AIX 7.2, PowerVM
UNIX, Virtualization, and Security
An administrator's guide

Sebastian Biedroń

Cover design: Paweł Kowalski

ISBN: 9781980457015

CONTENTS

Preface

When we talk about reliable and stable operating systems, we usually refer to systems from the UNIX family. They can be adapted to almost any need, in addition to being flexible and allowing for the far-reaching automation of administrative activities. One of the leading UNIX systems is AIX, which was created by IBM. This book is about that system.

The strength of AIX lies not only in the design of the operating system itself, but also in the hardware platform that it works on, namely IBM Power Systems servers. These servers are based on POWER processors (currently Power8 and Power9) and they outperform the x86 family. Indeed, many benchmarks confirm the high performance of POWER processor-based servers, for example, www.tpc.org.

This book is for people who want to administer UNIX systems. It describes the AIX system, its design, and its operating principles. It is based on AIX 7.2, but in many places, it also describes the differences between that version and previous ones. This provides readers with up-to-date knowledge about this release of AIX as well as earlier releases. Most of the described features of the system will remain current (for many years) in subsequent releases. The book describes the features of the system and provides guidance for administrators on how to perform specific operations using the system as well as the consequences of such operations.

The book is made up of chapters featuring content that requires the reader to have various levels of knowledge. The majority of chapters require only basic knowledge of UNIX or LINUX and computer networking, which makes them suitable for beginners. However, to understand the chapter on virtualization, the reader should have a general understanding of operating system virtualization. Moreover, the security chapter presents issues that will be most easily understood by people with intermediate or advanced knowledge of AIX.

The AIX operating system includes extensive documentation on its various features. Readers can find the answers to many questions directly within it. Nevertheless, the knowledge provided in this way can appear highly fragmented and difficult to understand for beginners. Therefore, the most appropriate approach is to familiarize yourself with the features of the system in a consolidated, clear, and transparent manner, which is facilitated by this book. Another natural step toward expanding your knowledge is to search for interesting issues in the documentation and then analyze them.

Additional information can be found:

- https://www.redbooks.ibm.com - This site contains IBM Redbooks, which describe IBM products and their features.
- https://www.ibm.com/support/knowledgecenter - This site features documentation concerning IBM products. There is a lot of information to be found in the IBM Knowledge Center, but in most cases, the information is in the form of facts, which require a good understanding of the basics.
- There are also many other sources available, such as blogs, that you can easily find using a search engine. Their key advantage is their way of transferring knowledge and experience "from one administrator to another" which is easy to understand and addresses the real problems encountered during the work.

From the author

I was born in the seventies. Why am I writing about it? Because at a relatively young age I had the opportunity to experience the very rapid development of IT. IT companies had grown at a fantastic pace. Computer hardware and software appeared in the home. Many people who lived through those days with open minds do not face the two basic disadvantages that hamper today's administrator:

1. **Not understanding how the different layers of software work**

   A typical IT administrator today is only prepared to work in a very limited field. He usually has highly specialized skills. He is an operating systems specialist (or even a single system specialist). He is a specialist in, for example, databases, application servers, backup systems, or even hardware or virtualization. Specialization is important, but such an isolated approach leads to many deficits that prevent good cooperation and good communication within an organization. A lack of understanding regarding the needs of another group of specialists leads to the wasting of a lot of time in order to make arrangements between teams. As a result, such arrangements may be inappropriate. Scheduled work often fails, which in turn leads to the teams blaming each other for the situation.

   During the late nineties or at the beginning of the present century, there was generally no division into limited specializations. A typical administrator dealt with every product. This allowed him to develop versatility as well as to understand the specifics of tasks performed within different layers. If this versatility has been maintained, it should today result in the rapid understanding of the needs and problems of colleagues from other highly specialized teams.

2. **Not looking for your own solutions**

   Many times, I have heard the phrases "it cannot be done because there is no such information in the documentation" and "it must work in such a way, since the documentation says so." In many cases this is true and, to ensure the correct operation of our part of the environment, it is worth following the relevant documentation. Often, however, product documentation suggests just one solution from among the many possible options. Sometimes, solutions are not found within the documentation, but when you combine several available functionalities, you get something new.

   Again, during the late nineties or at the beginning of the present century, we often worked on products that had poor documentation. Sometimes they had no documentation at all. Through this work we learned to look for new solutions. We became convinced that just because something is missing from the documentation, it does not mean that it cannot be done. It only means that someone did not write about it. Such an attitude makes us want to search and build new solutions that are often not available "in the package."

   Does this mean that the young administrator does not have these skills? No. It simply means that it is harder for him to develop them. It requires a lot of effort, since the specialized world is not conducive to it. So, dear administrators, IT people, do not be pigeonholed. Have an open mind. Work on understanding the whole solution, not just "your part." Seek out new solutions.

The book you hold is the result of my seventeen years of IT experience. Much of my work was related to UNIX systems, especially AIX, the IBM Power Systems platform, and PowerVM virtualization. I developed the standards for the operation of this environment in one of the largest banks in Poland. During a dozen or so years I have created a standardized, fully monitored environment with high availability, disaster recovery, and security solutions, which was optimized for performance and the cost of the software licenses running there. The size of the environment had reached 2000 virtual AIX systems. In my work, having broad knowledge, which extended far beyond the operating systems, was always important - and for years I have tried to keep it up. Nowadays, based on my experience, I participate in a variety of projects related to infrastructure, automation, and cloud solutions.

If you order IBM Authorized Training and would like to be taught by an experienced and open-minded instructor, please get in touch. I willingly share my knowledge and work with all IBM training providers. If you require more information, you can find me on LinkedIn: https://www.linkedin.com/in/sebastian-biedro%C5%84-8372209/

Basic concepts

In order to understand the content of the book, it is necessary to clarify the meaning of several concepts that frequently appear:

Program - process (thread) - daemon

Simply put, a program is a file, or a set of files designed to perform certain functions.

When you run a program, the operating system loads its code into the memory and then creates the structures needed to process that code. In this way you create a process (all the processes in the system can be displayed using the *ps* command).

For a user, the process is a single entity, but when you consider it from the operating system side, you see that it consists of several independent structures created in the memory. The two main structures are the program code and the data. When you run the same program repeatedly, many independent processes are created. Each of them has a separate structure that stores the program code as well as a separate structure that stores the data. Instead of starting subsequent processes, you can create subsequent threads of a given process. These threads will have separate structures that store data, although they will all use a single structure containing the program code. This allows the system to perform the same tasks using less memory. Multithreading must be implemented when the program is created. Kernel processes use this mechanism as well as certain external services, such as the Network File System (all the processes with threads can be viewed using the *ps -emo THREAD* command).

A running process usually runs in the foreground, and it is associated with a specific session and the terminal from which it was started. This means that when closing a session, this process will also be closed. Service programs, when launched, typically create session-independent processes and run in the background. This process is known as a daemon.

Firmware

Firmware is the internal software of a particular device, which is stored in its memory. It is responsible for the way the device works. It can be updated so as to eliminate software malfunctions or improve its functionality. On the AIX server, the firmware can be associated with the entire machine, and it also contains a hypervisor (for more information on the hypervisor, see the virtualization chapter). Firmware can also be associated with devices such as disks, controllers, network adapters, and so on.

Disk - tape - optical drive (bootable)

The term "bootable" refers to devices that can boot the operating system (e.g., a disk, DVD, or tape).

Boot order

The boot order is the order in which the server examines the devices in search of the one from which it will boot. For example, a boot order might look like this: *rmt0 cd0 hdisk0 hdisk1*. This means that the *rmt0* device (the tape drive) will be checked first. If there is no bootable tape in it, then the *cd0* device (DVD drive) will be checked. If it does not contain a bootable disk, the *hdisk0* (hard drive) device will be checked. If it also does not contain an operating system, then another device from the list will be checked.

Dump

Dump is otherwise referred to as a memory dump. It usually occurs due to the critical failure of the system, which prevents it from further operation. The final move of the system is to save its state on the dedicated device. It saves the memory contents, processes, CPU registers, and so on. All this information is intended to help you determine the cause of the malfunction.

UNIX systems

When writing about UNIX systems or just UNIX, I mean all the systems based on the concepts and working methods of the original UNIX system. Therefore, UNIX refers to AIX, as well as Solaris, HP-UX, and many more.

This book describes many system commands. As with any UNIX system, they can be run with a number of parameters (flags) that determine how they work. When describing particular commands, I do not include complete information about the parameters that you can use. You can use the manual (*man* command) or the *Commands Reference* to find out more about the capabilities of individual commands. You can also find relevant information in the *IBM Knowledge Center*.

History of AIX

The foundations of the AIX system were laid during the late eighties. Like most UNIX systems, it stems from UNIX System V. Since the introduction of the original version, many changes have taken place. Over the years, the system has gained new features, while new tools have been added and the existing ones have been improved. The changes made in each version of the system can be tracked by reviewing the documentation describing the differences. Such documentation is usually published for

new releases. The Redbooks available in this series are:

- IBM AIX Version 7.1 Differences Guide;
- IBM AIX Version 6.1 Differences Guide;
- AIX 5L Differences Guide Version 5.3 Edition;
- AIX 5L Differences Guide Version 5.2 Edition;
- AIX Version 4.3 Differences Guide; and
- AIX Version 4.2 Differences Guide.

The changes made to the different AIX versions are shown in Table 0-1.

Table 0-1: History of AIX.

Year	Operating system version
2015	AIX 7.2
2010	AIX 7.1
2007	AIX 6.1
2004	AIX 5L v 5.3
2002	AIX 5L v 5.2
2001	AIX 5L v 5.1
2000	AIX 5L alpha
1999	AIX 4.3.3
1998	AIX 4.3.2
1997	AIX 4.3
1994	AIX 4.2.5 AIX 4.1
1993	AIX 4.0 AIX 3.2.4
1990	AIX 3.2 AIX 3.1
1989	AIX/6000 v3
1987	AIX/RT 2.2.1
1986	AIX/RT 2.1.2 AIX/RT 2
1984	UNIX System V Release 2
1983	UNIX System V
1982	UNIX System IV
1981	UNIX System III
1978	CB UNIX 3 CB UNIX 2 CB UNIX 1

Chapter 1. IBM Power Systems

The AIX operating system we focus on here can only work on one hardware platform, the IBM Power Systems platform. However, such a limitation brings with it several advantages at the operating system level, including:

- Stability - The limited number of hardware components means that they are finer and less problematic in the operating system.
- Performance - Through knowing the environment in which it operates, the operating system can optimize its work. This is achieved by features such as directing processes to the CPU near the memory of a given process, which results in more efficient processing.

The name "Power" is an acronym for *Performance Optimization With Enhanced RISC*. The server core processor is also known as Power, and the entire server line based on this architecture is called Power Systems. The names used in the technology world tended to change frequently. In this case, it must be acknowledged that the Power Systems name has been around for quite a long time, since 2008 in fact. Indeed, the year 2008 was significant for this architecture, because two separate worlds were merged on one platform: the world of AIX and the world of IBM i. Before the two worlds (server lines) were united, they repeatedly changed their names:

- For AIX systems: RS/6000, pSeries.
- For IBM i systems: AS/400, iSeries.

Currently, an expanded number of operating systems can be used on the IBM Power platform:

- AIX (*Advanced Interactive eXecutive*) - This book is about the AIX operating system.
- IBM i - An operating system with multiple built-in high availability features and an integrated DB2 database. It can be said that it was one of the first object-oriented operating systems. In this system, we work with objects rather than with the usual files.
- Linux in several distributions:
  - Red Hat Enterprise Linux.
  - SUSE Linux Enterprise Server.
  - Ubuntu.

This kind of hardware platform needs unique features in order to attract customer interest. There are several such features:

- **Reliability and availability of systems**. The platform has many RAS (*Reliability, Availability, and Serviceability*) features that make the servers work reliably. The operating system can continue to work in the case of many types of failures, and a given failure can often be eliminated without having to shut down the server. More can be read about this topic in the documentation that can be easily found using the keywords "Power Systems RAS."
- **Server performance**. Many tests have shown that the Power Systems servers can be twice as efficient as the x86 servers of the same class. This gives you the ability to both run more virtual machines on a single server and better consolidate your systems. This can also justify the difference in the price of the Power and x86 solutions.

- **Scalability and virtualization**. Power servers are available in many variants and sizes. The server maximum capacity used to be 256 cores (server 795, Power7). Currently, at the time of publication, the servers can have up to 192 cores (server 880, Power8). On Power servers, we can use PowerVM virtualization for all operating systems (for Linux solutions, KVM can also be used). PowerVM Virtualization has a unique feature, namely the high scalability of its performance. This means that the systems can be fully or partially virtualized. They can use a virtual or physical core as well as virtual or physical LAN and SAN adapters. These features allow you to handle any heavy load, regardless of which server part (CPU, LAN, or SAN) is operating the most.

IBM Power Systems: Server management

To manage the servers at a regular, basic level you can use an HMC or IVM console. However, if you want to manage Power servers in the IAAS (*Infrastructure as a Service*) model, you need PowerVC.

The **Hardware Management Console (HMC)** is a "hardware" console for managing one or more servers in terms of the configuration, maintenance, and virtualization. This type of console is not necessary for a virtualized Power Systems environment. However, when it comes to managing multiple servers and using more advanced virtualization features, it becomes desirable.

In fact, the HMC is usually referred to as an appliance. It is a physical x86 server with a sufficient number of LAN ports and management software installed. The HMC can also be installed as a virtual machine on a hypervisor on an x86 platform. The HMC console is shown in Figure 1-1.

