMIC RECONFIG: validate_phase: Current
Phase: 0005 Requested Phase: 0006 Flags: 0001
38F    9.184007042       0.003630                   DYNAMIC RECONFIG: Run_notify: Perform DR
Check/Pre/Post/Posterror Phases
38F    9.184007214       0.000172                   DYNAMIC RECONFIG: dr_callout: DR Callout
index: 0000 DR Phase: 0006
38F    9.184189925       0.182711                  DYNAMIC RECONFIG: Dr_notify: DR Phase: 0006
Flags: 0002 Timeout in secs: 0000 Input: 0000000000000000
38F    9.184190528       0.000603                   DYNAMIC RECONFIG: Validate_notify: DR
Phase: 0006 Flags: 0002
38F    9.184190756       0.000228                   DYNAMIC RECONFIG: validate_phase: Current
Phase: 0006 Requested Phase: 0006 Flags: 0002
38F    9.184191051       0.000295                   DYNAMIC RECONFIG: Run_notify: Perform DR
Check/Pre/Post/Posterror Phases
38F    9.184192196       0.001145                   DYNAMIC RECONFIG: dr_send_signal: Posting
signal (003A) to all processes catching
38F    9.184340075       0.147879                  DYNAMIC RECONFIG: dr_send_signal: Number of
processes posted: 0000
38F    9.186468805       2.128730                   DYNAMIC RECONFIG: Dr_unregister:
Unregistering DR operation
38F    9.186471334       0.002529                   DYNAMIC RECONFIG: dr_callout: DR Callout
index: 0000 DR Phase: 0009
38F    9.186472743       0.001409                   DYNAMIC RECONFIG: Clearing DR Kernel
Data...
38F    9.186475444       0.002701                   DYNAMIC RECONFIG: Unregister_dr_event: DR
Operation: 40000000
38F    9.201400247      14.924803                   DYNAMIC RECONFIG: HA_proc: Checking with
Kernel for BAD CPU: Input: 0001 Event: 0000000000000001 Retry: 0000000000000000
002   11.867866488    2666.466241                   TRACE OFF channel 0000 Sat Jul 27 17:20:28
2002
```

Using the AIX error log facility

AIX generates an error log entry when a DLPAR operation fails due to a kernel, kernel extension, or other platform failures. The following three examples provide sample error log entries:

- Example C-9
- Example C-10 on page 468
- Example C-11 on page 469

See Table 3-16 on page 122 for further detailed information about these error log entries.

Example: C-9 Sample AIX error log entry: DR_MEM_UNSAFE_USE

```
LABEL:  DR_MEM_UNSAFE_USE
IDENTIFIER: 12337A8D

Date/Time:       Fri May 24 07:47:39 CDT
Sequence Number: 637
Machine Id:      003579124C00
Node Id:         thimblelp4
Class:           S
Type:            TEMP
Resource Name:   DR_KER_MEM

Description
Affected memory not available for DR removal

Probable Causes
Kernel extension not DR aware

Failure Causes
Memory marked as non removable

    Recommended Actions
    Contact kernel extension owner

Detail Data
Return Code
         114
Memory Address
0000 0000 6927 2000
LR Value
0000 0000 0010 30DC
Module Name
/usr/lib/drivers/testmod
```

Example: C-10   Sample AIX error log entry: DR_DMA_MIGRATE_FAIL

```
LABEL:  DR_DMA_MIGRATE_FAIL
IDENTIFIER: 4DA8FE60

Date/Time:       Fri May 24 04:10:29 CDT
Sequence Number: 622
Machine Id:      003579124C00
Node Id:         thimblelp4
Class:           S
Type:            TEMP
Resource Name:   DR_KER_MEM

Description
Memory related DR operation failed

Probable Causes
DMA activity to memory being removed

Failure Causes
DMA specific memory migration failed

    Recommended Actions
    Quiesce the device causing DMA to the memory

Detail Data
Return Code
         0           2
Memory Address
0000 0003 FF11 1000
Hypervisor return code
           -2
LIOBN
0000 0008
DMA Address
0000 0000 0000 0000 0080 C000
```

Example: C-11   Sample AIX error log entry: DR_DMA_MAPPAER_FAIL

```
LABEL:  DR_DMA_MAPPER_FAIL
IDENTIFIER: 268DA6A3

Date/Time:       Fri May 24 04:10:29 CDT
Sequence Number: 621
Machine Id:      003579124C00
Node Id:         thimblelp4
Class:           S
Type:            TEMP
Resource Name:   DR_KER_MEM