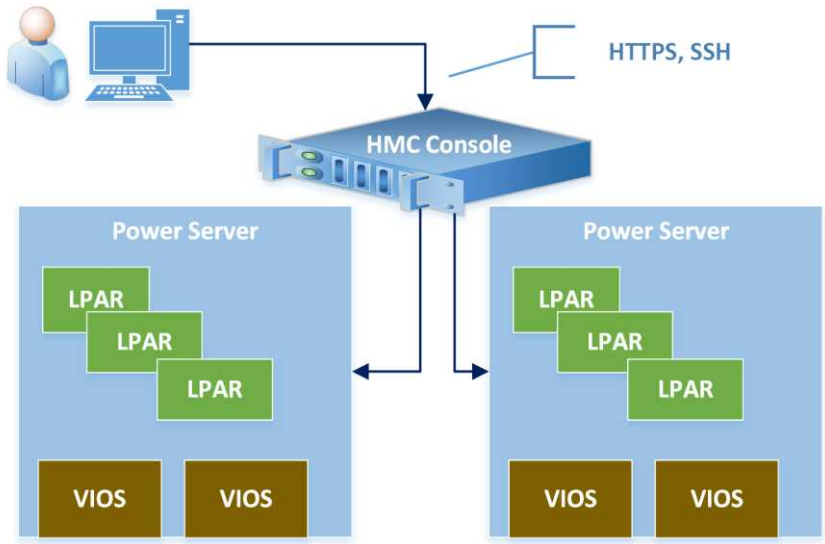

Figure 1-1: The HMC.

The **Integrated Virtualization Manager (IVM)** is a software console that is installed as part of a VIOS if you do not have the HMC. It is convenient to use and sufficient for small, undemanding environments. Despite having some restrictions when compared to the HMC, the use of the IVM

works well in small environments. The primary limitations of this console are the ability to have only one VIOS and the lack of LPAR capability to have physical devices (the terms used here will be expanded in the section on virtualization). The IVM console is shown in Figure 1-2.

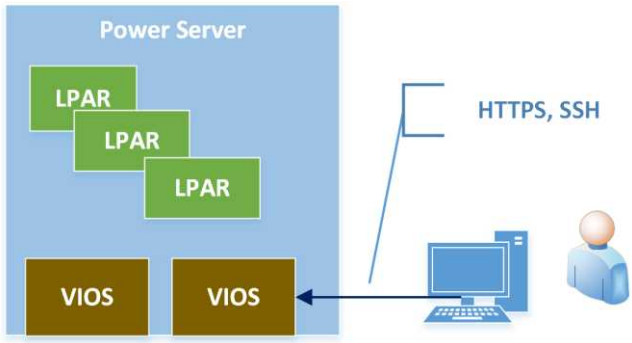

Figure 1-2: The IVM.

The **Power Virtualization Center (PowerVC)** is an advanced automation solution for managing virtual environments. It is based on OpenStack. It enables the automatic deployment of virtual machines with AIX, IBM i, and Linux operating systems on the IBM Power Systems platform. This solution simplifies the administrator's work, and it is a first step toward cloud solutions on this platform. The main features of the product are:

- Simplified server management;
- Quickly create new virtual machines;
- Self-service portal for basic functions;
- Policy-based management of systems;
- Image system management; and
- Machine capture, machine duplication, and automatic deployment.

From an architectural point of view, the PowerVC's cooperation with the Power platform is presented in Figure 1-3. It is worth noting that the solution works with servers supported by the HMC. In the latest versions, it can also work with PowerVM NovaLink, a special LPAR that communicates with the server hypervisor. The use of NovaLink will allow for the wider use of the SDN's (Software Defined Network) functionality, which is developed within OpenStack.

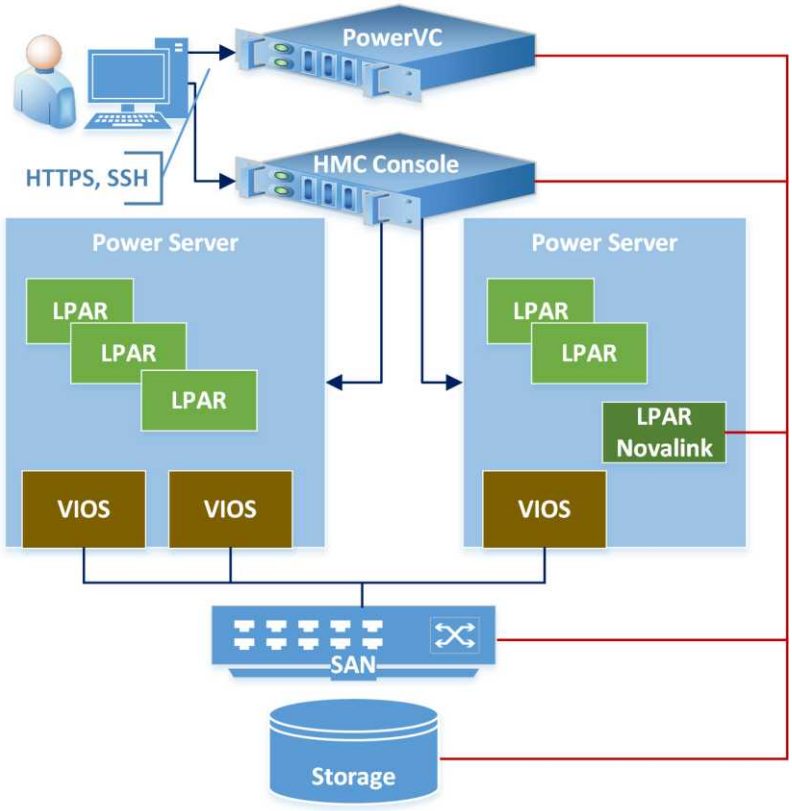

Figure 1-3: The PowerVC.

The PowerVC is actually a separate topic. This publication mentions it, but will not explore it, focusing instead on the operating system and virtualization elements.

Chapter 2. AIX: Basic Information

If you require basic knowledge about an operating system, you should know about the tools necessary for reading the system configuration and parameters, the location of the data, and the configuration tools.

In every UNIX system, the entire set of commands is used to manage the operating system. These commands are used in the shell environment by means of the command line. Such an environment is not user-friendly for new users, although as the user's experience grows, it shows its superiority over graphical interfaces. This superiority includes the ease of building scripts based on commands, the ability to manipulate the command results into the shape that interests us, and the ability to use the output of one command as the input for others.

Using operating system commands is a key skill for any administrator. Nevertheless, a large part of the configuration work can be performed using the configuration tools. For AIX, the primary configuration tool is the SMIT (*System Management Interface Tool*). It allows you to configure a large part of the system parameters. Previously, there was also a WSM graphical tool (*Web-based System Manager*), which enabled the operating system to be configured, although it was removed with AIX 7.1. Another tool that helped to manage some aspects of the system was IBM Systems Director. At the time of writing this book, however, it had been withdrawn from sale.

As with other UNIX systems, some of the configuration data are stored in flat files. The feature that distinguishes AIX is the fact that the operating system has some configuration data in the configuration database represented by ODM (*Object Data Manager*) files.

The SMIT (System Management Interface Tool)

The SMIT is an AIX-specific configuration tool. Most of the command line operations can be performed with the SMIT, without having to know what command to use and what parameters to use with that command.

The SMIT provides a menu from which you can select what to do on your system. It is very important for the administrator that the manner in which the tool performs tasks is very clear. All the menu actions are performed using system commands with the appropriate parameters or using system scripts. Commands called by the SMIT can be previewed and parsed before execution, and it can also be used to build custom scripts for system management.

This tool is available in two modes with the same capabilities, differing only in terms of the environment in which it operates:

- **Graphical mode** - Launched using the X Window System (X11). It is started using the *smit* or *smit -M*. With the *smit* command without parameters, the tool can run in graphical or text mode. If the graphical environment is running or the DISPLAY variable is set to point to an external host displaying screen (for example, a PC with Xming, Cygwin, etc.), the tool will launch in graphical mode. Otherwise, it will start in text mode. The appearance of the graphical

version is shown in Figure 2-1.

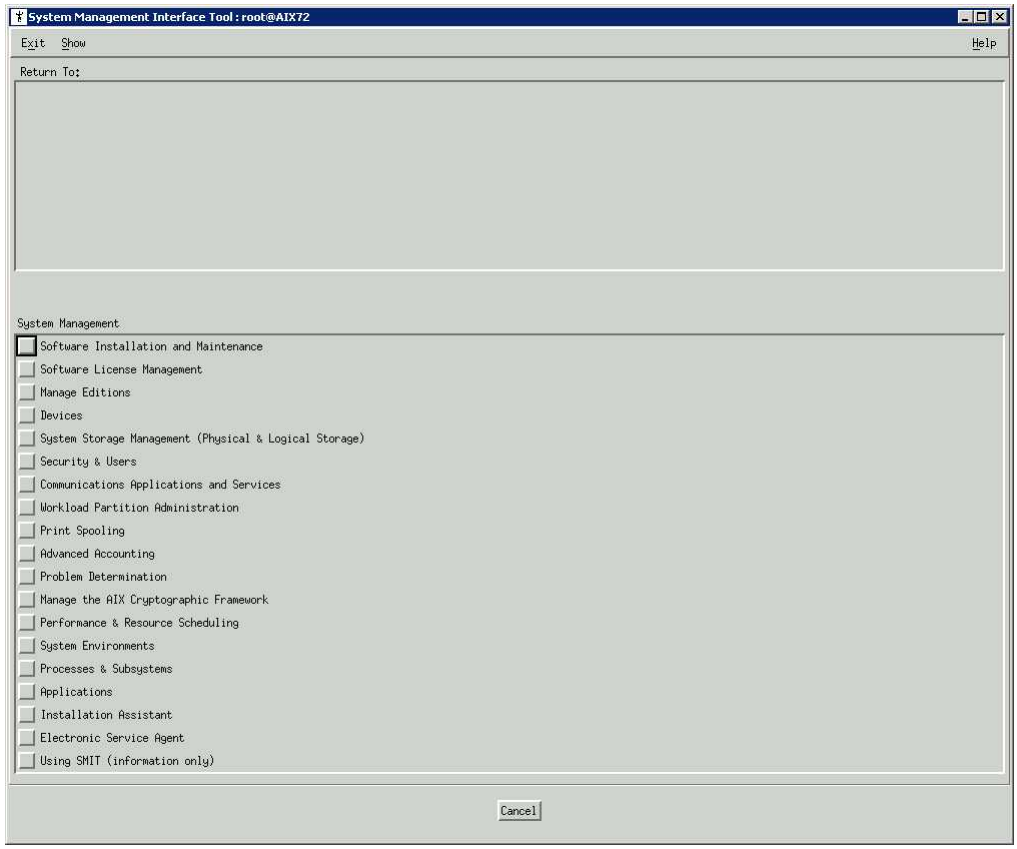

Figure 2-1: The SMIT in graphical mode.

- **Text mode** - To be run from the text console or by the terminal emulator. It starts with a *smitty* or *smit -C* command. You can also use the *smit* command without parameters if you have the appropriate system configuration. The appearance of the text version is shown in Figure 2-2.

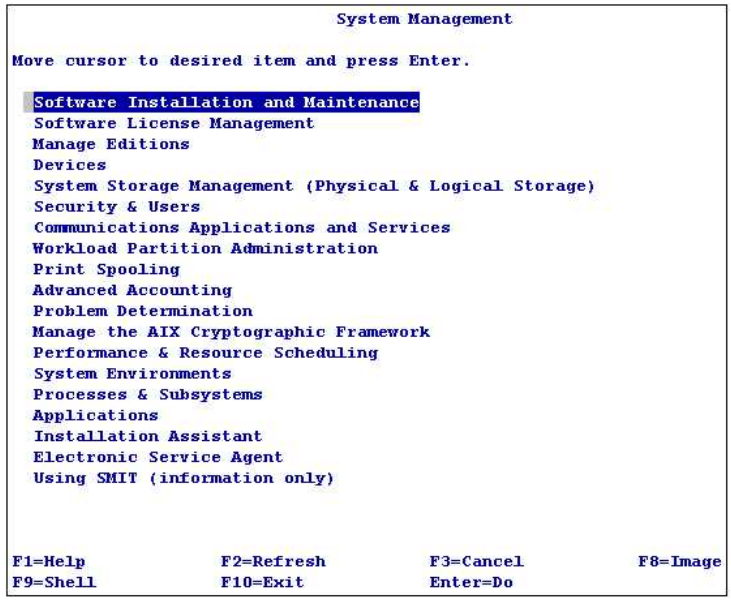

Figure 2-2: The SMIT in text mode.

The text version of the SMIT is most commonly used, since it can be operated using only the keyboard. In the case of the graphical version, a mouse is required. Therefore, throughout the rest of the book, most SMIT-related screens will be presented in text mode.

The name of each option available in the menu uniquely identifies the activity for which it was created. To obtain more information about the option you want to use, use the *F1* key to display additional information. The *F1* key is always available when working with the SMIT tool.

Before you run the action selected through the menu, you can view the command that will be called. To view the command, use the *F6* key. Figure 2-3 shows how this function works:

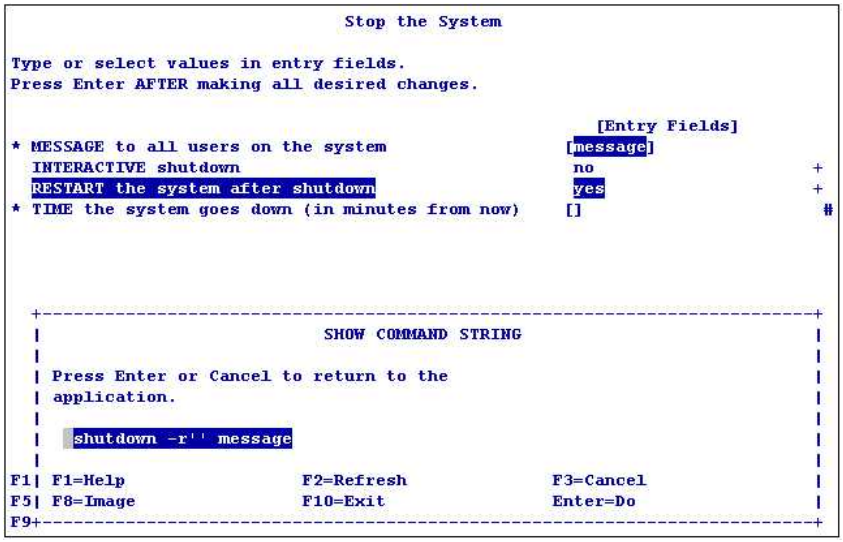

Figure 2-3: The SMIT - command string to execute.