Description
Memory related DR operation failed

Probable Causes
DMA Mapper DR handler failure

Failure Causes
DMA specific memory mapper failed

    Recommended Actions
    Try DR operation on other memory resources

Detail Data
Return Code
           4         -16
Memory Address
0000 0000 4096 A000
Handler Address
0000 0000 01F2 A1A4
Module Name
/usr/lib/drivers/pci_busdd
```

D

Using the Job Scheduling Console

This appendix shows how to define jobs and job streams using the Job Scheduling Console (JSC) with IBM Tivoli Workload Scheduler.

Using the Job Scheduling Console

Open the Job Scheduling Console on your Windows-based workstation as shown in Figure D-1.

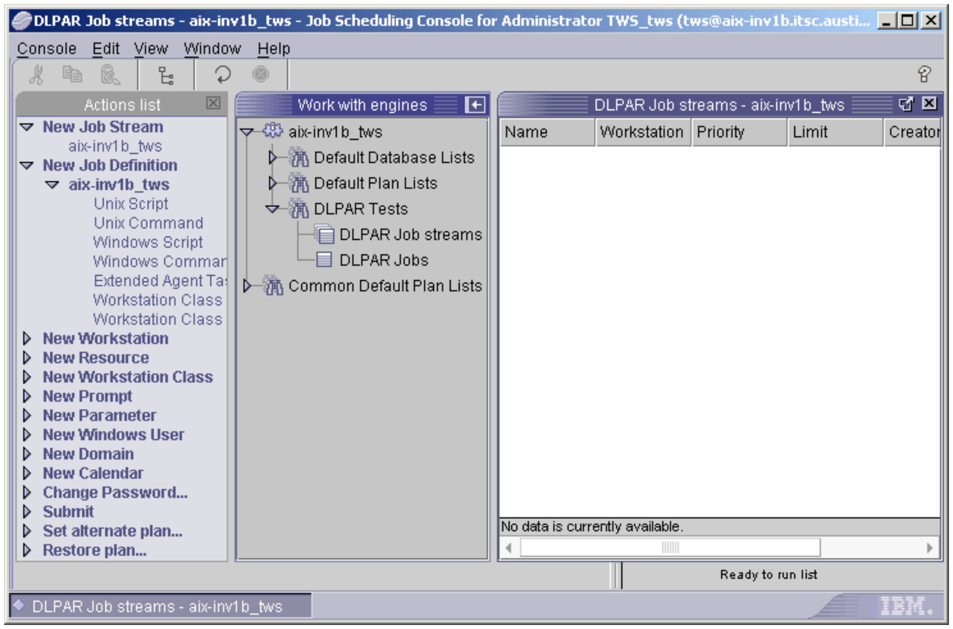
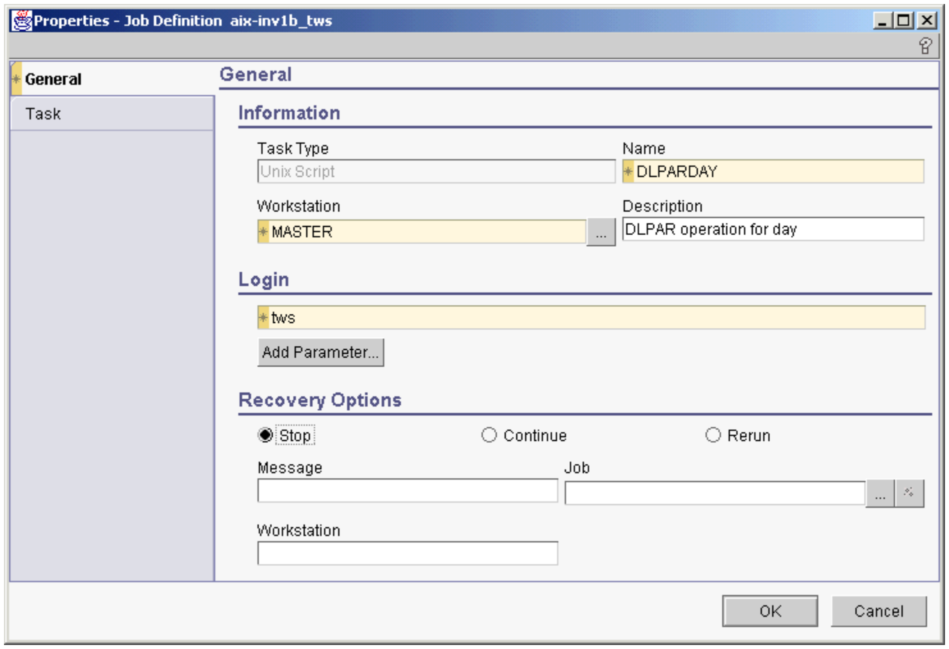


Figure D-1 Job Scheduling Console

Create a new job

To create a job in the Job Scheduling Console, do the following:

1. In the Actions list pane, open the **New Job Definition**.
2. Select the scheduler engine you want to use.
3. Select the **Unix Script** job type. The Properties - Job Definition window is displayed as shown in Figure D-2 on page 473.


Properties - Job Definition aix-inv1b_tws

General

Task

General

Information

Task Type: Unix Script

Name: DLPARDAY

Workstation: MASTER

Description: DLPAR operation for day

Login

tws

Add Parameter...

Recovery Options

[✓ Stop] Continue Rerun

Message

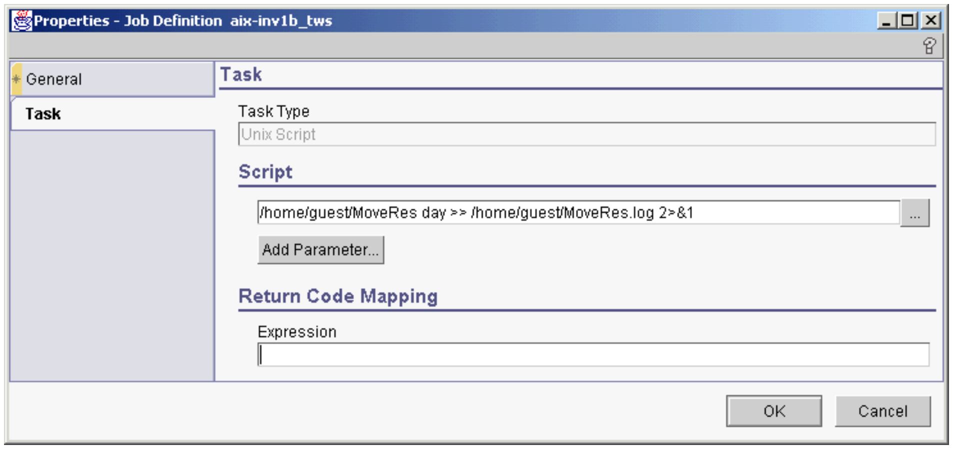
Job

Workstation

OK Cancel

Figure D-2 Properties - Job Definition: General

4. Complete at least the next fields:
   - Name
   - Workstation
   - Login
5. Click **Task** in the left pane as shown in Figure D-3 on page 474.



Task Type: Unix Script

Script: /home/guest/MoveRes day >> /home/guest/MoveRes.log 2>&1

Return Code Mapping

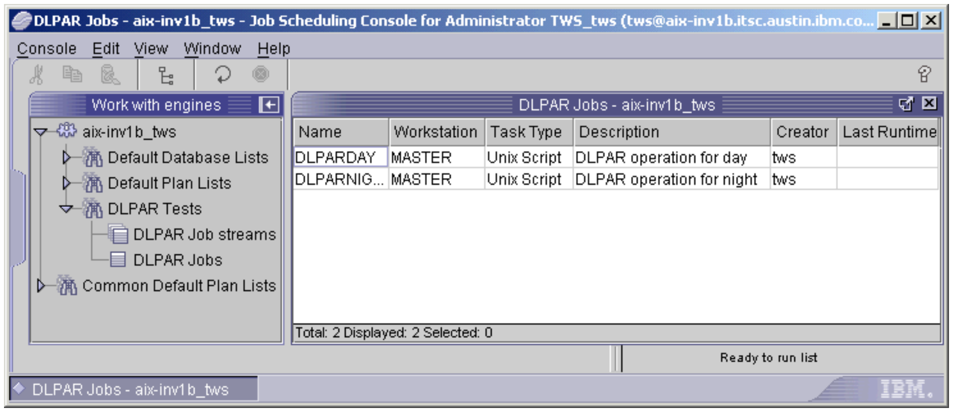
Expression:

Figure D-3 Properties - Job Definition: Task

6. Enter the script name in the Script field.
7. Click **OK** to add the new job.

Repeat the previous steps for the second job (DLPARNIGHT).

Figure D-4 shows the two jobs we just created, DLPARDAY and DLPARNIGHT.


DLPAR Jobs - aix-inv1b_tws - Job Scheduling Console for Administrator TWS_tws (tws@aix-inv1b.itsc.austin.ibm.co...
Console Edit View Window Help
Work with engines
aix-inv1b_tws
Default Database Lists
Default Plan Lists
DLPAR Tests
DLPAR Job streams
DLPAR Jobs
Common Default Plan Lists
DLPAR Jobs - aix-inv1b_tws
Ready to run list
DLPAR Jobs - aix-inv1b_tws

Name	Workstation	Task Type	Description	Creator	Last Runtime
DLPARDAY	MASTER	Unix Script	DLPAR operation for day	tws	
DLPARNIG...	MASTER	Unix Script	DLPAR operation for night	tws	

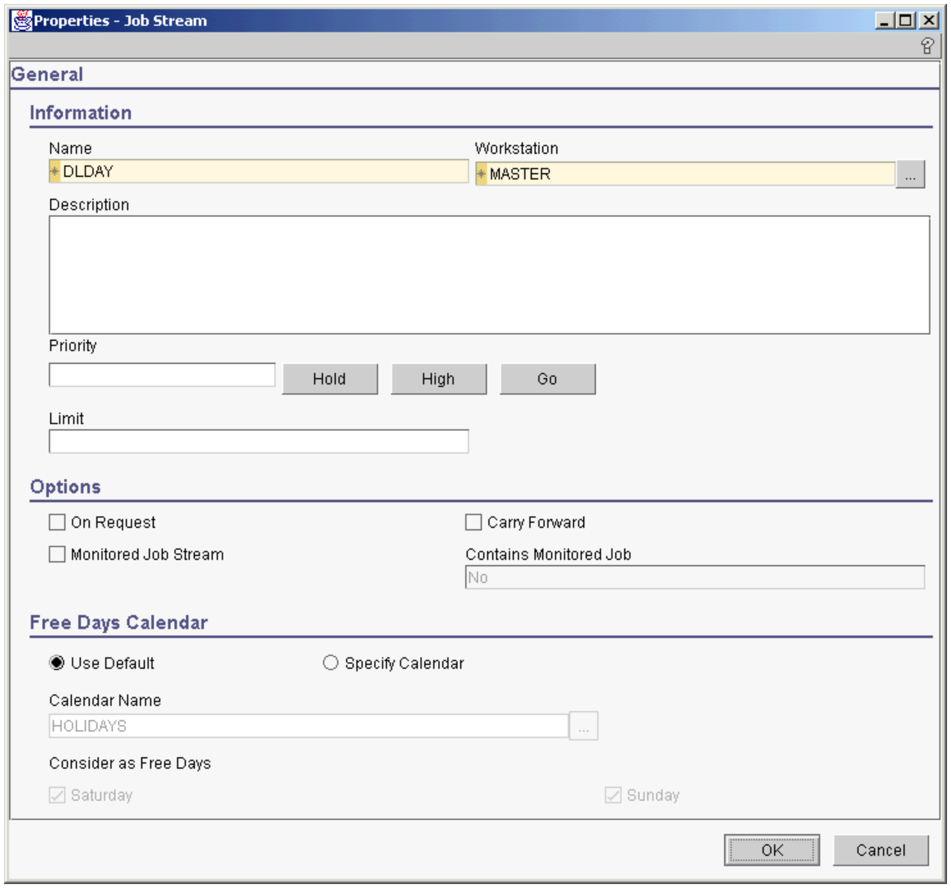
Total: 2 Displayed: 2 Selected: 0

Figure D-4 Jobs List

Create a new job stream

This process shows you how to add a job stream in the Job Scheduling Console. To create a job stream, do the following:

1. In the Actions list pane, open the **New Job Stream**.
2. Select the scheduler engine you want to use. The Properties - Job Stream window is displayed, as shown in Figure D-5.


Properties - Job Stream

General

Information

Name: DLDAY

Workstation: MASTER

Description:

Priority:

Limit:

Options

[ ] On Request
[ ] Carry Forward
[ ] Monitored Job Stream

Contains Monitored Job: No

Free Days Calendar

[● Use Default] ○ Specify Calendar

Calendar Name: HOLIDAYS

Consider as Free Days

[✓ Saturday] [✓ Sunday]

Figure D-5 Properties - Job Stream

3. Complete at least the next fields:
   – Name
   – Workstation
4. Click **OK**. The Job Stream Editor window is displayed, as shown in Figure D-6 on page 476.

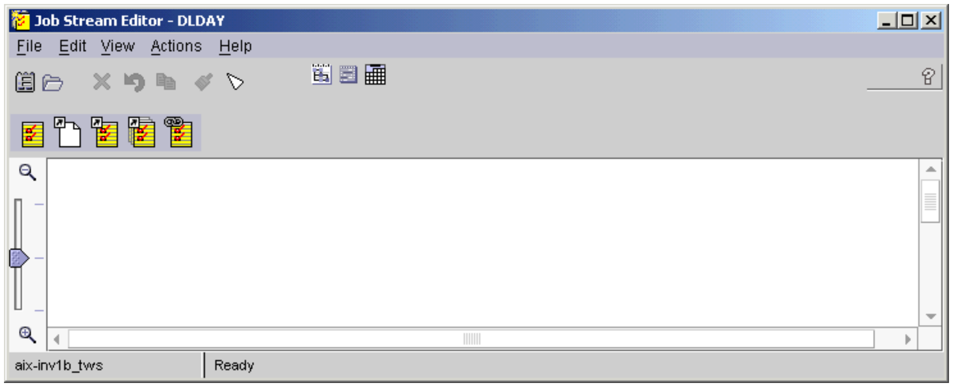


Figure D-6 Job Stream Editor

5. From the menu, select **Actions** → **Add Job** → **Job Definition**. The Properties - Job window is displayed, as shown in Figure D-7.

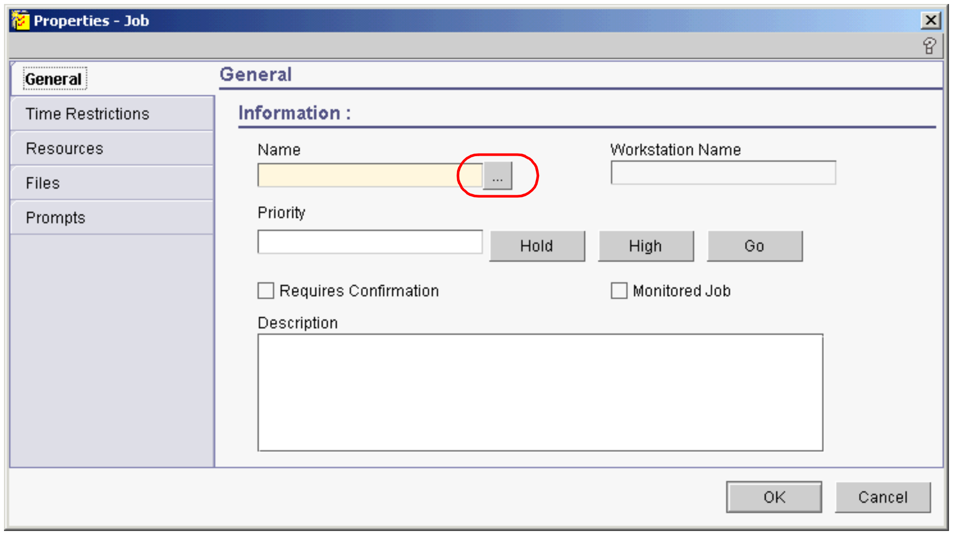


Figure D-7 Properties - Job

6. Click the **...** button. The Find Job window is displayed, as shown in Figure D-8 on page 477.

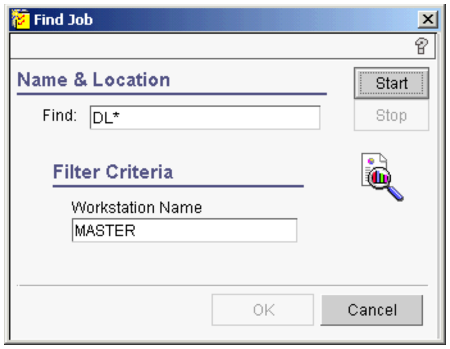
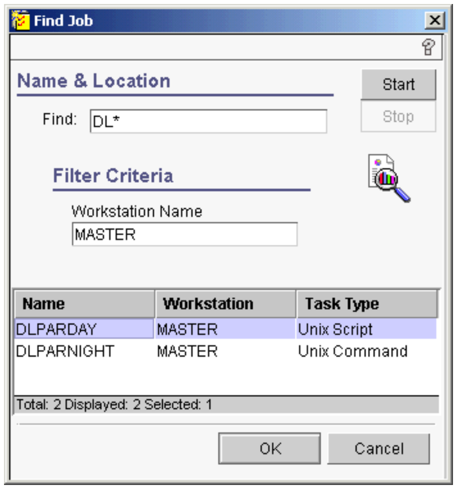


Figure D-8 Find Job

7. Complete the Find and Workstation Name fields.
8. Click **Start**. A list of jobs matching the criteria is displayed, as shown in Figure D-9.

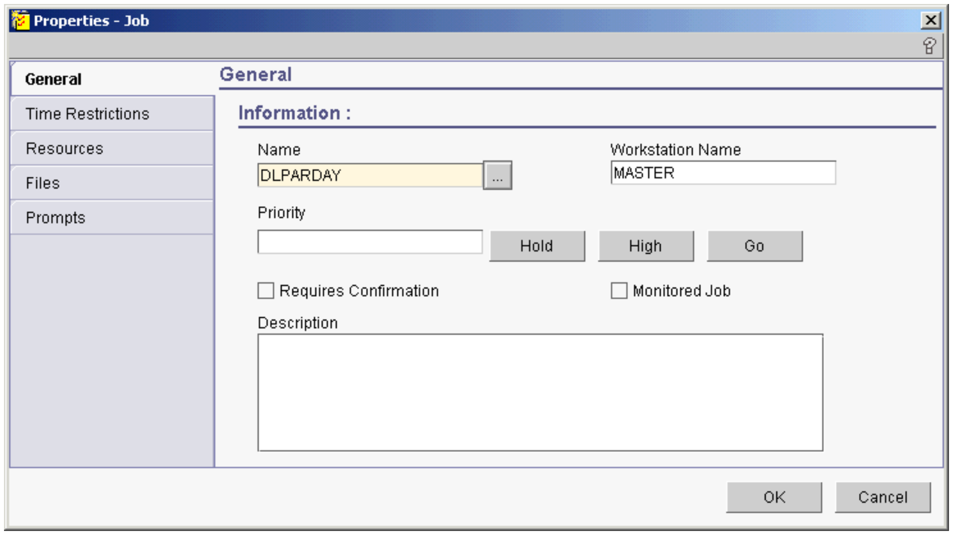


Name	Workstation	Task Type