In the above example, you can see that after setting the appropriate options in the SMIT menu, the system will run the corresponding command with the appropriate parameters based on the entered information. In the above case, it is the command to shut down and restart the system:

```
# shutdown -r'' message
```

The SMIT can be run using the fast path, so that the user is immediately moved to a specific screen. You can check which fast path leads to the currently active window using the *F8* key, which is described as *image*. This way, if you want to launch a menu on a user management menu, you can use a quick path *"user"*, for example:

```
# smit user
```

It is worth noting the meaning of several symbols that you can encounter when using the SMIT menu:

- **[ ]** - Indicates that you can enter the selected values in this field.
- **<** or **>** - Indicates that the left or right side contains text that does not fit on the screen. You can access it using the arrow keys.
- ***** - Indicates that the field is required.
- **+** - Indicates that a field selection list is available. To view the list, press the *F4* key.
- **#** - Indicates that the field is numeric.
- **x** - Indicates that the field requires a hexadecimal value.

Files created by the SMIT

When working with the SMIT tool in the home directory of the user (the directory pointed to by the *$HOME* system variable), two files are created: *$HOME/smit.log* and *$HOME/smit.script*.

The *smit.log* file stores all the actions that the user performed while on the SMIT menu. This allows you to accurately trace its actions. A sample log generated by the SMIT is shown below. It shows that we

entered the SMIT, chose *System Storage Management (Physical & Logical Storage)*, and then left the tool.

```
# tail -20 smit.log   #Display the last 20 lines of the smit.log file.

[Oct 14 2016, 06:14:45]
    Starting SMIT

(Menu screen selected,
        FastPath    = "top_menu",
        id_seq_num  = "0",
        next_id     = "top_menu",
        title       = "System Management".)

(Menu screen selected,
        FastPath    = "storage",
        id_seq_num  = "030",
        next_id     = "storage",
        title       = "System Storage Management (Physical & Logical Storage)".)


[Oct 14 2016, 06:15:05]
    Exiting SMIT
```

Each action triggered from the SMIT is run by calling a specific command or script built using the tool based on the settings chosen on the menu. A command or script that is generated and called in this way is written to the *smit.script* file. The generated script shows how the action was performed. Its parts can also be copied and used to build your own scripts. The following example shows the script generated by the SMIT when creating a file system from the menu:

```
# tail -46 smit.log
    Command_to_Execute follows below:
>> x() {
LIST=
FLAG=0
for i in "$@"
do
        case "$i" in
        Megabytes)      FLAG=1;;
        Gigabytes)      FLAG=2;;
        512bytes)       ;;
        quota=all)      LIST="$LIST \"quota=userquota,groupquota\"" ;;
        size=*) case "$FLAG" in
                1)      LIST="$LIST \"$i\"M"
                        FLAG=0;;
                2)      LIST="$LIST \"$i\"G"
                        FLAG=0;;
                0)      LIST="$LIST \"$i\""
                        ;;
                esac
                ;;
        *)      LIST="$LIST \"$i\""
                ;;
        esac
done
eval /usr/sbin/crfs -v jfs2 $LIST
}
x -g'rootvg' 'Megabytes' -a size='100' -m'/testfs1' -A'yes' -p'rw' -a agblksize='4096' -a
isnapshot='no'


    Output from Command_to_Execute follows below:
```

```
---- start ----

File system created successfully.
114480 kilobytes total disk space.
New File System size is 229376

---- end ----

[Oct 14 2016, 06:20:08]



[Oct 14 2016, 06:20:13]
    Exiting SMIT
```

The ODM (Object Data Manager)

A large part of the system configuration data is stored in an ODM-managed database. Examples of the information stored by the ODM include:

- Device configuration in the operating system;
- Information displayed by the SMIT tool (forms, dialog boxes, etc.);
- Information about the installed software;
- Network configuration; and
- System resources information.

The meaning and existence of this system element are similar to those of a Windows system registry. The main differences between the two are:

- Increased stability of this mechanism on AIX;
- Using the ODM primarily for system-related information rather than for information generated by the installed applications; and
- Access to the required data using system commands, in most cases without the need for manual intervention in the ODM database.

The ODM data are stored in three locations:

- */usr/lib/objrepos*. (This is the file in which most of the information is stored.)
- */usr/share/lib/objrepos*.
- */etc/objrepos*.

"Manual" modification of database content is usually not recommended, since it may lead to unexpected system behavior. However, if it does prove necessary, we should protect the above-mentioned files (copy them to a safe place) so that they can be restored quickly in case of problems. In addition to copying the ODM files, always remember to make a full system backup, since it may be useful in a crisis.

In this case, "manual" ODM modification means the direct manipulation of the ODM using dedicated commands. Other modifications are made at every step, often without the user being aware of it. For

example, these unconscious actions arise when creating structures such as a file system or a volume group, or adding or changing the characteristics of the hardware. The following commands are used for the "manual" modification of the ODM database:

- *odmadd*.
- *odmchange*.
- *odmcreate*.
- *odmdelete*.
- *odmdrop*.
- *odmget*.
- *odmshow*.

These commands will not be described further, since it is rarely necessary to use them. Nevertheless, it is worth knowing that in extreme situations there are such tools available.

The SRC (System Resource Controller)

One of the tools that facilitates system management is the System Resource Controller, which manages the subsystems and subservers.

- Subsystem - This is a program, process, or set of programs or processes that perform specific tasks. Subsystems are combined into groups that perform more tasks.
- Subserver - This is a subsystem that controls the running of other processes/services. An example subserver in AIX is the *inetd*, which, if necessary, needs to run the services described in its */etc/inetd.conf* configuration file.

If software installed on a server is a separate entity and operates on a persistent background, then there is usually a separate subsystem. The SRC has a number of advantages, including:

- Easy to start, stop, and refresh the subsystem configurations;
- Logs problems with subsystems in the system error log (errorlog);
- Tracks the subsystem status; and
- Remote control of the subsystems (one system may have control over the subsystems of the other system).

The process responsible for the SRC is */usr/sbin/srcmstr*. It is started when the system boots with the corresponding entry in */etc/inittab*:

```
srcmstr:23456789:respawn:/usr/sbin/srcmstr # System Resource Controller
```

In order to be able to remotely manage SRC resources, the *scrmstr* on the remote system must be run with the *r* parameter.

The following subsections discuss the basic capabilities of the SRC.

Display information about the subsystems

You can use the *lssrc* command to display the subsystem information. It presents the defined subsystems in the following format:

- Name of the subsystem;
- The group to which it belongs;
- The process identifier (in case the subsystem is running); and
- The subsystem status: *active* - activated or *inoperative* - non-activated.

An example of running the *lssrc* command follows.

```
# lssrc -a   #Display all defined subsystems.
Subsystem         Group            PID          Status
 syslogd          ras              4063618      active
 sendmail         mail             4456844      active
 portmap          portmap          3735900      active
 inetd            tcpip            3604964      active
 snmpd            tcpip            3539326      active
 snmpmibd         tcpip            4129158      active
 aixmibd          tcpip            4194696      active
 hostmibd         tcpip            4522382      active
 aso                               4587920      active
 biod             nfs              4260236      active
 rpc.lockd        nfs              6095346      active
 clcomd           caa              6160852      active
 qdaemon          spooler          4849944      active
 writesrv         spooler          4784430      active
 pfcdaemon                         6291720      active
 ctrmc            rsct             5439924      active
 IBM.ConfigRM     rsct_rm          6947288      active
 IBM.MgmtDomainRM rsct_rm          7078364      active
 IBM.HostRM       rsct_rm          5570904      active
 IBM.ServiceRM    rsct_rm          6750676      active
 IBM.DRM          rsct_rm          4391316      active
 lpd              spooler                       inoperative
 keyserv          keyserv                       inoperative
 ypbind           yp                            inoperative
 gsclvmd                                        inoperative
 cdromd                                         inoperative
 ndpd-host        tcpip                         inoperative
 ndpd-router      tcpip                         inoperative
 netcd            netcd                         inoperative
 tftpd            tcpip                         inoperative
 routed           tcpip                         inoperative
# All unnecessary information has been omitted.
```

Start, stop, and refresh the subsystem configurations

The *startrc*, *stopsrc*, and *refresh* commands are used to perform the main actions. Each of them can operate on a single subsystem (parameter *-s* subsystem_name) or on their group (parameter *-g* group_name). Examples of their use follow.

To stop the *automountd* subsystem:

```
# lssrc -s automountd   #Automountd before stopping.
Subsystem         Group            PID          Status
 automountd       autofs           180362       active

# stopsrc -s automountd   #Stop automountd.
0513-044 The automountd Subsystem was requested to stop.

# lssrc -s automountd   #Automountd after stopping.
Subsystem         Group            PID          Status
 automountd       autofs                        inoperative
```

To start the *nfs* subsystem group:

```
# lssrc -g nfs   #Group of nfs subsystems before startup.
Subsystem         Group            PID          Status
 biod             nfs                           inoperative
 nfsd             nfs                           inoperative
 rpc.mountd       nfs                           inoperative
 nfsrgyd          nfs                           inoperative
 gssd             nfs                           inoperative
 rpc.lockd        nfs                           inoperative
 rpc.statd        nfs                           inoperative

# startsrc -g nfs   #Start subsystems belonging to the nfs group
0513-059 The biod Subsystem has been started. Subsystem PID is 184426.
0513-059 The nfsd Subsystem has been started. Subsystem PID is 176242.
0513-059 The rpc.mountd Subsystem has been started. Subsystem PID is 139312.
0513-059 The nfsrgyd Subsystem has been started. Subsystem PID is 213098.
0513-059 The gssd Subsystem has been started. Subsystem PID is 196766.
0513-059 The rpc.lockd Subsystem has been started. Subsystem PID is 307418.
0513-059 The rpc.statd Subsystem has been started. Subsystem PID is 303318.

# lssrc -g nfs   #Group of nfs subsystems after startup.
Subsystem         Group            PID          Status
 biod             nfs              184426       active
 nfsd             nfs              176242       active
 rpc.mountd       nfs              139312       active
 rpc.lockd        nfs              307418       active
 rpc.statd        nfs              303318       active
 nfsrgyd          nfs                           inoperative
 gssd             nfs                           inoperative
```

The *inoperative* status of the last two subsystems indicates that they failed to start. For more information about the cause of the failure, you can check the system error log (*errpt* command).

To refresh the subserver configuration:

```
# refresh -s inetd   #Refresh the inetd subserver configuration.
0513-095 The request for subsystem refresh was completed successfully.
```

The *inetd* subsystem (subserver) configuration is located in the */etc/inetd.conf* file. It is read when starting the *inetd* daemon. In order for changes made to this file during the subsystem configuration to work, you must instruct the subsystem to re-read the file. You can do this by running the *refresh* command. Alternatively, you can use the *kill -HUP inetd_process_number* command.

Check information about the devices in the system

It is a primary responsibility of the system administrator to know the configuration of the system devices. A separate device on AIX is any hardware or virtual component (such as a network interface, disk, logical volume, or memory). Each device is represented by a file in the */dev* directory, and it can be manipulated in the same way that we operate on files.

There are virtual devices not related to hardware:

- ***/dev/zero*** - The device returns zero at reading. It can be used, for example, to erase the contents of the disk (*cat /dev/zero > /dev/hdisk0*).
- ***/dev/null*** - An empty device. By redirecting any information to this device, we cause it to be ignored. For example, *find / -name smi* 2> /dev/null* will redirect the error output (*2*) to the */dev/null* device, thereby causing the error messages to not be displayed on the screen.

There are also special devices related to operating system objects. They specify certain operating system parameters. For example:

- ***sys0*** - A system object that contains properties and some system-wide parameters; and
- ***inet0*** - A system object that contains properties and some network parameters.

Objects of this type are not mapped as a file in the */dev* directory.

There are several commands needed to check the system configuration, including *lsdev*, *lsattr*, *lscfg*, and *prtconf*. It is good to know that the information provided by these commands comes from the ODM database. These commands, like almost all UNIX systems, have many parameters. The following sections describe the basic capabilities of these commands.

The lsdev command

The *lsdev* command displays general information about the logical and physical devices on the system.

Key parameters:

- ***-C*** - Devices defined in the ODM database (existing within the system).
- ***-P*** - Predefined devices (all that the system can handle). In other words, a list of built-in device drivers.
- ***-c device_class*** - Shows only devices of a certain class, such as *disk*, *bus*, *adapter*, or *if*.
- **-H** - Displays the header.

```
# lsdev -CH
name       status    location description

L2cache0   Available          L2 Cache
cache0     Defined            SSD Cache virtual device
```

```
cd0        Available          Virtual SCSI Optical Served by VIO Server
cengine0   Available          SSD Cache engine
cluster0   Available          Cluster Node
en0        Available          Standard Ethernet Network Interface
ent0       Available          Virtual I/O Ethernet Adapter (l-lan)
et0        Defined            IEEE 802.3 Ethernet Network Interface
fslv00     Available          Logical volume
hd1        Defined            Logical volume
hd2        Defined            Logical volume
hd3        Defined            Logical volume
hd4        Defined            Logical volume
hd5        Defined            Logical volume
hd6        Defined            Logical volume
hd8        Defined            Logical volume
# All unnecessary information has been omitted.
proc0      Available 00-00    Processor
proc4      Available 00-04    Processor
# All unnecessary information has been omitted.
```

The above command displays all the devices in the system. The *location* column provides information about the physical location of the device. The numbers displayed here are the so-called location codes, which can be interpreted using the service guide available for each server model. This way you can obtain information such as the PCI slot in which the interface is located, where this drive is located, etc. Further, the status column shows the state of the device. The device can be in two states:

- ***Available*** - The device is fully accessible, that is, it can be used. In various IBM sources, we find the device described as "configured."
- ***Defined*** - The device exists within the system, but we cannot operate on it. This device is referred to as "unconfigured."

To see the difference between the two states, let's look at the *en0* device that is configured (*available* state) and the *et0* device that is unconfigured (*defined* state). The *en0* is a network adapter. It has an assigned IP address, and it is the only device that can communicate with other computers in the network. The *et0* is a device without an IP address, and it has not been activated. You can say that the status "available" refers to an enabled device, while the status "defined" refers to a disabled device.