DLPARDAY	MASTER	Unix Script
DLPARNIGHT	MASTER	Unix Command

Total: 2 Displayed: 2 Selected: 1

Figure D-9 Find Job with job list

9. Select the **DLPARDAY** job.
10. Click **OK**. The Properties - Job window is updated with the selected job, as shown in Figure D-10 on page 478.



Name: DLPARDAY

Workstation Name: MASTER

Priority:

☐ Requires Confirmation

☐ Monitored Job

Description:

Figure D-10 Properties - Job: General

11. Click **OK** to add the job to the job stream editor window, as shown in Figure D-11.

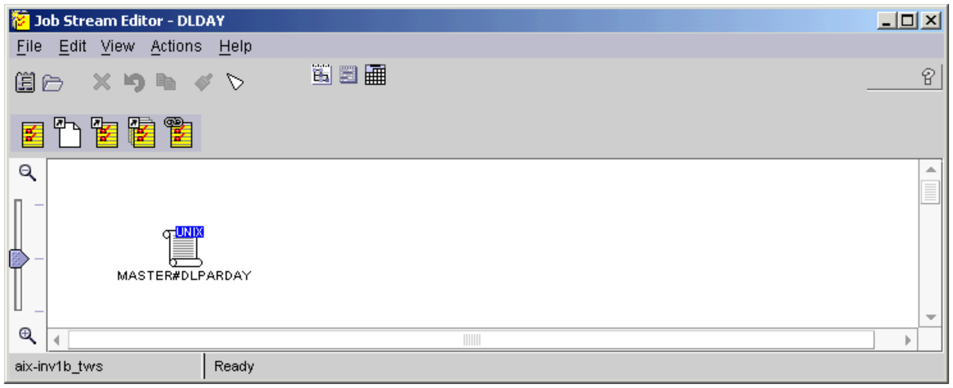
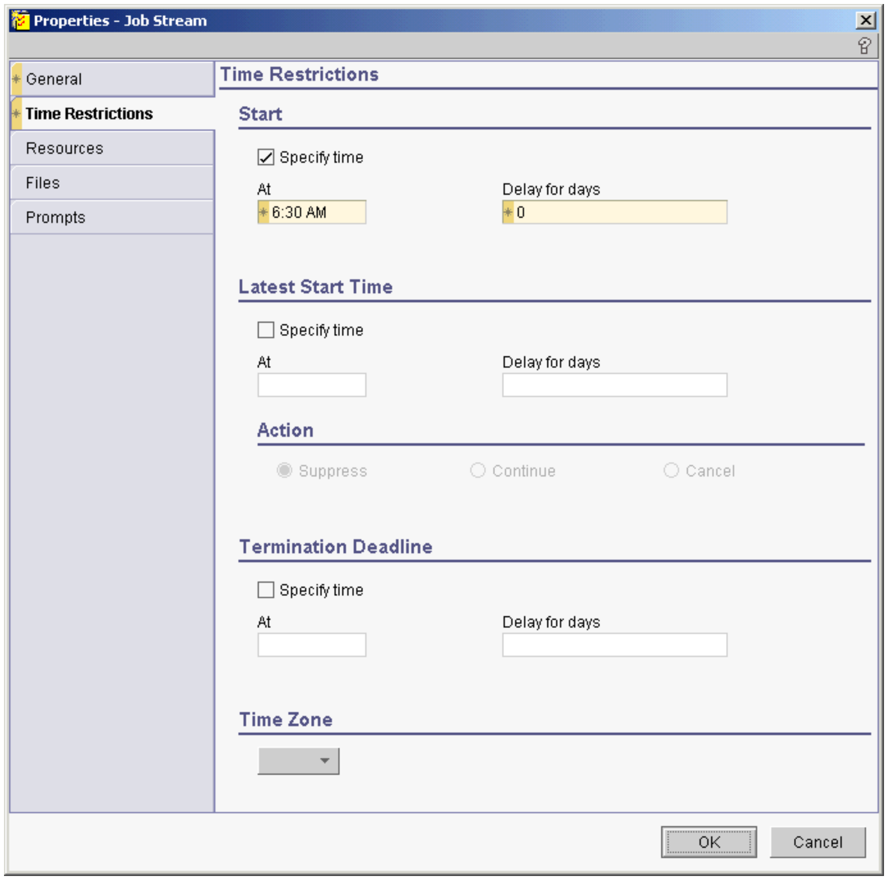


Figure D-11 Job Stream Editor with a job

12. From the menu, select **File** → **Properties**. The Properties - Job Stream window is displayed.

13. Click **Time Restrictions** in the left pane, as shown in Figure D-12 on page 479.


Properties - Job Stream

Time Restrictions

Start

[✓] Specify time

At: 6:30 AM

Delay for days: 0

Latest Start Time

[ ] Specify time

At:

Delay for days:

Action

(●) Suppress ( ) Continue ( ) Cancel

Termination Deadline

[ ] Specify time

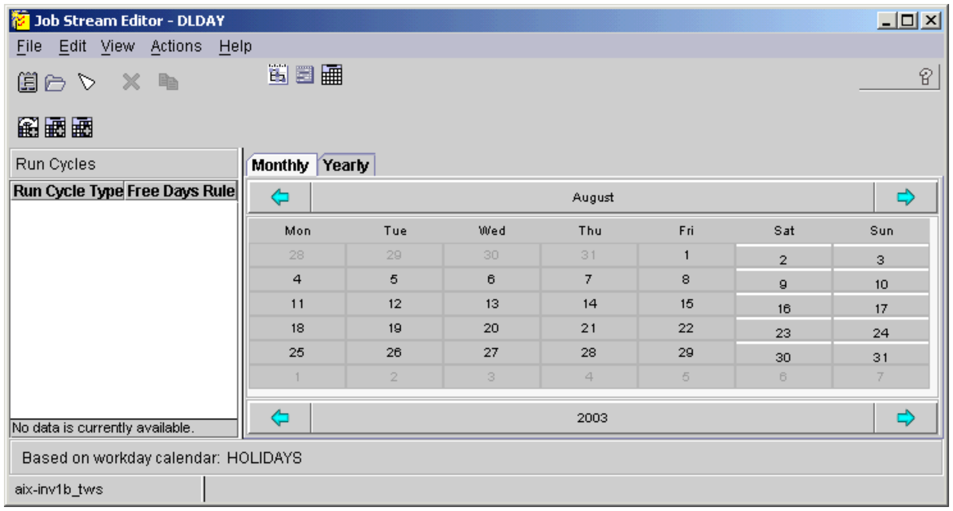
At:

Delay for days:

Time Zone

Figure D-12 Properties - Job Stream: Time Restriction

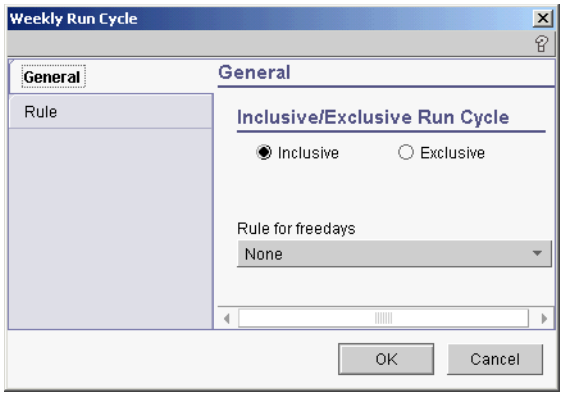
14. Select **Specify time**.
15. Enter the beginning time of the job stream in the field At.
16. Click **OK**.
17. From the menu, select **View** → **Run Cycle**. The run cycle view is displayed as shown in Figure D-13 on page 480.


Job Stream Editor - DLDAY
File Edit View Actions Help
Run Cycles
Run Cycle Type
Free Days Rule
No data is currently available.
Monthly
Yearly
August
2003
Based on workday calendar: HOLIDAYS
aix-inv1b_tws

Mon	Tue	Wed	Thu	Fri	Sat	Sun
28	29	30	31	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31
1	2	3	4	5	6	7

Figure D-13 Job Stream Editor: Run Cycle

18. From the menu, select **Actions** → **Add** → **Weekly Run Cycle**. The Weekly Run Cycle window is displayed, as shown in Figure D-14.

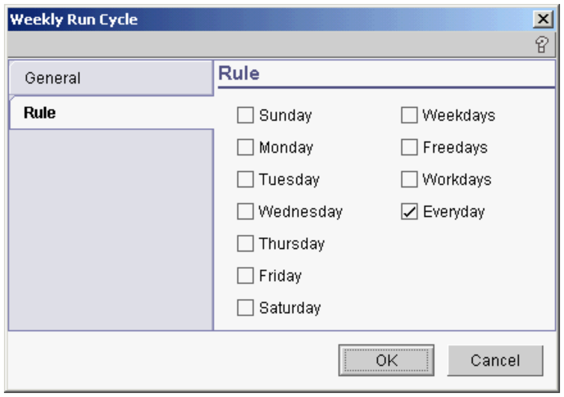


Inclusive/Exclusive Run Cycle: [✓ Inclusive] Exclusive

Rule for freedays: None

Figure D-14 Weekly Run Cycle

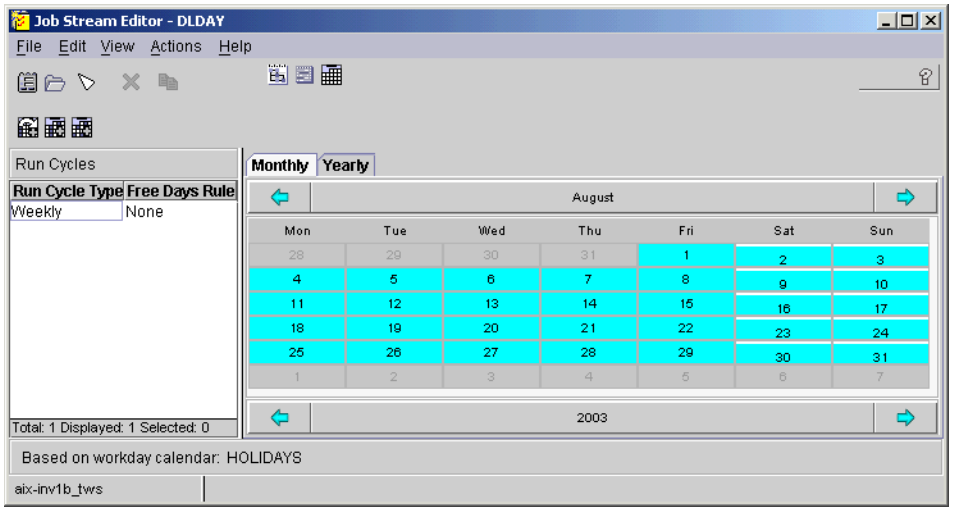
19. Click **Rule** in the left pane. The rule view is displayed, as shown in Figure D-15 on page 481.



[✓ Everyday]

Figure D-15 Weekly Run Cycle: Rule

20. Select **Everyday**.
21. Click **OK** to update the job run cycle, as shown in Figure D-16.



Run Cycle Type	Free Days Rule
Weekly	None

Total: 1 Displayed: 1 Selected: 0

Mon	Tue	Wed	Thu	Fri	Sat	Sun
28	29	30	31	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31
1	2	3	4	5	6	7

Based on workday calendar: HOLIDAYS

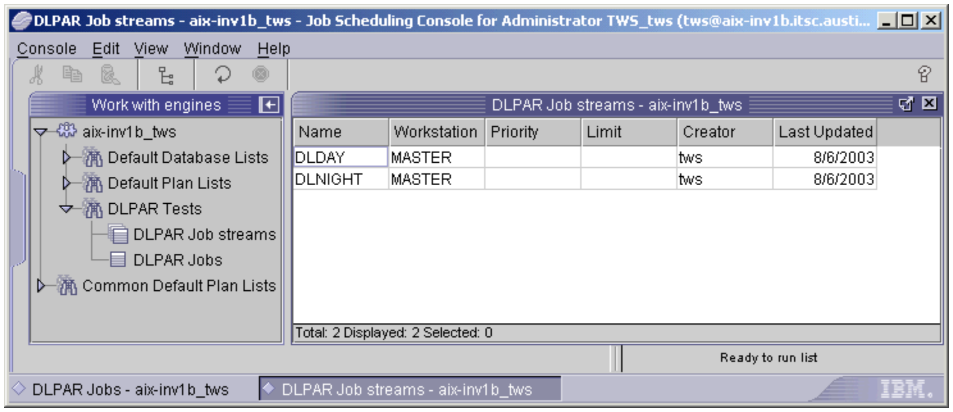
aix-inv1b_tws

Figure D-16 Job Stream Editor: Run Cycle

22. From the menu, select **File** → **Save**.
23. From the menu, select **File** → **Close**.

Repeat the previous steps for the second job stream (DLNIGHT).

Figure D-17 on page 482 shows the two job streams we just created, DLDAY and DLNIGHT.


DLPAR Job streams - aix-inv1b_tws - Job Scheduling Console for Administrator TWS_tws (tws@aix-inv1b.itsc.austi...
Console Edit View Window Help
Work with engines
aix-inv1b_tws
Default Database Lists
Default Plan Lists
DLPAR Tests
DLPAR Job streams
DLPAR Jobs
Common Default Plan Lists
DLPAR Job streams - aix-inv1b_tws
Ready to run list
DLPAR Jobs - aix-inv1b_tws
DLPAR Job streams - aix-inv1b_tws

Name	Workstation	Priority	Limit	Creator	Last Updated
DLDAY	MASTER			tws	8/6/2003
DLNIGHT	MASTER			tws	8/6/2003

Total: 2 Displayed: 2 Selected: 0

Figure D-17 Job Streams List

E

Advanced DLPAR script examples

The examples in this appendix show how the configuration file of the IHS is modified with Perl scripts. There is one script for changing the number of child processes the IHS spawns at startup (StartServers), and another that changes the maximum number of connections to the IHS (MaxClients).

E.1 Changing StartServers

Example E-1 shows the Perl code used for modifying the configuration file of the IHS. The script opens the configuration file and searches for the StartServers stanza. The number behind that stanza is changed to the number given on the command line.

To change the number of httpd daemon processes to 40, use the following command:

```
chStartServers.pl 40
```

With this code we write a new configuration file including the modification to a temporary file. Then the temporary file is copied over the configuration file. By using this method, the configuration file is changed in one single operation and there is no risk of corrupting the configuration file.

Example: E-1 Perl code for changing StartServers in httpd.conf

```
#!/usr/bin/perl -w
use strict;

# set stanza to change
my $stanza = "StartServers";

# check if command line argument is given,
# we probably would like to check if this is a number
if ($#ARGV < 0) {
    # print usage information and exit if no arguments are found
    print("usage: ch$stanza number\n");
    exit 1;
}
# get number to set from command line
my $n = $ARGV[0];

# set file names
my $filename = "/usr/HTTPServer/conf/httpd.conf";
my $tmpfile  = "/usr/HTTPServer/conf/httpd.conf.tmp";

# open config file read mode
open(FILE, "$filename") or die "can't open $filename";
# open temporary file in write mode
open(TMP,  ">$tmpfile")  or die "can't open $tmpfile";

# read file
while (<FILE>) {
    my $line = $_;
    # find stanza with regular expression
    if ($line =~ m/^\s*$stanza\s+\d+\s*$/ ) {
        # change value
        $line =~ s/$stanza\s+\d+/$stanza $n/;
    }
    # print line to temporary file
    print TMP $line;
}

close(FILE);
close(TMP);

# cp temporary file to config file
rename($tmpfile, $filename);

exit;
__END__
```

Example E-2 shows the Perl code for printing the current setting for StartServers in the httpd.conf file. We open the configuration file and search the StartServers stanza with a regular expression. The number behind the StartServers keyword is printed to standard output.

Example: E-2 Perl script for reading the value of StartServers

```
#!/usr/bin/perl -w
use strict;