Another example of using the *lsdev* command to display information on all the *disk* class devices:

```
# lsdev -Cc disk
hdisk0 Available 10-60-00-10,0 16 Bit LVD SCSI Disk Drive
hdisk1 Available 10-60-00-11,0 16 Bit LVD SCSI Disk Drive
```

The lsattr command

The *lsattr* command displays the parameters of devices, some of which can be changed using the *chdev* command.

Key parameters:

- ***-E*** - Displays the currently set values.
- ***-D*** - Displays the default values for the device.
- ***-R*** - Displays the range of values that can be assigned to the individual attribute of the device.

- ***-l deviceName*** - Specifies what device the command applies to. This is a mandatory parameter.
- ***-a attribute*** - Specifies the attribute to which the command applies.

```
# lsattr -El hdisk0
PCM             PCM/friend/vscsi                 Path Control Module        False
PR_key_value    none                             N/A                        True
algorithm       fail_over                        Algorithm                  True
hcheck_cmd      test_unit_rdy                    Health Check Command       True+
hcheck_interval 0                                Health Check Interval      True+
hcheck_mode     nonactive                        Health Check Mode          True+
max_transfer    0x40000                          Maximum TRANSFER Size      True
pvid            00f621774d9300420000000000000000 Physical volume identifier False
queue_depth     8                                Queue DEPTH                True
reserve_policy  no_reserve                       Reserve Policy             True
```

The above command displays the current parameters of the *hdisk0* device. The following information is displayed: The parameter's name, its value, a brief description, and whether the parameter can be changed (*True*) or not (*False*). For some parameters, clarifying their operation would require a deep knowledge of how AIX drives work. At this stage, however, we will focus on determining how to obtain information rather than on how to interpret it in detail.

The following examples show the additional capabilities of the *lsattr* command, that is, displaying the value that a given parameter can accept as well as displaying its default value.

```
# lsattr -Rl hdisk0 -a queue_depth
1...256 (+1)   #The parameter can range from 1 to 256 with a jump of 1.

# lsattr -Dl hdisk0 -a queue_depth
queue_depth 8 Queue DEPTH True   #The default value of queue_depth for this device is 8.
```

Let's now consider the *sys0* device. This is a virtual device that symbolizes the general configuration of our system. By looking at the parameters of this device, we can obtain a lot of interesting information.

```
# lsattr -El sys0
SW_dist_intr    false             Enable SW distribution of interrupts             True
autorestart     true              Automatically REBOOT OS after a crash            True
boottype        disk              N/A                                              False
capacity_inc    0.01              Processor capacity increment                     False
capped          false             Partition is capped                              False
chown_restrict  true              Chown Restriction Mode                           True
clouddev        0                 Recreate ODM devices on next boot                True
conslogin       enable            System Console Login                             False
cpuguard        enable            CPU Guard                                        True
dedicated       false             Partition is dedicated                           False
enhanced_RBAC   true              Enhanced RBAC Mode                               True
ent_capacity    2.00              Entitled processor capacity                      False
frequency       6400000000        System Bus Frequency                             False
fullcore        false             Enable full CORE dump                            True
fwversion       IBM,AL730_149     Firmware version and revision levels             False
ghostdev        0                 Recreate ODM devices on system change/modify PVID True
id_to_partition 0X80000341CC500016 Partition ID                                    False
id_to_system    0X80000341CC500000 System ID                                       False
iostat          false             Continuously maintain DISK I/O history           True
keylock         normal            State of system keylock at boot time             False
log_pg_dealloc  true              Log predictive memory page deallocation events   True
max_capacity    2.00              Maximum potential processor capacity             False
max_logname     9                 Maximum login name length at boot time           True
maxbuf          20                Maximum number of pages in block I/O BUFFER CACHE True
```

```
maxmbuf         0                   Maximum Kbytes of real memory allowed for MBUFS   True
maxpout         8193                HIGH water mark for pending write I/Os per file   True
maxuproc        128                 Maximum number of PROCESSES allowed per user      True
min_capacity    0.10                Minimum potential processor capacity              False
minpout         4096                LOW water mark for pending write I/Os per file    True
modelname       IBM,8233-E8B        Machine name                                      False
ncargs          256                 ARG/ENV list size in 4K byte blocks               True
nfs4_acl_compat secure              NFS4 ACL Compatibility Mode                       True
ngroups_allowed 128                 Number of Groups Allowed                          True
pre430core      false               Use pre-430 style CORE dump                       True
pre520tune      disable             Pre-520 tuning compatibility mode                 True
realmem         2097152             Amount of usable physical memory in Kbytes        False
rtasversion     1                   Open Firmware RTAS version                        False
sed_config      select              Stack Execution Disable (SED) Mode                True
systemid        IBM,02062177P       Hardware system identifier                        False
variable_weight 128                 Variable processor capacity weight                False
```

Examples of the information that you can obtain here include:

- The firmware version of the server - *fwversion*;
- Whether the system collects statistics on input/output operations - *iostat*;
- The amount of memory available - *realmem*; and
- The server serial number - *systemid*.

The lscfg command

The lscfg command displays detailed information about the devices in the system. This way, we can determine the serial numbers of the devices in the system, their firmware versions, and their manufacturer. The information obtained can be very detailed, for example, we can read the physical location of the devices within the server.

Key parameters:

- ***-v*** - Detailed information about the devices.
- ***-p*** - Additional platform-specific information.
- ***-l deviceName*** - Specifies the device to use in the command (without this parameter, all the devices are used).

```
# lscfg -vpl fcs0
  fcs0             U78A0.001.DNWHWGD-P1-C2-T1  8Gb PCI Express Dual Port FC Adapter
(df1000f114108a03)

        Part Number.................10N9824
        Serial Number...............1C019087F8
        Manufacturer................001C
        EC Level....................D76482B
        Customer Card ID Number.....577D
        FRU Number..................10N9824
        Device Specific.(ZM)........3
        Network Address.............10000000C9B01108
        ROS Level and ID............02781174
        Device Specific.(Z0)........31004549
        Device Specific.(Z1)........00000000
        Device Specific.(Z2)........00000000
        Device Specific.(Z3)........09030909
```

```
        Device Specific.(Z4)........FF781116
        Device Specific.(Z5)........02781174
        Device Specific.(Z6)........07731174
        Device Specific.(Z7)........0B7C1174
        Device Specific.(Z8)........20000000C9B01108
        Device Specific.(Z9)........US1.11X4
        Device Specific.(ZA)........U2D1.11X4
        Device Specific.(ZB)........U3K1.11X4
        Device Specific.(ZC)........00000000
        Hardware Location Code......U78A0.001.DNWHWGD-P1-C2-T1


  PLATFORM SPECIFIC   #Information for the -p parameter.
  Name:  fibre-channel
    Model:  10N9824
    Node:  fibre-channel@0
    Device Type:  fcp
    Physical Location: U78A0.001.DNWHWGD-P1-C2-T1
```

The above example presents detailed hardware information about the physical Fiber Channel controller, for example:

- The location and interface speed - the first line of the result;
- The manufacturer's data - "Manufacturer;" and
- The firmware version – "ROS Level and ID."

The prtconf command

The prtconf command displays the system information in the most user-friendly manner. This command presents us with a set of the most frequently searched for information that details various aspects of the system configuration. It has the answer to 90% of the very basic questions asked about the system and the server. It can be called up with different parameters that are used to represent the information parts. A sample of the information that can be obtained using *prtconf* follows:

```
# prtconf
System Model: IBM,8233-E8B
Machine Serial Number: 062177P
Processor Type: PowerPC_POWER7
Processor Implementation Mode: POWER 7
Processor Version: PV_7_Compat
Number Of Processors: 2
Processor Clock Speed: 3300 MHz
CPU Type: 64-bit
Kernel Type: 64-bit
LPAR Info: 22 SB5437_AIX72_c2
Memory Size: 2048 MB
Good Memory Size: 2048 MB
Platform Firmware level: AL730_149
Firmware Version: IBM,AL730_149
Console Login: enable
Auto Restart: true
Full Core: false
NX Crypto Acceleration: Not Capable

Network Information
        Host Name: AIX72c2
```

```
        IP Address: 10.10.177.4
        Sub Netmask: 255.255.0.0
        Gateway: 10.10.10.1
        Name Server:
        Domain Name:

Paging Space Information
        Total Paging Space: 512MB
        Percent Used: 2%

Volume Groups Information
==============================================================================
Active VGs
==============================================================================
rootvg:
PV_NAME           PV STATE          TOTAL PPs   FREE PPs    FREE DISTRIBUTION
hdisk0            active            511         214         14..00..00..98..102
==============================================================================
# All unnecessary information has been omitted.
```

Chapter 3. AIX: Virtualization

The origins of systems virtualization are already far behind us. Today, most operating systems are subject to virtualization. Almost all of us have some virtual machines on our laptops that can be run at any time, both for educational purposes and to do work in an isolated environment. The same applies to commercial solutions and server virtualization.

On the market commonly known as "x86," there are many potential server virtualization products available. You can choose between VMware ESXi hypervisor, Hyper-V, KVM, or XEN. However, on the market for UNIX platforms (such as IBM POWER or Oracle SPARC) the choice is more limited, and you can only use virtualizers provided by the platform provider.

Each of the above solutions has both advantages and disadvantages. An obvious disadvantage of the POWER platform is the limited choice of available virtualization methods. However, this actually has the advantage of ensuring the better integration of the virtualizer with the platform. With it, you gain higher efficiency and availability of systems, as well as a higher level of security.

The Power platform provides two types of virtualizers. PowerVM, which is advanced and has been available for many years, and a simple solution from the world of Linux, namely KVM. In addition, you can apply virtualization at the AIX operating system level by using WPARs (Workload Partitions). This chapter will discuss PowerVM, which is the most popular means of virtualization available on the Power platform.

To properly understand this chapter, it is advisable to have basic, general knowledge about the issues related to the virtualization of operating systems. If you are interested in aspects that relate to only the operating system, you can skip this chapter and return to it later.

PowerVM

PowerVM is the most widely recognized and most powerful virtualization method of all three methods mentioned above. It has been used for over a dozen years. If you look deeper into the genesis of this method, you will notice that it derives from solutions that existed on mainframe platforms.

The described virtualization platform is characterized by very good integration with both the hardware platform and the operating system. The reason for such deep integration is the creation of all these layers within one company, where the flow of information and mutual interests converge.

A very important feature of this virtualization method is its highly scalable performance. This is achieved thanks to one of the other key features, that is, the ability to assign virtual and physical resources to virtualized systems. With this method, if you need a very high bandwidth LAN or SAN network, you can allocate any number of physical adapters to handle the associated traffic.

As of 2018, PowerVM is sold in two versions: *standard* and *enterprise*. They differ in terms of only a few additional advanced features that are available in the higher version, namely *Active Memory Sharing*, *Live Partition Mobility*, and *PowerVP Performance Monitor*. The differences between the standard and enterprise

versions are presented in Table 3-1.

Table 3-1: PowerVM - standard vs enterprise versions.

Feature	PowerVM version	
	Standard	Enterprise
Virtual I/O Server	OK (DUAL VIOS)	OK (DUAL VIOS)
Suspend/Resume	OK	OK
NPIV	OK	OK
Shared Processor Pools	OK	OK
Shared Storage Pools	OK	OK
Thin Provisioning	OK	OK
Active Memory Sharing	-	OK
Live Partition Mobility	-	OK
PowerVP Performance Monitor	-	OK
SR-IOV	OK	OK

The features presented in the table are described in more detail in this chapter.

Components of virtualization

The key to understanding the way in which PowerVM virtualization works is to split the entire mechanism into the basic parts that perform various functions throughout the solution. This is demonstrated in Figure 3-1.

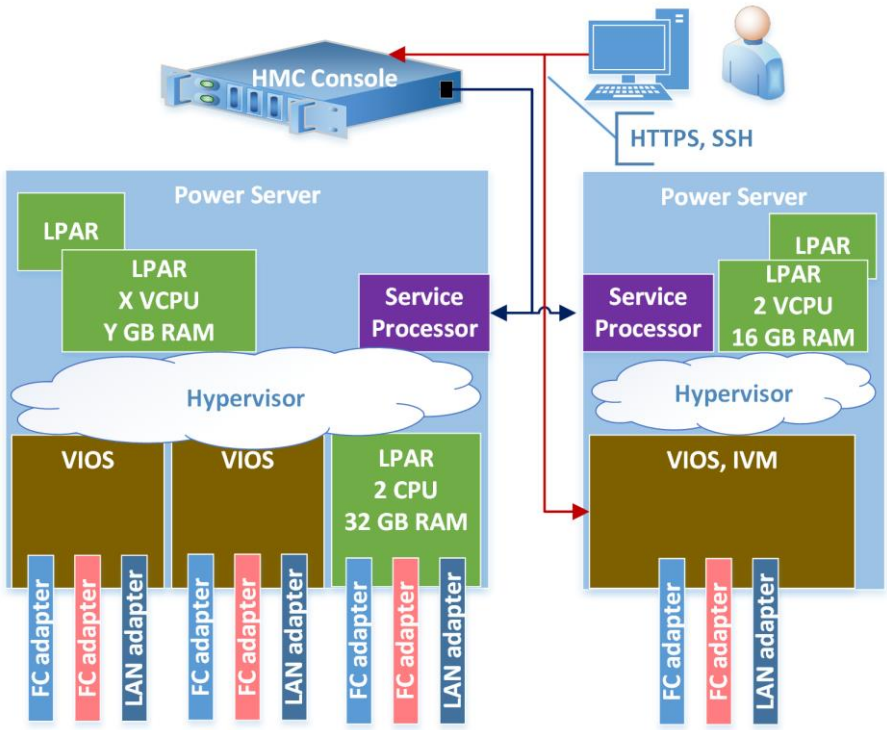

Figure 3-1: Components of PowerVM virtualization.