# set stanza to read
my $stanza = "StartServers";

# set file name
my $filename = "/usr/HTTPServer/conf/httpd.conf";

# open file in read mode
open(FILE, "$filename") or die "can't open $filename";

# value in stanza
my $value;

# read file into array
my @array = <FILE>;
# close file
close(FILE);

# find stanza with regular expression
foreach (@array) {
    if ($_ =~ m/^\s*$stanza\s+(\d+)\s*$/ ) {
        # read value
        $value = $1;
    }
}

printf("$value\n");

exit;
__END__
```

E.2 Changing MaxClients

The Perl code in Example E-3 is very similar to Example E-1 on page 484. We have changed the name of the configuration stanza that has to be changed in the configuration file. Again the script is called with the new value as a command line argument.

If we want to change the maximum number of connections to the IHS to 600, we use the following command:

```
chMaxClients.pl 600
```

Example: E-3 Perl code for changing MaxClients in httpd.conf

```
#!/usr/bin/perl -w
use strict;

# set stanza to change
my $stanza = "MaxClients";

# check if command line argument is given,
# we probably would like to check if this is a number
if ($#ARGV < 0) {
    # print usage information and exit if no arguments are found
    print("usage: ch$stanza number\n");
    exit 1;
}
# get number to set from command line
my $n = $ARGV[0];

# set file names
my $filename = "/usr/HTTPServer/conf/httpd.conf";
my $tmpfile  = "/usr/HTTPServer/conf/httpd.conf.tmp";

# open config file read mode
open(FILE, "$filename") or die "can't open $filename";
# open temporary file in write mode
open(TMP,  ">$tmpfile")  or die "can't open $tmpfile";
```

```
# read file
while (<FILE>) {
    my $line = $_;
    # find stanza with regular expression
    if ($line =~ m/^\s*$stanza\s+\d+\s*$/ ) {
        # change value
        $line =~ s/$stanza\s+\d+/$stanza $n/;
    }
    # print line to temporary file
    print TMP $line;
}

close(FILE);
close(TMP);

# cp temporary file to config file
rename($tmpfile, $filename);

exit;
__END__
```

Example E-4 shows the Perl code for reading the current value of MaxClients from the httpd.conf file and printing the value to standard output.

Example: E-4 Perl script for reading the value of MaxClients

```
#!/usr/bin/perl -w
use strict;

# set stanza to read
my $stanza = "MaxClients";

# set file name
my $filename = "/usr/HTTPServer/conf/httpd.conf";

# open file in read mode
open(FILE, "$filename") or die "can't open $filename";

# value in stanza
my $value;

# read file into array
my @array = <FILE>;
# close file
close(FILE);

# find stanza with regular expression
foreach (@array) {
    if ($_ =~ m/^\s*$stanza\s+(\d+)\s*$/ ) {
        # read value
        $value = $1;
    }
}

printf("$value\n");

exit;
__END__
```

F

Autonomic application example

This appendix contains an autonomic C program example discussed in Chapter 12, “Autonomic applications” on page 375.

F.1 Autonomic C program example

Example F-1 shows an autonomic C program example discussed in Chapter 12, “Autonomic applications” on page 375.

To compile the program, do the following:

```
$ cc _r -q64 seek_IBM.c
```

To run the compiled executable program, a.out, do the following:

```
$ ./seek_IBM | tee seek_IBM.log
```

If you need to terminate the program before it ends, use Ctrl-C on the terminal where you have invoked the program, or send the SIGINT signal to the program process. This instructs the program to release acquired shared memory segments before it terminates.

Note: The program is designed as a 64-bit multi-threaded application.

Example: F-1 seek_IBM.c

```
#include <pthread.h>
#include <stdio.h>
#include <unistd.h>
#include <stddef.h>
#include <stdlib.h>
#include <string.h>
#include <stdarg.h>
#include <sys/shm.h>
#include <sys/dr.h>
#include <errno.h>

#define ONE_SEG (256 * 1024 * 1024)

pthread_mutex_t prt_log_l;
pthread_mutex_t st_l;
int cpu_units;
int mem_units;
int IBM_found = -1;
enum runflag {
    SEEK_IBM_RUN,
    SEEK_IBM_STOP
} runflag = SEEK_IBM_RUN;

enum st {
    ST_READY,
    ST_RUNNING,
    ST_STOPPING,
    ST_DONE
} *st = NULL;
int stmax = 8;

int mem_for_os = 1024;
int mem_per_task = 256;
int task_per_cpu = 2;
int mem_allowable_diff = 512;
char *dr_phase[] = { “check”, “pre”, “post”, “error” };
char *dr_op[] = { “add”, “remove” };
char *dr_res[] = { “cpu”, “memory”, “other” };
sigset_t set;

void prt_log(char *fmt, ...)
{
    va_list argp;

    if (runflag == SEEK_IBM_STOP) {
        return;
    }
    pthread_mutex_lock(&prt_log_l);
    va_start(argp, fmt);
    vprintf(fmt, argp);
    fflush(stdout);
    va_end(argp);
    pthread_mutex_unlock(&prt_log_l);
}


int checkst(int task)
{
    pthread_mutex_lock(&st_l);
    if (st[task] == ST_STOPPING) {
        /* Aborted */
        st[task] = ST_READY;
        pthread_mutex_unlock(&st_l);
        prt_log(“%d: stopped.\n”, task);
        return -1;
    }
    pthread_mutex_unlock(&st_l);
    return 0;
}

void *worker(void *arg)
{
    char *buf;
    int id, rc, task = *((int *) arg);
    time_t start, end;
```

```
int unit, c, ibmc = 0;
FILE *fp;
unsigned int seed;

free(arg);
prt_log("%d: started.\n", task);

if ((fp = fopen("/dev/urandom", "r")) == NULL) {
    prt_log("Can't open /dev/urandom.\n");
}
fread(&seed, 1, 4, fp);
fclose(fp);
prt_log("%d: random seed=%x\n", task, seed);

if ((id = shmget(IPC_PRIVATE, ONE_SEG, IPC_CREAT | 0600))
    == 0) {
    prt_log("%d: shmget error.\n", task);
    pthread_exit(NULL);
}
prt_log("%d: id=%d\n", task, id);
if ((buf = shmat(id, 0, 0)) == NULL) {
    prt_log("%d: shmat error.\n", task);
    pthread_exit(NULL);
}
prt_log("%d: buf=%x\n", task, buf);

time(&start);

/* Repeat 16 times; 16 MB x 16 = 256 MB. */
for (unit = 0; unit < 16; unit++) {
    int count;
    int *unitbuf = (int *) (buf + unit * 16 * 1024 * 1024);

    /* Fill up 16 MB of memory with random data. */
    time(&start);
    for (count = 0; count < 4 * 1024 * 1024; count++) {
        char *nowbuf = (char *) unitbuf + count;
        int n;

        /* Set random number in integer on 4 byte of charactor */
        unitbuf[count] = rand_r(&seed);
        unitbuf[count] *= unitbuf[count];

        /* Convert each bytes to printable characters */
        for (n = 0; n < 4; n++) {
            nowbuf[n] = nowbuf[n] % 26 + 'A';
        }
    }
    time(&end);
```

```
            prt_log("task:%d has completed initializing of unit:"
                    "%d in %d sec.\n", task, unit, end - start);

            /* Check if this task is requested to stop. */
            if (checkst(task)) {
                goto end_of_task;
            }
        }

        /* Repeat 16 times; 16MBx16=256MB */
        for (unit = 0; unit < 16; unit++) {
            char *ubuf = buf + unit * 16 * 1024 * 1024;
            int max = unit == 15 ? 16 * 1024 * 1024 - 3 : 16 * 1024 * 1024;

            /* Scan 16 MB of memory */
            time(&start);
            for (c = 0; c < max; c++) {
                if (ubuf[c] == 'I' && ubuf[c + 1] == 'B' && ubuf[c + 2] == 'M') {
                    ibmc++;
                }
            }
            time(&end);

            prt_log("task:%d has completed scanning of unit:%d in %d sec"
                    " and found \"IBM\" %d times.\n", task, unit, end - start,
                    ibmc);

            /* Check if this task is requested to stop. */
            if (checkst(task)) {
                goto end_of_task;
            }
        }
        pthread_mutex_lock(&st_l);
        /* It's done. */
        st[task] = ST_DONE;
        if (IBM_found < 0) {
            IBM_found = ibmc;
        } else {
            IBM_found += ibmc;
        }
        pthread_mutex_unlock(&st_l);
        prt_log("task %d completed. \"IBM\" found %d times.\n", task, ibmc);

      end_of_task:
        if (rc = shmdt(buf)) {
            prt_log("%d: shmdt error.\n", task);
            pthread_exit(NULL);
        }
```