The basic parts necessary for effective virtualization are:

An ***LPAR***, ***logical partition***, ***partition***, ***micro-partition***, and ***SPLPAR (Shared Processor Logical Partition)***, which are terms that are often used interchangeably. The above terms refer to a virtual system (virtual server) that runs on the POWER platform. So, similar to the "virtual machine" in the x86 world. The given LPAR has virtual resources, for example, a processor, memory, and I/O devices (disks, LAN, SAN). It can also have physical devices, such as a physical processor and physical IO adapters. You can run this LPAR using the management console and install the operating system. From a logical point of view, the VIOS server described below is an LPAR with the appropriate software installed.

Historically, a micro-partition is a newer concept than an LPAR. It indicates a reduction in the granulation of the resources allocated to the virtual entity. In the first implementations of virtualization, the processor resource allocation was at the level of the entire processor (core). Later, you could allocate at the level of the processor's fractions. Today, the above-mentioned terms colloquially mean the same thing, and they will be used in such a way in this book.

The unique feature of the virtualization of the POWER platform is that an LPAR can be fully virtualized or it can have physical components (dedicated processor cores, I/O adapters for LAN, SAN, and other solutions). The different types of LPARs are shown in Figure 3-2. An important feature of any LPAR is the ability to install non-AIX systems, such as IBM i, which was historically known as OS/400, or Linux (Red Hat, SLES, Ubuntu).

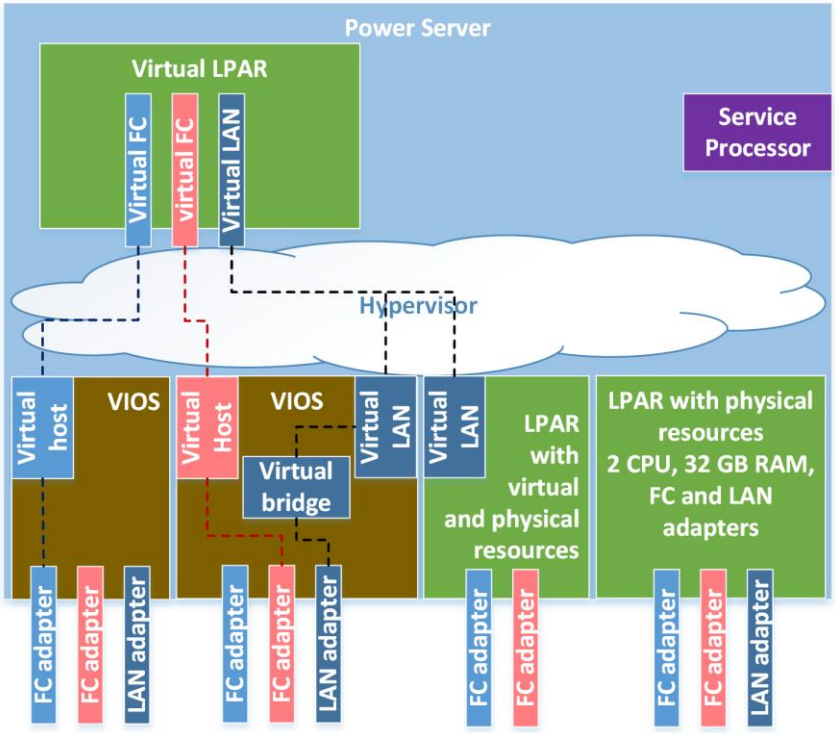

Figure 3-2: Different types of LPARs.

It is worth noting that the LPAR configuration is stored in two places, namely in the partition profile and in its "current configuration." Information is important because the parameters in both places do not have to overlap. For example, if you add a processor, the current configuration will reflect this operation, but the profile will not unless you explicitly update it.

The ***Hardware Management Console (HMC)*** is a hardware console that is used to manage one or more servers in terms of the configuration, maintenance, and management of some of the virtualization features. This type of console is not a necessary element of a standard virtualized POWER environment. However, it becomes necessary when you manage multiple servers and use more advanced virtualization functions.

In fact, the HMC is usually an appliance, that is, a physical x86 server with sufficient LAN ports and installed management software. The HMC can also be installed as a virtual machine on a hypervisor on the x86 platform. The way in which the user interacts with the console, as well as the connection between the console and the servers, are shown in Figure 3-3.

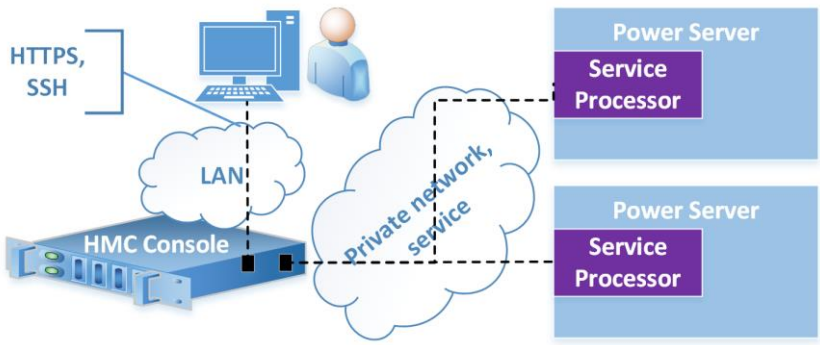

Figure 3-3: HMC network diagram.

The ***IVM (Integrated Virtualization Manager)*** is a "software" console that is installed as part of the VIOS if you do not have an HMC. It is convenient to use and sufficient for small, undemanding environments. The use of the IVM is an appropriate solution in such environments despite it having some limitations when compared to the HMC. The differences in the functionalities provided by the two consoles are presented in Table 3-2.

Table 3-2: Differences in HMC and IVM functionality.

Feature	HMC	IVM
Virtualization, create multiple LPARs on the server	YES	YES
Many servers connected to one console	YES	NO
Assigning physical components (LAN, SAN adapters) to the LPAR	YES	NO
DUAL VIOS	YES	NO
Live Partition Mobility	YES (HMC-HMC)	YES (IVM-IVM)
NPIV	YES	YES
Shared Ethernet Adapter (SEA) with HA option	YES (DUAL VIOS)	NO

The base function of the ***hypervisor*** is controlled by lightweight code embedded in the server, which is part of its firmware. The role of the hypervisor is the virtualization of the CPU, the memory, and the internal part of the LAN. Yet, not all virtualization is based on a hypervisor. For instance, it does not virtualize I/O devices. A separate part known as the Virtual I/O Server (VIOS) is used for I/O virtualization.

Separating the virtualization of the CPU and the memory from the virtualization of the I/O makes a lot of sense in terms of server performance and availability. Thanks to the short code located closer to the "heart" of the server (in its firmware), the server is able to manage resources more efficiently while being less vulnerable to failure. With the advanced functions of some server models, it is possible to enable hypervisor mirroring at the level of the RAM, which can also significantly affect its availability. However, the nature of the I/O operation allows for the use of many VIOS.

The ***VIOS (Virtual I/O Server)*** is the part responsible for the virtualization of I/O operations. The VIOS provides LAN and disk space for LPARs. In fact, it is an LPAR with installed software, which can in turn be described as the AIX customization for performing the function of an I/O server. There may be many VIOSs on a single server, although usually only one or two are used. Two VIOSs are used

when you want to ensure maximum system availability and allow the Virtual I/O Server to upgrade without shutting down the client LPARs. The VIOS functionality is described in more detail later in this chapter.

Processor virtualization

The hypervisor manages the virtualization of the processor and the memory. As mentioned earlier, the hypervisor is an integral part of the server, which is supplied with its firmware. The code is small and close to the hardware. Thanks to this, it can work more efficiently and is more stable than solutions operating without this closeness.

Processor virtualization has an advantage over the use of a physical processor in that you can better utilize the entire platform. The load on a large server that has multiple processors running dozens of LPARs often reaches 60–80% while still maintaining performance stability. When using dedicated physical servers instead of LPARs, their use often fluctuates between 10–20%. Another important feature is the possibility of a dynamic response to the increasing and decreasing load. In most cases, adding or removing subsequent processors to/from the system involves a few clicks that do not require a restart. The same action in the case of physical servers necessitates an upgrade of the server or migration to another, more efficient one.

An important and unique feature of PowerVM is the ability to assign a "physical" processor core to the LPAR. The LPAR in this case works in Dedicated mode. The hypervisor determines which physical processor core (or cores) is to be assigned. The administrator has limited influence on this process. The hypervisor decides in such a way as to optimize the LPAR's performance and eliminate potential problems, for example, the large distance between the processor and the memory (so-called *affinity*).

It is worth mentioning the processor definition used by IBM. In the case of POWER servers, the term "processor" means something other than in the x86 world. So, following this line of reasoning, you can say that one Intel I7 processor has 4, 6, or 8 cores. In the case of the POWER platform, when you say "processor," you are referring to the processor core, unless it is explicitly stated that it is a socket or chip (there can be many chips and many processors in one socket). It is important to always keep this fact in mind when reading the IBM documentation.

Parameters of virtual processors

An LPAR has many parameters that you can define during its creation and then change later during its operation. These parameters have a significant impact on the way the system works as well as on the allocation of resources to the LPARs. There are two modes of LPAR operation: *Dedicated* and *Shared*. The screen for selecting the mode that is visible during the creation of an LPAR is shown in Figure 3-4.

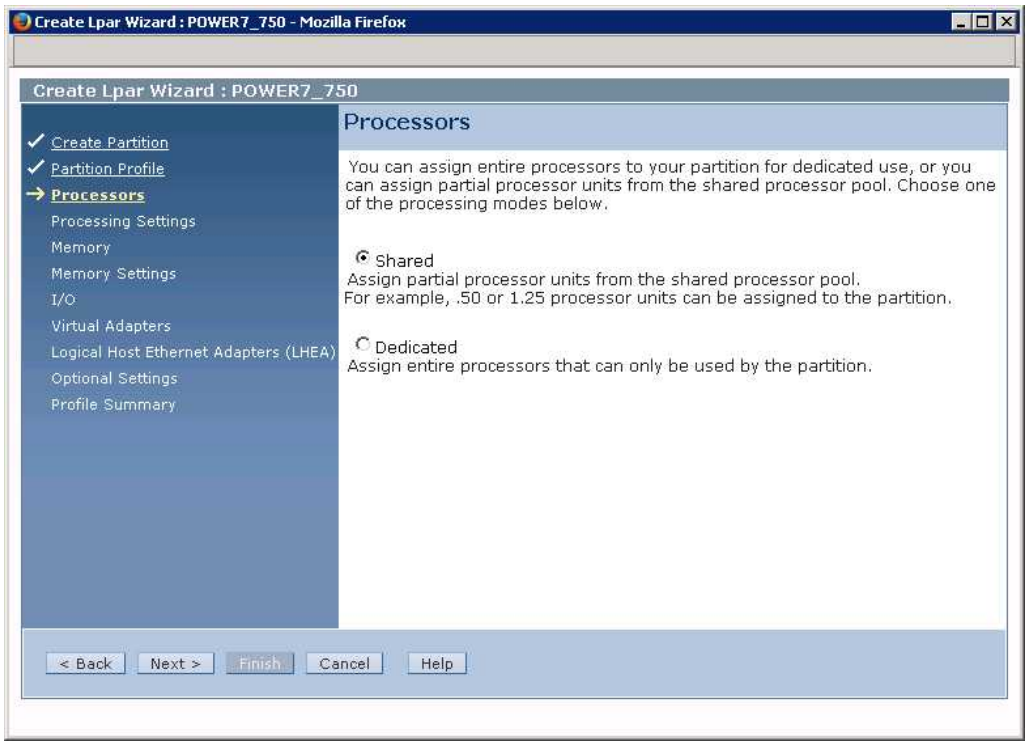

Figure 3-4: Choice of operation mode for an LPAR: Dedicated/Shared.

Dedicated mode

Dedicated mode entails allocating entire "physical" processors (cores) to the LPAR. If the LPAR does not use the processor, the unused CPU cycles are lost. From the operating system, it is visible in the form of idle processor cycles (*% idle*). If you do not want to lose the dedicated processor cycles, you can specify its mode of operation as *donating*. This setting means that the unused time of the dedicated processor will be put into use by the other LPARs on the server, unless the donating LPAR is heavily loaded. By default, an LPAR loaded at 80% will not donate the processor time to the pool. You can control this value using the *ded_cpu_donate_thresh* setting that you can change with the *schedo* command from the operating system.

In *Dedicated* mode, you can specify three more detailed operating parameters:

1. ***Minimum processors*** - The minimum number of processors with which the LPAR can boot. If the minimum number of processors are not available when you attempt to boot the LPAR, the LPAR will not start and will instead report an error.
2. ***Desired processors*** - The desired value. An LPAR always starts with the number of processors specified here. The only exception occurs due to a lack of free resources on the server in the "desired" amount when the partition is started. In this case, the LPAR will boot with the highest number of processors available above the minimum value. You can change this value dynamically during the system operation from the minimum to the maximum.
3. ***Maximum processors*** - The maximum number of processors that you can allocate to the

LPAR without a restart. If the partition has two processors (*desired*) visible in the system and four processors set as the maximum value, the administrator can add another two processors from the management console. This way, he can increase the current partition computing power without a reboot. The whole operation takes about 30 seconds. However, if it is necessary to increase the LPAR power above the maximum value, you should:

a. Change the "maximum" value in the profile;
b. Stop the LPAR; and
c. Start the LPAR from the changed profile.

Why does this (maximum) setting exist? Is it not possible to apply the rule that each LPAR has a "maximum" that is set to 256 by default (256 being the maximum number of processors (cores) supported by AIX)? Well, the problem is twofold:

1. ***Performance*** - The operating system must be ready to handle the dynamic addition of resources such as a processor or memory. It must have appropriate structures in place to support them. The system builds these structures during the boot. For optimal performance, it is unwise to build structures to support processors and memory if they will never be used. Such an approach would waste resources and have a negative impact on system performance.
2. ***License*** - Some software providers may require the "maximum" to be set to a number corresponding to your software licenses. This makes it difficult to add resources freely and hence breach the license rights. At the moment, the largest software providers do not pay attention to this value. However, the licensing policy is a "policy" in the full sense of the word. It is not subject to logic and can be changed at any time.

The "maximum" parameter should be set in a reasonable way; if possible, you need to provide a reasonable amount of resources to modify existing resources in case of performance problems.