```
    if (rc = shmctl(id, IPC_RMID, NULL)) {
        prt_log("%d: shmctl error(%d).\n", task, rc);
        pthread_exit(NULL);
    }
    pthread_exit(NULL);
}

void task_scheduler(int n)
{
    pthread_t ch;
    int c, d = 0;
    int *no;

    pthread_mutex_lock(&st_l);
    for (c = 0; c < stmax; c++) {
        if (st[c] == ST_RUNNING) {
            d++;
            continue;
        }
        if (st[c] == ST_READY && d < n && runflag == SEEK_IBM_RUN) {
            no = malloc(sizeof(int));
            *no = c;
            pthread_create(&ch, NULL, worker, (void *) no);
            pthread_detach(ch);
            prt_log("create thread %d %d\n", *no, d);
            st[c] = ST_RUNNING;
            d++;
        } else if (st[c] == ST_RUNNING && d >= n) {
            st[c] = ST_STOPPING;
        }
    }
    pthread_mutex_unlock(&st_l);
}

int taskleft()
{
    int c, t = 0;

    pthread_mutex_lock(&st_l);
    if (runflag == SEEK_IBM_RUN) {
        /* Check if it is not completed yet. */
        for (c = 0; c < stmax; c++) {
            if (st[c] == ST_READY) {
                t++;
            }
        }
    } else {
        /* Check if it is not stopped yet, while runflag=SEEK_IBM_STOP. */
        for (c = 0; c < stmax; c++) {
            if (st[c] == ST_STOPPING) {
                t++;
            }
        }
    }
    pthread_mutex_unlock(&st_l);
    return t;
}

int threadnum()
{
    int c, t = 0;

    pthread_mutex_lock(&st_l);
    for (c = 0; c < stmax; c++) {
        if (st[c] == ST_RUNNING) {
            t++;
        }
    }
    pthread_mutex_unlock(&st_l);
    return t;
}


int reconfig_check(int cpu, int mem)
{
    int mem_for_task = mem_per_task * task_per_cpu * cpu;

    prt_log("** You will have %d cpu and %d MB of memory.\n"
            "** As a cpu can run %d task and a task requires %d MB of\n"
            "** memory, and you specified to save %d MB for OS,\n"
            "** you need %d MB of memory in total for %d task.\n",
            cpu, mem, task_per_cpu, mem_per_task, mem_for_os,
            mem_for_task + mem_for_os, cpu * task_per_cpu);

    if (mem < mem_for_os) {
        prt_log
            ("** Hey! You will have only %d MB of memory, that is not less\n"
             "** than your setting in mem_for_os=%d\n.\n", mem,
             mem_for_os);
        return -1;
    } else if (mem_for_task + mem_for_os - mem_allowable_diff > mem) {
        prt_log
            ("** Hey! You will have %d MB of memory that is much less than\n"
             "** required memory size, %d MB.  Allowable difference between\n"
             "** these two values is less than %d MB.\n", mem,
             mem_for_task + mem_for_os, mem_allowable_diff);
        return -1;
    } else if (mem_for_task + mem_for_os + mem_allowable_diff < mem) {
        prt_log
            ("** Hey! You will have %d MB of memory that is much more than\n"
             "** required memory size, %d MB.  Allowable difference between\n"
             "** these two values is less than %d MB.\n", mem,
             mem_for_task + mem_for_os, mem_allowable_diff);
        return -1;
    } else {
        int memslots = (mem - mem_for_os) / mem_per_task;
        int cpuslots = task_per_cpu * cpu;

        if (cpuslots < memslots) {
            prt_log("** Memory size (%d MB) that will be required for task"
                    " and OS is less than \n"
                    "** amount of memory (%d MB) you will have.  Let's run"
                    " %d tasks.\n",
                    mem_for_task + mem_for_os, mem, cpuslots);
            return cpuslots;
        } else {
            prt_log
                ("** Memory size (%d MB) that will be required for task and"
                 " OS is more than \n"
                 "** equal amount of memory (%d MB) you will have.  You can"
                 " not run %d tasks.\n"
                 "** You want to save %d MB for OS, then you can use %d MB"
                 " for task out of\n"
                 "** %d MB total memory size.  As each task requires %d MB,"
                 " you can run\n"
                 "** %d task.\n", mem_for_task + mem_for_os, mem, cpuslots,
                 mem_for_os, mem - mem_for_os, mem, mem_per_task,
                 memslots);
            return memslots;
        }
    }
}


void reconfig_manager()
{
    dr_info_t dr;
    int rc, now, n, c;
    char *phase, *op, *res;

    prt_log("- A reconfig request is received.\n");
    if ((rc = dr_reconfig(DR_QUERY, &dr)) == 0) {
        if (dr.check) {
            phase = dr_phase[0];
        } else if (dr.pre) {
            phase = dr_phase[1];
        } else if (dr.post) {
```

```
        phase = dr_phase[2];
    } else
        phase = dr_phase[3];

    if (dr.add) {
        op = dr_op[0];
    } else {
        op = dr_op[1];
    }

    if (dr.cpu) {
        res = dr_res[0];
    } else if (dr.mem) {
        res = dr_res[1];
    } else {
        res = dr_res[2];
    }

    prt_log("- %s phase to %s %s\n", phase, op, res);

    if (dr.mem) {
        n = (int) (dr.req_memsz_change / (1024 * 1024));
        if (dr.add) {
            if (dr.pre || dr.check) {
                prt_log("- add memory %d\n", n);
            } else if (dr.post &&
                       (c =
                        reconfig_check(cpu_units,
                                       mem_units + n)) > 0) {
                prt_log("- add memory %d + %d\n", mem_units, n);
                task_scheduler(c);
                mem_units += n;
            } else {
                dr_reconfig(DR_EVENT_FAIL, &dr);
                prt_log("- Reconfig request is denied as "
                        "%dMB of memory is too large for %d CPUs.\n",
                        mem_units + n, cpu_units);
                return;
            }
        } else if (dr.rem) {
            if (dr.post || dr.check) {
                prt_log("- remove memory %d\n", n);
            } else if (dr.pre &&
                       (c =
                        reconfig_check(cpu_units,
                                       mem_units - n)) > 0) {
                prt_log("- remove memory %d - %d\n", mem_units, n);
                task_scheduler(c);
                mem_units -= n;
            } else {
                dr_reconfig(DR_EVENT_FAIL, &dr);
                prt_log("- Reconfig request is denied as "
                        "%dMB of memory is too little for %d CPUs.\n",
                        mem_units - n, cpu_units);
                return;
            }
        }
    } else if (dr.cpu) {
        n = 1;
        if (dr.add) {
            if (dr.pre || dr.check) {
                prt_log("- add CPU %d\n", n);
            } else if (dr.post &&
                       (c =
                        reconfig_check(cpu_units + 1,
                                       mem_units)) > 0) {
                prt_log("- add CPU %d + %d\n", cpu_units, n);
                task_scheduler(c);
                cpu_units += n;
            } else {
                dr_reconfig(DR_EVENT_FAIL, &dr);
                prt_log("- Reconfig request is denied as "
                        "%d CPUs are too many for %d MB of memory.\n",
                        cpu_units + n, mem_units);
                return;
            }
        } else if (dr.rem) {
            if (dr.post || dr.check) {
                prt_log("- remove CPU %d\n", n);
            } else if (dr.pre &&
                       (c =
                        reconfig_check(cpu_units - 1,
                                       mem_units)) > 0) {
                prt_log("- remove CPU %d - %d\n", cpu_units, n);
                task_scheduler(c);
                cpu_units -= n;
            } else {
                dr_reconfig(DR_EVENT_FAIL, &dr);
                prt_log("- Reconfig request is denied as "
                        "%d CPUs are too few for %d MB of memory.\n",
                        cpu_units - n, mem_units);
                return;
            }
        }
    }
    dr_reconfig(DR_RECONFIG_DONE, &dr);
    prt_log("- Reconfig request is accepted.\n");
} else {
        prt_log("dr_reconfig() error.\n");
    }
}

void *sigwatcher(void *arg)
{
    int rc;
    sigset_t newset;

    sigemptyset(&newset);
    sigaddset(&newset, SIGRECONFIG);
    sigaddset(&newset, SIGINT);
    while (!sigwait(&newset, &rc)) {
        fprintf(stderr, "sigwatcher: rc=%d\n", rc);
        if (rc == SIGINT) {
            pthread_mutex_lock(&st_l);
            runflag = SEEK_IBM_STOP;
            pthread_mutex_unlock(&st_l);
            task_scheduler(0);
            break;
        } else if (rc == SIGRECONFIG) {
            reconfig_manager();
        }
    }
    pthread_exit(NULL);
}

int setup(char *name)
{
    FILE *fp;
    char buf[BUFSIZ];
    char *p;
    int c;

    if ((fp = fopen(name, "r")) == NULL) {
        printf("Can't open file %s\n", name);
        exit(1);
    }
    while (fgets(buf, sizeof(buf), fp)) {
        if (strncmp(buf, "mem_for_os=", 11) == 0) {
            mem_for_os = atoi(buf + 11);
        } else if (strncmp(buf, "mem_per_task=", 13) == 0) {
            mem_per_task = atoi(buf + 13);
        } else if (strncmp(buf, "task_per_cpu=", 13) == 0) {
            task_per_cpu = atoi(buf + 13);
        } else if (strncmp(buf, "mem_allowable_diff=", 19) == 0) {
            mem_allowable_diff = atoi(buf + 19);
        } else if (strncmp(buf, "max_task=", 9) == 0) {
            stmax = atoi(buf + 9);
        }
    }
    fclose(fp);

    if (stmax <= 0
        || (st = (enum st *) malloc(sizeof(int) * stmax)) == NULL) {
        printf("Can't allocate task table.\n");
        exit(1);
    }
    for (c = 0; c < stmax; c++) {
        st[c] = ST_READY;
    }

    cpu_units = sysconf(_SC_NPROCESSORS_ONLN);

    if ((fp = popen("lsattr -E0l mem0", "r")) == NULL ||
        fgets(buf, 20, fp) == NULL ||
        fgets(buf, 20, fp) == NULL || (p = strchr(buf, ':')) == NULL) {
        printf("Can't get memory size by lsattr -E0l mem0.\n");
    }
    *p = 0;
    mem_units = atoi(buf);
    if ((c = reconfig_check(cpu_units, mem_units)) < 0) {
        printf("CPU and memory are out of balance. "
               "(mem_units=%d MB, cpu_units=%d).\n", mem_units, cpu_units);
        exit(3);
    }
    return c;
}


void main(int argc, char **argv)
{
    pthread_t ch;
    int c, n;

    sigemptyset(&set);
    sigaddset(&set, SIGINT);
    sigaddset(&set, SIGRECONFIG);
    pthread_sigmask(SIG_BLOCK, &set, NULL);

    if (argc > 1) {
        c = setup(argv[1]);
    } else {
        c = setup("seek_IBM.conf");
    }

    pthread_mutex_init(&prt_log_l, NULL);
    pthread_mutex_init(&st_l, NULL);
    pthread_create(&ch, NULL, sigwatcher, NULL);

    task_scheduler(c);

    while (1) {
        while (1) {
            if (!(c = threadnum())) {
                break;
            }
            if (IBM_found < 0) {
                prt_log
                    (“+ CPU:%d, memory:%d [MB], threads: %d, tasks: %d\n”,
                     cpu_units, mem_units, c, n);
            } else {
                prt_log(“+ CPU:%d, memory:%d [MB], threads: %d, tasks: %d,”
                        “ \”IBM\” found %d times so far.\n”, cpu_units,
                        mem_units, c, n, IBM_found);
            }
            sleep(5);
        }
        if ((n = taskleft()) <= 0) {
            break;
        }
        task_scheduler(reconfig_check(cpu_units, mem_units));
    }
    if (IBM_found >= 0) {
        printf(“*** The program found \”IBM\” %d times in total. ***\n”,
               IBM_found);
    }
    exit(0);
}
```

G

Additional material

This redbook refers to additional material that can be downloaded from the Internet as described below.

Locating the Web material

The Web material associated with this redbook is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

`ftp://www.redbooks.ibm.com/redbooks/SG247039`

Alternatively, you can go to the IBM Redbooks Web site at:

`ibm.com``/redbooks`

Select the **Additional materials** and open the directory that corresponds with the redbook form number, SG247039.

Using the Web material

The additional Web material that accompanies this redbook includes a tar archive file, sg247039-01.tar, that includes the following files:

```
# tar tvf sg247039-01.tar
drwxr-sr-x   0 0       0 Sep 22 13:50:00 2003 ./
drwxr-sr-x   0 0       0 Sep 22 13:40:51 2003 ./ch03/
-rw-r--r--   0 0   28757 Nov 15 09:56:12 2002 ./ch03/DLPAR_appl.c
-rw-r--r--   0 0   16885 Nov 15 12:45:27 2002 ./ch03/dr_IBM_template.pl
-rw-r--r--   0 0   14109 Nov 12 16:31:19 2002 ./ch03/dr_IBM_template.sh
drwxr-sr-x   0 0       0 Sep 22 13:49:02 2003 ./ch09/
-rwxr--r--   0 0    2850 Sep 22 13:45:18 2003 ./ch09/MoveCD
-rwxr--r--   0 0    1359 Sep 22 13:45:30 2003 ./ch09/MoveRes
-rwxr-----   0 0    4346 Aug 14 14:28:15 2003 ./ch09/ChangeResConfig
-rw-r--r--   0 0      46 Aug 12 17:12:43 2003 ./ch09/test.cfg
-rw-r--r--   0 0      23 Aug 11 15:07:01 2003 ./ch09/app1.cfg
-rw-r--r--   0 0      22 Aug 13 16:10:50 2003 ./ch09/app1_stby.cfg
-rw-r--r--   0 0      23 Aug 13 16:35:58 2003 ./ch09/app1app2NodeA.cfg
-rw-r--r--   0 0      23 Aug 14 11:24:39 2003 ./ch09/app1app2NodeB.cfg
-rw-r--r--   0 0      23 Aug 14 11:24:14 2003 ./ch09/app2.cfg
-rw-r--r--   0 0      22 Aug 14 11:24:22 2003 ./ch09/app2_stby.cfg
-rwxr--r--   0 0     638 Aug 14 14:54:26 2003 ./ch09/start_app1
-rwxr--r--   0 0     638 Aug 14 14:54:53 2003 ./ch09/start_app2
-rwxr--r--   0 0     633 Aug 14 14:53:56 2003 ./ch09/stop_app1
-rwxr--r--   0 0     631 Aug 14 14:55:19 2003 ./ch09/stop_app2
drwxr-sr-x   0 0       0 Aug 14 10:15:16 2003 ./ch10/
-rwxr-xr-x   0 0   13598 Aug 13 17:53:35 2003 ./ch10/IBM_template.sh
-rw-r--r--   0 0     216 Aug 13 18:04:01 2003 ./ch10/README
-rwxr-xr-x   0 0     517 Aug 13 18:18:51 2003 ./ch10/lsMaxClients.pl
-rwxr-xr-x   0 0    1049 Aug 14 09:37:32 2003 ./ch10/chStartServers-1.pl
-rwxr-xr-x   0 0     979 Aug 14 09:43:23 2003 ./ch10/chStartServers-2.pl
-rwxr-xr-x   0 0    1105 Aug 14 09:40:51 2003 ./ch10/chStartServers-3.pl
-rw-r--r--   0 0    3432 Aug 14 10:15:38 2003 ./ch10/dr_http_reconfig.diff
-rwxr-xr-x   0 0     211 Aug 14 09:48:07 2003 ./ch10/sync.sh
-rw-r--r--   0 0   14665 Aug 14 10:12:22 2003 ./ch10/dr_http_reconfig.sh
-rwxr-xr-x   0 0    1047 Aug 14 09:37:24 2003 ./ch10/chMaxClients-1.pl
-rwxr-xr-x   0 0     977 Aug 14 09:45:28 2003 ./ch10/chMaxClients-2.pl
-rwxr-xr-x   0 0    1103 Aug 14 09:45:02 2003 ./ch10/chMaxClients-3.pl
-rwxr-xr-x   0 0     519 Aug 13 18:18:33 2003 ./ch10/lsStartServers.pl
drwxr-sr-x   0 0       0 Aug 14 11:16:09 2003 ./ch11/
-rw-r--r--   0 0     309 Aug 13 14:16:13 2003 ./ch11/Makefile
-rw-r--r--   0 0   34431 Aug 12 16:35:34 2003 ./ch11/DLPAR_appl_autonomic.c
-rw-r--r--   0 0     731 Aug 13 14:27:23 2003 ./ch11/block_cpu.c
-rw-r--r--   0 0     371 Aug 13 15:45:10 2003 ./ch11/rset_create.c
-rw-r--r--   0 0     221 Aug 13 18:05:15 2003 ./ch11/README
-rw-r--r--   0 0     473 Aug 08 14:02:20 2003 ./ch11/rset_create_cpu.c
-rw-r--r--   0 0     474 Aug 08 13:57:35 2003 ./ch11/rset_create_mem.c
-rw-r--r--   0 0    7781 Aug 13 18:05:07 2003 ./ch11/DLPAR_appl_autonomic.diff
```