Dedicated mode has the disadvantage of wasting unused resources. The ability to donate unused dedicated processors (donating mode) partially mitigates this disadvantage. This functionality can be enabled at any time, but even in such a case, part of the processor cycles will remain unused. When it comes to the advantages of Dedicated mode, it guarantees constant performance that is not disturbed by the mechanisms of virtualization. You can feel the improvement in performance when operating in this mode, especially if the server is very dynamically used (i.e., many LPARs are created, deleted, and modified). According to the documentation, the difference in performance between a system using dedicated and shared processors is around 7–10%. However, you should consider the fact that the achieved result largely depends on the number of LPARs on the server, its total load, and the correctness of the distribution of the LPARs (affinity).

This mode is mainly used for systems that are characterized by the large and constant use of processor resources. If you want to optimally utilize the entire platform, you should use it in a careful and reasonable fashion (that is, rather rarely).

Shared mode

Shared mode means that virtual processors are assigned to the LPAR. They can use cycles of different processors from the available processor pool, although the maximum virtual processor computing power will never exceed the power of the physical processor core.

This mode offers you the most benefits in the form of the good utilization of the whole server. It provides a lot of parameters that you can use to control its work as well as the allocation of resources. The parameters you can modify are displayed in Figure 3-5.

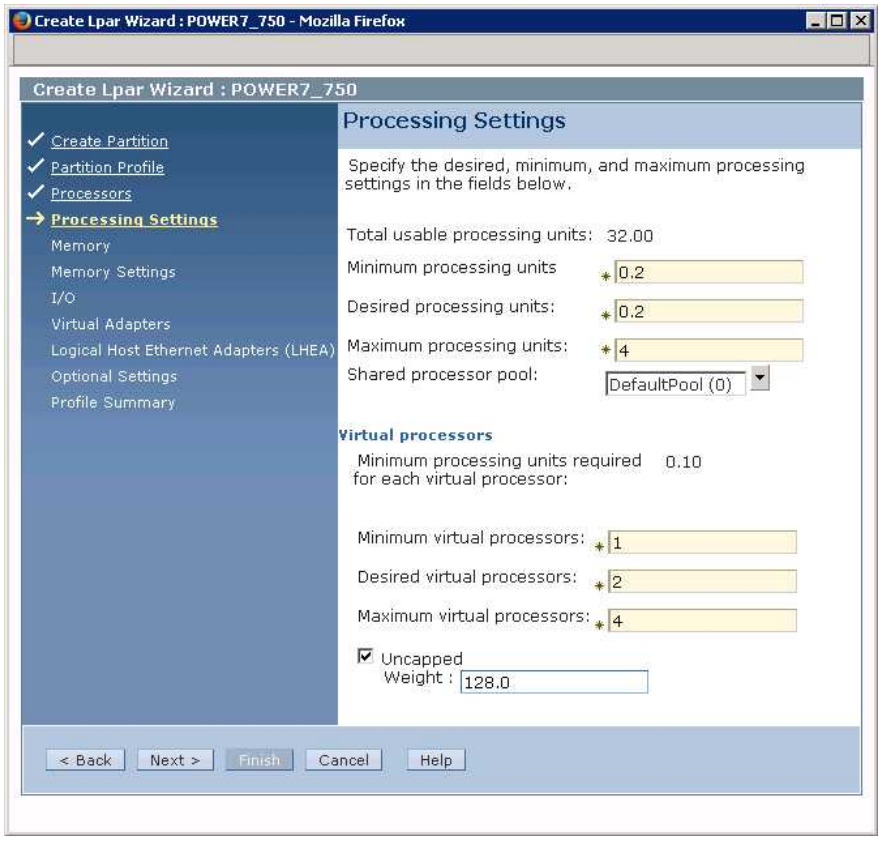

Figure 3-5: The HMC - the parameters of Shared mode.

Description of the parameters:

- ***Minimum virtual processors***, ***Desired virtual processors***, and ***Maximum virtual Processors*** - Parameters play a similar role to that of their equivalents from Dedicated mode, with the difference being that in this case we are talking about virtual processors rather than physical ones.

- ***Minimum processing units*** - The minimum amount of processor computing power that this LPAR can have. If the server does not have this amount of processor power available, the LPAR will not start. However, if you remove the processor's power dynamically, you will not reduce it below this level. The level is definable with an accuracy of 1/100 of the processor's computing power (a minimum of 1/10 CPU per one virtual processor). You can change the parameter, but it requires reloading the LPAR profile, that is, stopping the LPAR and then restarting it.

- ***Desired processing units*** - The LPAR will always boot with the processor computing power specified here. The only exception occurs in the case of a lack of free resources on the server in the *desired* quantity at the time of starting the partition. In this case, the LPAR will boot with the highest available processor power, which will not be less than the minimum value. You can change this value dynamically during system operation from the minimum to the maximum with an accuracy of 1/100 of the processor's power (a minimum of 1/10 CPU per one virtual processor).

  This value is important because the LPAR, even when operating in the *uncapped* mode, will never receive less power than the "Desired processing units." This is a guarantee of computing power for the partition. Yet, you should not think that it is permanently assigned to the LPAR. You should understand it as a guarantee that the LPAR will get 100% of the power if it has such a demand. However, if the LPAR does not need this computing power, the hypervisor will donate it to the pool of free resources and assign it to another LPAR that requires it.

- ***Maximum processing units*** - The maximum processor power that you can add to an LPAR without restarting the operating system. If it is necessary to increase the partition computing power above the maximum value, change the maximum value in the profile, close the LPAR, and then start the LPAR from the changed profile with the new maximum value. You can change this value to an accuracy of 1/100 of the processor's power (a minimum of 1/10 CPU per one virtual processor).

- ***Uncapped/Capped*** - Unless you select the *uncapped* mode, an LPAR works in *capped* mode. The differences are best illustrated using the example of two LPARs. Suppose you have two partitions created in the shared mode with the parameters as shown in Table 3-3.

Table 3-3: Examples of LPARs - capped/uncapped.

	Virtual Processors			Processing Units			
	Min.	**Desired**	**Max.**	**Min.**	**Desired**	**Max.**	**Cap/Uncap**
LPAR1	1	2	4	0.1	0.5	4	*Capped*
LPAR2	1	2	4	0.1	0.5	4	*Uncapped*

Both LPARs behave similarly. You can dynamically (without rebooting) add or remove processor resources. In both cases, changing the minimum and maximum parameters requires reloading the profile (stopping the LPAR, changing the profile, and starting the partition from the changed profile). However, there is a key difference:

- LPAR1, which works in the *capped* mode and has the value of "*Desired processing units*" set to 0.5, will never use more than 0.5 CPU processing power, unless you explicitly change its mode to uncapped or explicitly add CPU power.
- LPAR2, which operates in the *uncapped* mode and has the value of "*Desired processing units*" set to 0.5, will be able to use the power corresponding to the "*Desired virtual processors*," namely 2 CPU. LPAR2 will be able to use a processor power above 0.5, but only if the server has the available CPU capacity and the *shared processor pool*, which LPAR belongs to, has unallocated processor resources.

You may question whether it is better to create all LPARs in the uncapped mode, so that each LPAR will be able to increase its computing power in case of unusual and difficult to predict

demand. The answer is usually affirmative. This is a key mechanism for optimizing the use of the server. The only important factor that is contrary to the use of the uncapped mode is the aspect of the software license that works there. In the case of LPAR1, a license for 0.5 CPU is sufficient (if the software provider supports sub-capacity licensing). However, in the case of LPAR2, the license requirement for the software used is 2 CPU, since that is the maximum CPU utilization that LPAR2 can achieve without configuration changes.

It is worth noting that the LPAR1 configuration, which is used to show the mechanism of action, is a non-best practice setting in the context of performance. In the case of the capped mode, you should use the minimum number of "*desired*" processors that are sufficient to support the allocated capacity ("*Desired processing units*").

- ***Weight*** - Priority. The priority is a non-significant value, as long as the server utilization does not reach 100%. Up to that point, every LPAR that works in the uncapped mode uses as much power as it needs. When the server uses 100% of the processor's resources, the hypervisor algorithms become "visible." They cause the CPU power allocation to be proportional to the LPAR's set weight. Prioritizing, along with the number of virtual processors and the desired values, is the most important way to influence the CPU power distribution when the server load reaches 100%.

  The *Weight* parameter should be used wisely. For example, distancing the priority of test and production systems or distancing the systems in terms of their criticality and the possible costs of failure to meet deadlines.

Shared processor pools

A ***shared processor pool*** is a virtual entity that defines a set of processors. To be more precise, it specifies the processing power expressed in the number of processors, since it is not related to specific physical processors. Each LPAR belongs to a processor pool. If it is irrelevant to which pool it should belong to, it is created in the default pool, which determines the entire server resources.

The processor pool allows you to optimize the use of the license. Many software vendors, especially the biggest ones, include processor pools in their product licensing models. The following example best explains the operation of the processor pool. Let's assume that we have a server with 32 cores (in the world of POWER servers, one processor = one core). On this server, we want to run four LPARs, each with four processors, with a DB2 database installed, but we only have licenses for eight cores. The described example is illustrated in Table 3-4:

Table 3-4: Example LPARs in the shared processor pool.

	Virtual Processors			Processing Units					
	Min.	**Desired**	**Max.**	**Min.**	**Desired**	**Max.**	**Cap/Uncap**	**Weight**	**Pool**
LPAR1	1	*4*	4	0.1	*1*	4	*Uncapped*	64	***8 CPU***
LPAR2	1	*4*	4	0.1	*1*	4	*Uncapped*	64	
LPAR3	1	*4*	4	0.1	*1*	4	*Uncapped*	128	
LPAR4	1	*4*	4	0.1	*1*	4	*Uncapped*	128	

Without the use of a processor pool for this configuration, you would need 16 processors. Eight are needed when you use the pool. In the case of a configuration such as that in the table above, the following rules apply to the listed LPARs:

- The total load of all the LPARs will never exceed eight processors.
- Each LPAR can use up to four processors (*Desired virtual processors*) at any given time, provided the processor computing power is available on the server and the utilization of the processor pool has not reached eight processors.
- Regardless of how busy the server is, the LPAR is guaranteed the power of one processor (*Desired processing units*).
- If all the LPARs show the maximum demand for processing power at the same time and the server will be 100% loaded, the *Weight* (priority) parameter will start to work. This means that the hypervisor will divide the power between the LPARs in proportion to the priority (with the accuracy of the hypervisor algorithms). The expected effect will be as follows:
    - LPAR1 = 1.67 (+0.67 CPU due to *weight = 64*).
    - LPAR2 = 1.67 (+0.67 CPU due to *weight = 64*).
    - LPAR3 = 2.33 (+1.33 CPUs due to *weight = 128*).
    - LPAR4 = 2.33 (+1.33 CPUs due to *weight = 128*).

Memory virtualization

As with the processor, the hypervisor manages memory virtualization. The hypervisor is an integral part of the server, which is delivered with its firmware.

From the server's point of view, memory is allocated to each LPAR in Dedicated mode. Thus, typically an LPAR does not share this memory with other LPARs. The memory allocation is made using *Logical Memory Blocks* (LMB) defined at the point of server initialization. The typical size of a block is 256 MB. This is the size that you use when you add memory to an LPAR or reduce it. The exception is the *Active Memory Sharing* (AMS). In this mode, the memory is allocated with blocks of 4 KB, and it can be shared between multiple LPARs. The AMS is described later in this chapter.

The memory allocated to a specific LPAR in Dedicated mode may not be continuous from the server's point of view. However, from the LPAR's point of view, its memory is continuous. In order to achieve this status, memory addresses are virtualized for the LPAR. This process is shown in Figure 3-6.

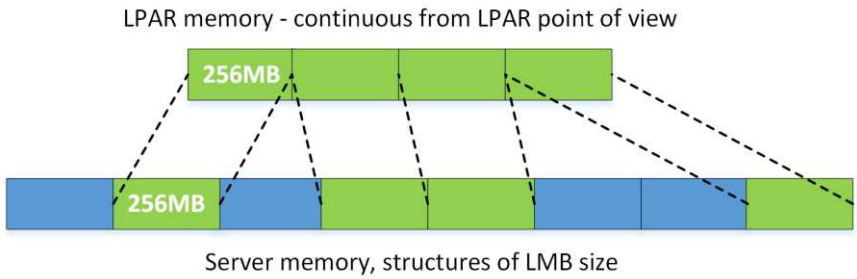

Figure 3-6: Virtualized LPAR memory.

When creating an LPAR, the hypervisor tries to allocate the memory in a uniform area, which also corresponds to the perspective of the physical location. Nevertheless, many operations that change the amount of LPAR memory (add or reduce) performed on multiple LPARs on the server can lead to a lack of uniformity. This is not a problem until the memory is "near" the allocated processor in multiprocessor solutions.

Another issue in the context of memory virtualization concerns the memory of the hypervisor itself. The hypervisor uses some of the memory that was not shared with the LPARs themselves. In some server models, it is possible to increase the availability of the solution by mirroring the hypervisor's memory. This feature is known as *Active Memory Mirroring* or *Hypervisor Memory Mirroring*. Please note that the hypervisor memory mirroring does not apply to the Virtual I/O Server, which will be discussed later in the chapter. However, you can also increase the availability of I/O operations by creating more Virtual I/O Servers that support the same network and disk traffic.

Basic parameters of the LPAR memory

The basic parameters of an LPAR's dedicated memory, as set during its creation, are shown in Figure 3-7.

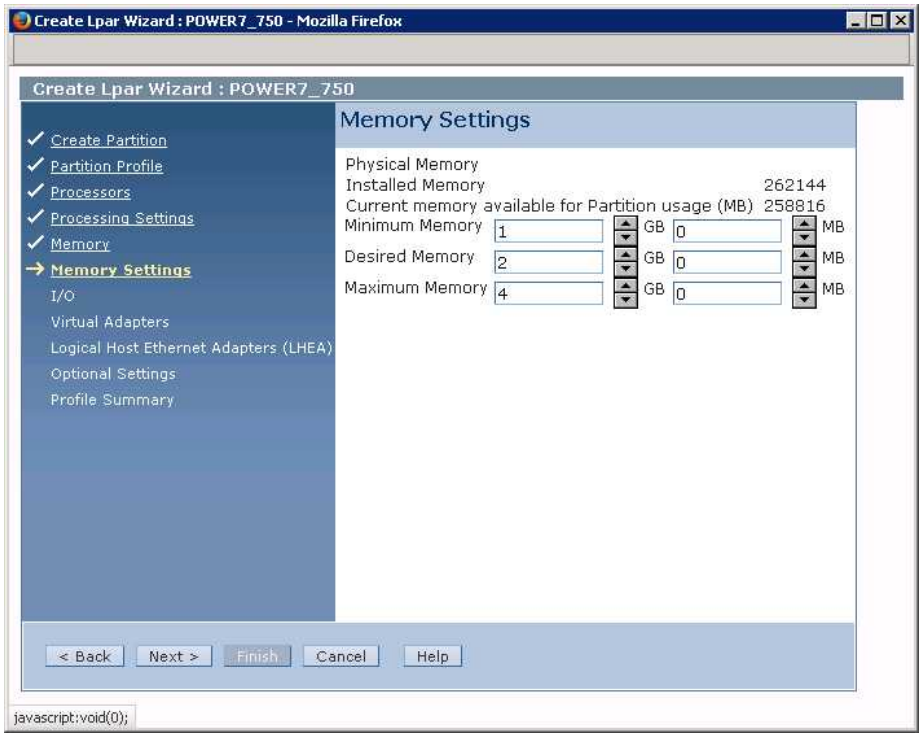

Figure 3-7: Basic memory parameters - HMC.

Minimum Memory - The minimum amount of memory that this LPAR can have. If the server does not have such an amount of available memory, the LPAR will not start. However, if you are removing

the memory dynamically from the LPAR, you will not reduce it below this level.

Desired Memory - The desired memory size for the LPAR. The LPAR will start with this amount of memory, so long as the server has enough free RAM.

Maximum Memory - The maximum amount of memory that you can assign to a partition without having to restart it. The value of this parameter should not be defined with too much excess, since in such a case we lose real memory in two places:

1. From the perspective of the entire server, you can see that when you increase the *Maximum* parameter, the amount of memory used by the hypervisor increases. Thus, the amount of memory available to the LPARs decreases (you lose about 1/64 of the memory defined as the *Maximum Memory*).
2. From the perspective of the LPAR, the available memory is occupied by the structures used to support the *Maximum Memory* size. These structures are created in the memory during the LPAR boot process.

Active Memory Expansion (AME)

The functionality of the *Active Memory Expansion* feature is based on memory compression. Although the functionality works at the level of the operating system, it is defined at the level of the management console, and it requires the appropriate license.

AME was introduced along with the POWER7 processor family. Running it on POWER7+ or newer processors works much more efficiently, since those processors support hardware compression and decompression. This feature allows you to create more LPARs or LPARs with more memory, while having limited physical memory. Of course, this is achieved at the cost of the processor, since it must perform additional work. However, with appropriate compression parameters, the overheads can be insignificant and range from only a few to a dozen or so percent.

When you create an LPAR with *Active Memory Expansion*, you must set two parameters:

Active Memory Expansion - Enable/disable the feature. Enabling or disabling memory compression requires restarting the LPAR.

Active Memory Expansion Factor - The compression factor of the memory of the LPAR, which is within the range of 1.00–10.00. Changing the factor, when AME is switched on, is done dynamically. The factor determines how much memory is visible to the operating system. To obtain this information, multiply the factor by the amount of memory allocated to the LPAR by the management console (*Desired memory*). Examples of LPARs with different compression ratios are shown in Table 3-5.

Table 3-5: AME - The amount of memory when using different compression factors.

	Desired Memory (GB)	AME	AME - Compression Factor	Memory Seen by the Operating System (GB)
LPAR1	*4*	ON	1.0	*4·1 = 4*
LPAR2	*4*	ON	1.5	*4·1.5 = 6*
LPAR3	*4*	ON	1.8	*4·1.8 = 7.2*

LPAR4	*4*	ON	3.0	*4·3 = 12*

Applications that are being run on the operating system are not aware of the memory compression mechanism, so its use is completely transparent to them. Of course, this assumption is true in the case of the reasonable selection of the compression factor. If the application uses all the available memory, the high ratio setting can have dramatic implications for performance. In such a case, although the application does not know that the system uses memory compression, it will slow down its performance by waiting for the processor cycles used for continuous compression and decompression.

When an operating system works with memory compression, it divides the available memory into two pools. For a good illustration of the solution, let's assume that an LPAR with 10 GB of memory has AME enabled with a compression factor of 2.0. This case is presented in Figure 3-8.

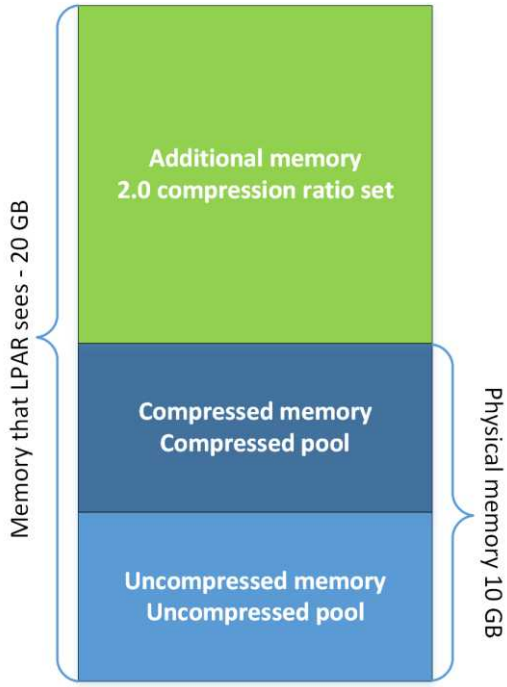

Figure 3-8: Memory compression.

Uncompressed Pool - Uncompressed memory, on which the operating system operates in a traditional way. Its size is variable over time. It depends on the compression ratio as well as the amount of memory used by the operating system.

Compressed Pool - Compressed memory. Its size, as in the case of the Uncompressed Pool, is variable over time, and it depends on the amount of compressed data.

The operating system, even with a very high compression ratio, may not use the *Compressed Pool*, so long as the currently used memory fits within the *Uncompressed Pool*.

If the *Uncompressed Pool* is unable to fit the utilized memory, AME compresses the least-used memory pages and moves them to the *Compressed Pool*, thereby freeing the memory from the Uncompressed

Pool.

If the application refers to a memory page that is in a compressed state, a *page fault* occurs. As a result, the memory page is decompressed and moved to the *Uncompressed Pool*. Then, the application performs the desired operations without being aware of the operation of decompression and the transfer of memory by the operating system.

Some facts about AME that you should pay attention to:

- AME does not compress *pinned* memory (i.e., memory that can never be transferred to a paging space or compressed by AME).
- AME does not compress the file system cache. It is hence good practice to limit the maximum size of the file cache when using AME (parameter *maxclient%* of the *vmo* command).
- Systems that support applications that work on compressed or encrypted data will not benefit from memory compression.
- AME only compresses 4 KB pages.
- The use of compression requires additional processor cycles, although with a reasonable configuration the load is very low.
- If you use too high a compression factor, you may experience a *memory deficit*. This means that the amount of memory that should be visible to the operating system cannot be achieved because the compression ratio is too low or too much data are not compressible. In this case, structures that do not fit into the memory are paged out to a paging space, which can significantly slow down the operating system. An example of a memory deficit is presented in Figure 3-9.

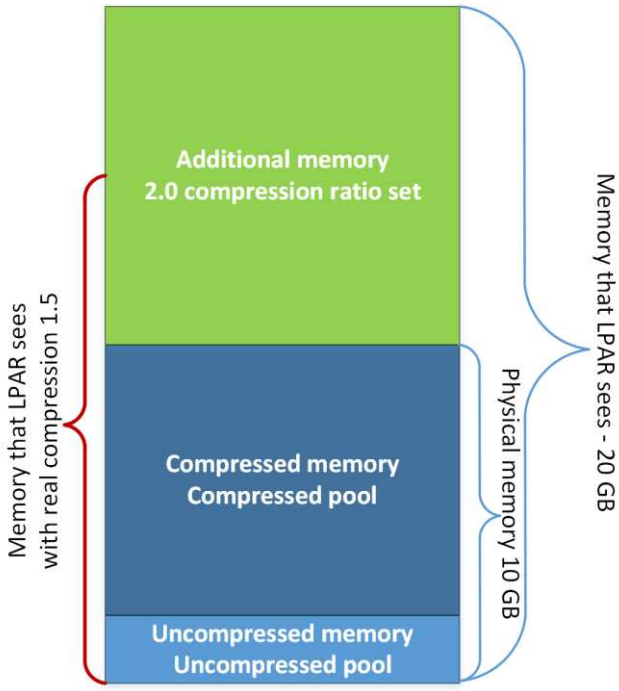

Figure 3-9: Memory compression - Memory deficit.

Despite the limitations mentioned above, most systems can successfully use this mechanism while minimizing the CPU power loss. This is easy to verify using the *amepat* tool, which is available on AIX regardless of the licenses you have.

Working with Active Memory Expansion

The launch of AME brings potential threats in the form of performance impacts as well as questions about what compression ratio will be most appropriate for the system. The solution to both those issues is the *amepat* tool. It is available from the operating system regardless of whether our server has licenses for this feature or not. The tool can perform the necessary analysis and present several system configuration proposals. When equipped with these suggestions and information about how they will affect the system load, you can choose the option that is most interesting in your case. An example of the use of *amepat* follows:

```
# amepat 1 9   # Launch of nine one-minute samples.
# All unnecessary information has been omitted.
Active Memory Expansion Modeled Statistics         :
-------------------------------------------
Modeled Expanded Memory Size :   4.00 GB
Achievable Compression ratio   :2.42   # The average compression ratio of the LPAR memory.

Expansion    Modeled True      Modeled             CPU Usage
Factor       Memory Size       Memory Gain         Estimate
---------    -------------     ------------------  -----------
     1.00          4.00 GB         0.00 KB [  0%]  0.00 [  0%]
     1.07          3.75 GB       256.00 MB [  7%]  0.00 [  0%]
     1.15          3.50 GB       512.00 MB [ 14%]  0.00 [  0%]
     1.24          3.25 GB       768.00 MB [ 23%]  0.00 [  0%]
     1.34          3.00 GB         1.00 GB [ 33%]  0.00 [  0%]
     1.46          2.75 GB         1.25 GB [ 45%]  0.00 [  0%]
     1.60          2.50 GB         1.50 GB [ 60%]  0.00 [  0%]

Active Memory Expansion Recommendation:
---------------------------------------
The recommended AME configuration for this workload is to configure the LPAR
with a memory size of 2.50 GB and to configure a memory expansion factor
of 1.60.  This will result in a memory gain of 60%. With this
configuration, the estimated CPU usage due to AME is approximately 0.00
physical processors, and the estimated overall peak CPU resource required for
the LPAR is 0.80 physical processors.
# All unnecessary information has been omitted.
```

As you can see from the example above, the result of the running analysis provides you with a few settings suggestions. The last one suggests using a compression ratio of 1.6 and limiting the physical memory from 4 GB to 2.5 GB. With these settings, the additional processor usage for compression will still be 0%. During the test, the LPAR used a small amount of the available memory, and its utilization did not exceed 2.5 GB. So, compression, although enabled, would not cause a loss of processor cycles.

In this case, the analysis showed that the LPAR has more memory than it needs, so the proposals are limited to reducing the physical memory and turning on the compression. As a result, memory will be released, and it can then be used for other purposes. However, from the point of view of the working applications, nothing will change.

The key element is choosing the right time at which to carry out the analysis. You should carry it out

when you are expecting a greater utilization of your system. It is good to run the analysis for a longer period, so that it will be as reliable as possible. Improperly chosen settings may negatively affect system performance.

Although memory compression is transparent for the application, you can easily read its settings at the operating system level. The command that gives you the correct information is *lparstat*:

```
# lparstat -i       # All unnecessary information has been omitted.
Online Memory                              : 6144 MB
Memory Mode                                : Dedicated-Expanded
Target Memory Expansion Factor             : 1.50
Target Memory Expansion Size               : 9216 MB
```

The command shows that the system has 6144 MB of physical memory. It has compression enabled and the factor is set to 1.5, which gives the system a total of 9216 MB of memory.

You can use tools such as *vmstat -c*, *lparstat -c*, or *topas* to monitor AME.

Active Memory Sharing (AMS)

The previous section described the *Active Memory Expansion* mechanism. This mechanism works at the LPAR level, and it allows the LPAR to use more memory than it actually has.

The Active Memory Sharing mechanism described in this section allows you to achieve a similar effect, that is, to use more memory than the server has. The main difference is that AMS works at the whole server level, while AME works at the LPAR level.

A diagram of the AMS operation is presented in Figure 3-10:

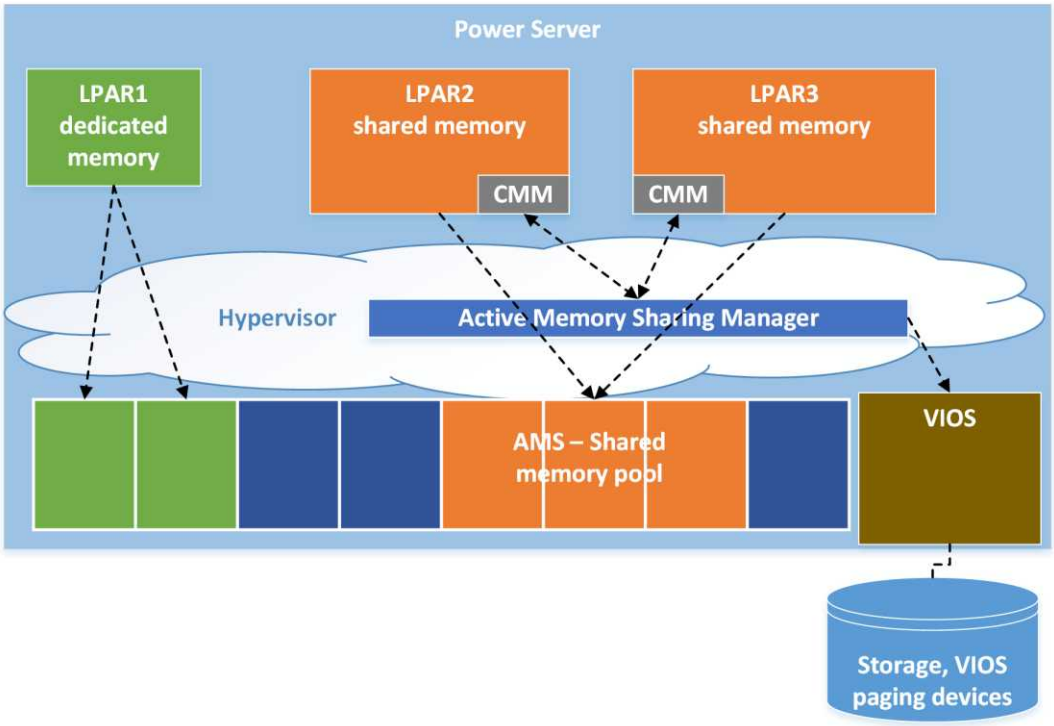

Figure 3-10: AMS - Operating scheme.