```
-rw-r--r--   0 0    28757 Aug 13 18:03:16 2003 ./ch11/DLPAR_appl.c
-rwxr-xr-x   0 0     3295 Aug 08 16:17:28 2003 ./ch11/test.sh
-rw-r--r--   0 0      595 Aug 08 12:00:38 2003 ./ch11/rset_remcpu.c
-rw-r--r--   0 0      594 Aug 08 14:01:53 2003 ./ch11/rset_remmem.c
-rwxr-xr-x   0 0      208 Aug 08 11:23:52 2003 ./ch11/sync.sh
-rw-r--r--   0 0      660 Aug 13 14:26:04 2003 ./ch11/rset_list_cpu.c
-rw-r--r--   0 0      590 Aug 08 13:56:48 2003 ./ch11/rset_addcpu.c
-rw-r--r--   0 0      591 Aug 08 14:02:04 2003 ./ch11/rset_addmem.c
-rw-r--r--   0 0      182 Aug 08 12:08:26 2003 ./ch11/rset_delete.c
drwxr-sr-x   0 0        0 Sep 22 13:51:04 2003 ./ch12/
-rw-r--r--   0 0    16166 Sep 22 13:51:04 2003 ./ch12/seek_IBM.c
```

System requirements for downloading the Web material

The following system configuration is recommended:

Hard disk space:	3 MB minimum
Operating System:	AIX 5L Version 5.2

How to use the Web material

Create a subdirectory on your AIX systems, and un-tar the contents of the Web material file in this folder.

Abbreviations and acronyms

AIX	Advanced Interactive Executive
ALPAR	affinity logical partition
APAR	authorized program analysis report
API	application programming interface
ASCII	American Standard Code for Information Interchange
ATS	Advanced Technical Support
BOS	basic operating system
CD	compact disc
CD-R	compact disc - recordable
CD-ROM	compact disc - read only media
CD/DVD	compact disc/digital versatile disc
CDT	central daylight time
CEC	central electronics complex
CHRP	Common Hardware Reference Platform
CIM	Common Interface Model
CIMOM	Common Interface Model Object Manager
CPU	central processing unit
CST	central standard time
DLPAR	dynamic logical partitioning
DMA	Direct Memory Access
DMTF	Desktop Management Task Force
DNS	Domain Name System
DR	Data Register
DR	dynamic reconfiguration
DRAF	Dynamic Reconfiguration Application Framework
DVD	digital versatile disc
DVD-RAM	digital versatile disc - random access memory
DVD-ROM	digital versatile disc - read only memory
EC	Engineering Change
EEH	Enhanced Error Handling
FC	Feature Code
FQDN	fully qualified domain name
FRU	field replaceable unit
FTSS	Field Technical Support Specialist
GB	gigabyte
GID	group ID
GPFS	General Parallel File System
GUI	graphical user interface
HACMP	high-availability cluster multiprocessing
HMC	Hardware Management Console
HPC	High Performance Computing
HTML	Hypertext Markup Language
I/O	input/output
IBM	International Business Machines Corporation
IPLA	IBM International Program License Agreement
ID	identification
IDE	internal data equipment
IP	Internet Protocol
IPL	initial program load
IR	Instruction Register

ISA	Industry Standard Architecture
ISO	International Organization for Standardization
ITSO	International Technical Support Organization
JFS	Journaled File System
KB	kilobyte
KDB	kernel debugger
LAN	local area network
LDAP	Lightweight Directory Access Protocol
LED	light emitting diode
LMB	logical memory block
LPAR	logical partitioning
LUN	logical unit
LV	logical volume
LVD	Low Voltage Differential
LVM	Logical Volume Manager
M/T	machine type
MB	megabyte
MCM	multichip module
MDL	model
MSR	Machine Status Register
NFS	Network File System
NIM	Network Installation Manager
NIS	Network Information Services
NVRAM	nonvolatile random access memory
ODM	Object Data Manager
OEM	original equipment manufacturer
OS	operating system
PAP	Password Authentication Protocol
PC	personal computer
PCI	peripheral component interconnect
PFT	Page Frame Table
PHB	PCI host bridge
PID	process ID
PMB	physical memory block
POWER	performance optimization with enhanced RISC
PPID	parent process ID
PSSP	Parallel System Support Program
QBB	Quad Building Block
R/W	read/write
RAM	random access memory
RAN	remote asynchronous node
RAS	reliability, availability, and serviceability
RDBMS	relational database management system
RIP	Routing Information Protocol
RISC	reduced instruction-set computer
RMC	Resource Monitoring and Control
RML	Real Mode Limit
RMO	real mode offset
ROM	read only memory
RPM	Red Hat Package Manager
RSCT	Reliable and Scalable Cluster Technology
RSPC	RISC System Personal Computer
RTAS	Run-Time Abstraction Service
S/N	serial number
SCSI	small computer system interface
SMIT	System Management Interface Tool
SMP	symmetric multiprocessing

SMS	System Management Services
SP	scalable parallel
SPOT	shared product object tree
SSA	serial storage architecture
SSH	Secure Shell
SSL	Secure Sockets Layer
TCB	trusted computing base
TCE	translation control entry
TCP	Transmission Control Protocol
TLB	Translation Look-aside Buffer
TOD	time-of-day
TTY	Teletype
UDB	Universal Database
UDF	Universal Disk Format
UID	user ID
UP	uniprocessor
URL	Universal Resource Locator
USB	Universal Serial Bus
VMM	Virtual Memory Manager
VPD	Vital Product Data
VSD	Virtual Shared Disk
WLM	Workload Manager

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 515.

- *A Practical Guide for Resource Monitoring and Control*, SG24-6615
- *IBM @server pSeries 670 and pSeries 690 System Handbook*, SG24-7040
- *Linux Applications on pSeries*, SG24-6033
- *Managing AIX Server Farms*, SG24-6606
- *POWER4 Processor Introduction and Tuning Guide*, SG24-7041
- *The Complete Partitioning Guide for IBM @server pSeries Servers*, SG24-7039

IBM Redpapers

IBM Redpapers are available in softcopy only.

- *IBM @server pSeries 615 Models 6C3 and 6E3 Technical Overview and Introduction, REDP0160*
- *IBM @server pSeries 630 Models 6C4 and 6E4 Technical Overview and Introduction*, REDP0195
- *IBM @server pSeries 650 Model 6M2 Technical Overview and Introduction*, REDP0194

pSeries hardware publications

The following publications are shipped with the IBM @server pSeries servers. These publications are also available at the following URL (click the corresponding model name):

`http://www.ibm.com/servers/eserver/pseries/library/hardware_docs/`

- *128-Port Asynchronous PCI Adapter Installation and User's Guide*, SA23-2563
- *8-Port Asynchronous PCI Adapter Installation and User's Guide*, SA23-2562
- *Adapter, Devices, and Cable Information for Multiple Bus Systems*, SA38-0516
- *D10 I/O Drawer Installation Guide*, SA23-1296
- *D20 I/O Drawer Installation Guide*, SA23-1295
- *Installation Guide 61D I/O drawer 61R Second I/O Rack*, SA23-1281
- *IBM @server pSeries 615 Model 6C3 and 6E3 Installation Guide*, SA38-0628
- *IBM @server pSeries 615 Model 6C3 and 6E3 Service Guide*, SA38-0629
- *IBM @server pSeries 615 Model 6C3 and 6E3 User's Guide*, SA38-0630
- *IBM @server pSeries 630 Model 6C4 and 6E4 Installation Guide*, SA38-0605
- *IBM @server pSeries 630 Model 6C4 and 6E4 Service Guide*, SA38-0604
- *IBM @server pSeries 630 Model 6C4 and 6E4 User's Guide*, SA38-0606
- *IBM @server pSeries 650 Model 6M2 Installation Guide*, SA38-0610
- *IBM @server pSeries 650 Model 6M2 Service Guide*, SA38-0612
- *IBM @server pSeries 650 Model 6M2 User's Guide*, SA38-0611
- *IBM @server pSeries 655 Installation Guide*, SA38-0616
- *IBM @server pSeries 655 Service Guide*, SA38-0618
- *IBM @server pSeries 655 User's Guide*, SA38-0617
- *IBM @server pSeries 670 Installation Guide*, SA38-0613
- *IBM @server pSeries 670 Service Guide*, SA38-0615
- *IBM @server pSeries 670 User's Guide*, SA38-0614
- *IBM @server pSeries 690 Installation Guide*, SA38-0587
- *IBM @server pSeries 690 Service Guide*, SA38-0589
- *IBM @server pSeries 690 User's Guide*, SA38-0588
- *IBM @server pSeries 7311 Model D10 and Model D20 Service Guide*, SA38-0627
- *IBM Hardware Management Console for pSeries Maintenance Guide*, SA38-0603
- *IBM Hardware Management Console for pSeries Installation and Operations Guide*, SA38-0590

- *PCI Adapter Placement References*, SA38-0538

AIX official publications

The following publications are contained in the AIX 5L for POWER V 5.2 Documentation CD, 5765-E62, that is shipped as a part of the AIX 5L Version 5.2 CD-ROM media set. These publications are also available at the following URL (click “AIX 5.2”):

http://techsupport.services.ibm.com/server/library

- *AIX Installation in a Partitioned Environment*, SC23-4382
- *AIX 5L Version 5.2 Asynchronous Communications Guide*
- *AIX 5L Version 5.2 Installation Guide and Reference*
- *AIX 5L Version 5.2 Reference Documentation: Commands Reference*
- *AIX 5L Version 5.2 Security Guide*
- *AIX 5L Version 5.2 System Management Guide: AIX 5L Version 5.2 Web-based System Manager Administration Guide*
- *AIX 5L Version 5.2 System Management Guide: Communications and Networks*
- *AIX 5L Version 5.2 System Management Guide: Operating System and Devices*
- *AIX 5L Version 5.2 Understanding the Diagnostic Subsystem for AIX*
- *IBM Reliable Scalable Cluster Technology for AIX 5L, Messages*, SA22-7891
- *IBM Reliable Scalable Cluster Technology for AIX 5L, RSCT Guide and Reference*, SA22-7889
- *IBM Reliable Scalable Cluster Technology for AIX 5L, Technical Reference*, SA22-7890
- *IBM Reliable Scalable Cluster Technology for AIX 5L and Linux, Group Services Programming Guide and Reference*, SA22-7888

CSM for AIX official publications

The following publications are contained in the Cluster Systems Management for AIX 5L product (Program Number: 5765-F67). These publications are also available at the following URL:

http://www.ibm.com/servers/eserver/pseries/library/clusters/aix.html

- *IBM Cluster Systems Management for AIX 5L, Administration Guide*, SA22-7918
- *IBM Cluster Systems Management for AIX 5L, Hardware Control Guide*, SA22-7920
- *IBM Cluster Systems Management for AIX 5L, Planning and Installation Guide*, SA22-7919

CSM for Linux official publications

The following publications are contained in the Cluster Systems Management for Linux product (Program Number: 5765-E88). These publications are also available at the following URL:

http://www.ibm.com/servers/eserver/clusters/library/linux.html

- *IBM Cluster Systems Management for Linux, Administration Guide*, SA22-7873
- *IBM Cluster Systems Management for Linux, Hardware Control Guide*, SA22-7856
- *IBM Cluster Systems Management for Linux, Planning and Installation Guide*, SA22-7853
- *IBM Reliable Scalable Cluster Technology for Linux, Mesages*, SA22-7894
- *IBM Reliable Scalable Cluster Technology for Linux, RSCT Guide and Reference*, SA22-7892
- *IBM Reliable Scalable Cluster Technology for Linux, Technical Reference*, SA22-7893
- *IBM Reliable Scalable Cluster Technology for AIX 5L and Linux, Group Services Programming Guide and Reference*, SA22-7888

Other publications

These publications are also relevant as further information sources:

- *The PowerPC Architecture, IBM, Morgan Kaufmann Publishers, Inc.*, ISBN 1-55860-316-6 PB

Online resources

These Web sites and URLs are also relevant as further information sources:

- *AIX toolkit for Linux applications*

  http://www.ibm.com/servers/aix/products/aixos/linux/download.html
- *IBM @server pSeries Information Center*

  http://publib16.boulder.ibm.com/pseries/en_US/infocenter/base/index.htm
- *IBM @server pSeries & RS/6000* Microcode Updates

  http://techsupport.services.ibm.com/server/mdownload
- *IBM @server pSeries Support Hardware Management Console*

  https://techsupport.services.ibm.com/server/hmc?fetch=home.html
- *Electronic Service Agent for pSeries and RS/6000 User's Guide*

  ftp://service.software.ibm.com/aix/service_agent_code/AIX/svcUG.pdf
- *Electronic Service Agent for pSeries HMC User's Guide*

  ftp://service.software.ibm.com/aix/service_agent_code/HMC/HMCSAUG.pdf
- *Microcode Discovery Service*

  http://techsupport.services.ibm.com/server/aix.invscoutMDS
- *OpenSSH Web site*

  http://www.openssh.com
- *VPD Capture Service*

  http://techsupport.services.ibm.com/server/aix.invscoutVPD

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Index

IBM

Redbooks

The Complete Partitioning Guide for IBM @server pSeries Servers

(1.0” spine)
0.875”<->1.498”
460 <-> 788 pages

The Complete Partitioning Guide for IBM *@server* pSeries Servers

Redbooks

Detailed information about logical partitioning and implementation

How to configure partitions and manage DLPAR operations

Comprehensive AIX installation and migration tasks

This IBM Redbook provides a broad understanding of the logical partitioning on the IBM *@server* partitioning-capable pSeries servers. This is the most outstanding feature of these servers, because it enables the servers to run multiple operating system instances concurrently on a single system. We focus on the following topics:

- Logical partitioning overview
- Partitioning implementation on pSeries servers
- Dynamic logical partitioning
- Creating and managing partitions
- Installing and migrating AIX in a partitioned environment

This redbook is a single-source handbook for IBM and IBM Business Partner technical specialists who support the partitioning-capable pSeries servers, and for application developers who need to develop or modify DLPAR-aware applications.

SG24-7039-01 ISBN 0738499447