AMS is more complicated than AME. It requires more preparation to run. For the operation of Active Memory Sharing, it is necessary to separate a part of the memory into a special AMS shared memory pool structure. It is also necessary to have a Virtual I/O Server with dedicated logical volumes or disks intended for the so-called paging devices.

The structures shown in the figure above:

- ***AMS Shared Memory Pool*** - A separate memory pool, which is shared by a group of partitions. Only one such pool can exist on one server. The LPARs either fully use the shared memory or fully use the dedicated memory. It is not possible to mix different types of memory within one partition.

  The hypervisor can dynamically allocate and remove memory from the LPAR. This means that the same memory areas can be assigned to one LPAR at one time and then to another LPAR at another time. With this mechanism, the partition does not have unused memory, since such memory is returned to the pool. In the case of new demand, it is allocated to a given LPAR. Operations of this type work very effectively in terms of the memory used, since they operate with a block of 4 KB. This is a typical block size for the operating system.

- ***VIOS Paging Devices*** - The declared memory used by the LPARs running in the *AMS shared memory pool* may exceed the total size of this pool (*over-commitment*). If this type of pool is used, it is a desirable outcome, since it allows effective memory usage. Nevertheless, there may be a

situation in which many LPARs simultaneously report a memory demand that will be greater than the amount of memory available in the pool. The solution to this problem is the use of special paging devices operating on the designated VIOS (one or two for high availability). Unfortunately, the described case leads to the exchange of memory pages with the paging devices. This is a good defense mechanism, but it can lead to a significant slowdown of the LPARs affected by this process. It should be noted that for each LPAR, one dedicated paging device is assigned. Therefore, for the operation of the mechanism, it is necessary to have the same number of devices as the number of LPARs that work in the memory pool.

As mentioned previously, this mechanism is quite complex. It requires cooperation between several elements:

- ***Active Memory Sharing Manager (AMSM)*** – The code located in the hypervisor, which is the heart of the mechanism. It manages the pool of memory, allocates and removes memory from the LPARs, and works with the VIOS, ordering it to write or read memory pages from a paging device. Receiving memory pages from the LPARs is done in two ways.
  - ***Loaning*** - In this case, the operating system itself designates memory pages that it does not need. It gives back these pages voluntarily - lending them. The memory pages obtained in this way should not adversely affect system performance.
  - ***Stealing*** - In this case, the AMSM steals the pages in a "forceful" manner, having previously saved their contents via the VIOS on the appropriate paging device. The hypervisor always steals the least-used pages. The kernel of the operating system, as the object of action, provides the hypervisor with statistics regarding the use of memory pages.
- ***Collaboration Memory Manager (CMM)*** – The functionality of the operating system kernel. Its role is to work with the AMSM. It provides information about the weight of individual pages of memory. It also designates pages to donate and provides statistics on the use of pages. These statistics are used by the hypervisor to "steal" relevant pages.

The memory sharing process would not be complete without the deduplication mechanism. The mechanism (***Active Memory Deduplication - AMD***) is also available here. AMD can be enabled or disabled for all the LPARs using AMS. It is not possible to enable the mechanism for individual LPARs.

Deduplication significantly helps to reduce the amount of memory needed, and it thus provide a large memory *over-commitment*. You can imagine a simple example in which you run four LPARs with the same version of the operating system and a similar application running on the same software. The memory content of each LPAR is the same in many places because the same application codes are loaded. Thus, the deduplication mechanism does not have to store the same information four times for each individual LPAR, but can in fact store this information only once, "mapping" it to the appropriate memory address of each LPAR. This situation is presented in Figure 3-11.

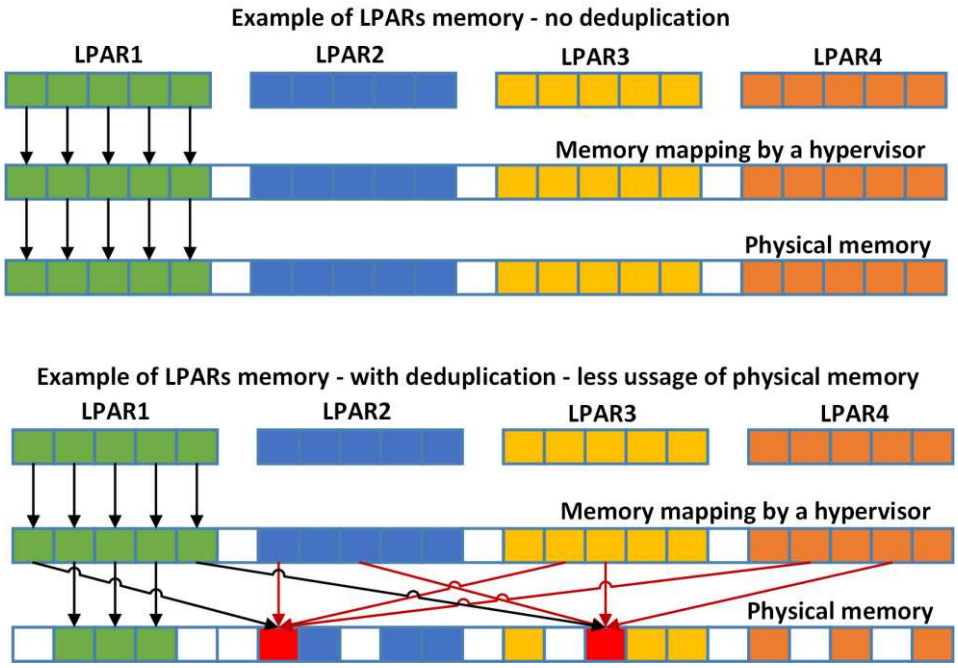

Figure 3-11: AMS - Deduplication.

The deduplication mechanism does not work immediately. You cannot expect that when several identical LPARs are started at the same time, the global memory utilization will be decreased at once. This mechanism works in the background. The hypervisor builds a memory map by calculating the signatures of each page. The hypervisor can search this map at a time when it has more free computing resources, and it combines the pages by performing the deduplication process. If any of the LPARs using the same page modify it, then the COW (*Copy on Write*) process takes place. The memory page is copied to a new location and mapped in place of the old LPAR memory page. Only then is its content modified.

Like any such mechanism, AMS has several limitations that you should be aware of:

- LPARs that use this feature must use the processors in shared mode (dedicated mode is not allowed).
- The LPARs can only have virtual devices (virtual SCSI, virtual Fiber Channel - NPIV, virtual serial, virtual Ethernet). Physical devices are not allowed.
- The operating system cannot use blocks other than 4 KB, which may have some negative performance implications for applications that use 64 KB pages on a massive scale. In the case of AMS, the applications must use the page of 4 KB, so their support may require more processor cycles than in the case of 64 KB.

Detailed information on *Active Memory Sharing* is available in the publication *IBM PowerVM Virtualization Active Memory Sharing*, which is available at *https://www.redbooks.ibm.com*.

Virtual I/O Server (VIOS) - I/O virtualization

The Virtual I/O Server is part of the POWER platform virtualization. It is responsible for the virtualization of I/O. The division of virtualization into separate components that virtualize the processor and the memory as well as separate components that virtualize the I/O operations may appear surprising, especially for people who deal with virtualization on the x86 platform. Nevertheless, it makes considerable sense in the form of:

- A stable, internal world of CPU and RAM virtualization that can be supported by a simple hypervisor code built into the server.
- A complex, variable world of I/O operations that can be supported by dedicated virtualization software - Virtual I/O Server.

The VIOS, from the point of view of the AIX administrator, is just an AIX system following significant customization in terms of cooperation with the hypervisor and the optimal support of I/O operations. When using the Virtual I/O Server, you usually work in a restricted shell with limited privileges and access to specially prepared commands. These commands are mostly aliases of AIX commands. By running the *oem_setup_env* command, you become the root of the VIOS and have access to traditional AIX commands.

The Virtual I/O Server is an LPAR to which you assign physical devices (disks, LAN adapters, and SAN adapters) in order to virtualize them and make them available to other LPARs on the server. There may be many VIOS on a single server. It is possible to create dedicated VIOSs for specific systems, or to create dedicated VIOSs for disk traffic and others to handle network traffic. Usually, two typical configurations are used with one and two VIOSs. These two typical configurations as well as one that is less typical with many VIOSs are shown in the following figures.

- **Server with one VIOS** - A typical configuration for small servers and test systems. A single VIOS does not limit virtualization in any way. On one VIOS, you can use multiple paths to access disks and LANs. This is very helpful in the event of the failure of individual adapters. Nevertheless, this configuration does not allow you to fully use the potential of the platform in terms of maximizing its availability.

  In the case of one VIOS, its failure results in the unavailability of all the LPARs that use it. Additionally, each time it is restarted (e.g., due to patching), the LPARs that use it should be shut down or migrated to another server, for example, via *Live Partition Mobility* mechanisms.

The virtualization scheme of one server with one VIOS is shown in Figure 3-12.

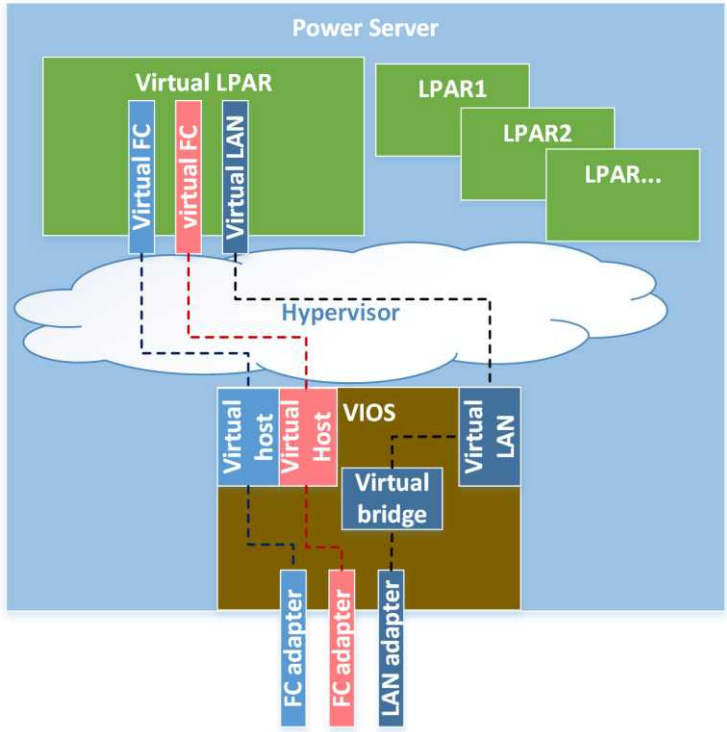

Figure 3-12: Server with one VIOS.

- **Server with two VIOSs** - A typical installation that allows full redundancy. Both VIOSs simultaneously work for the LPARs. The failure of one of them or a planned restart does not result in the unavailability of the partitions on the server, since the paths through the second VIOS are preserved.

  Having two VIOSs increases the availability of systems and, at the same time, allows the systematic patching of the VIOSs in alternate mode.

The virtualization scheme on one server with two VIOS is shown in Figure 3-13.

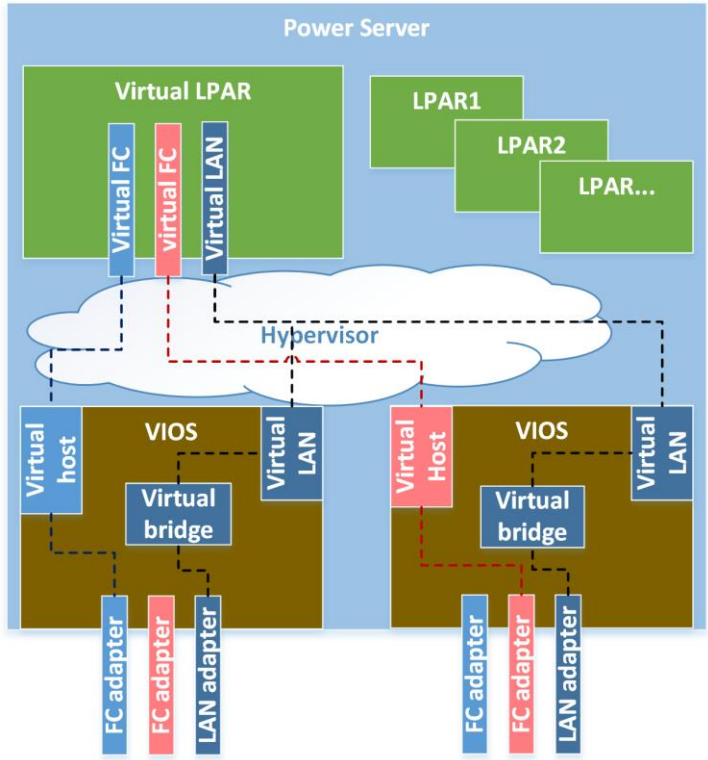

Figure 3-13: Server with two VIOS partitions.

- **Server with four VIOSs** - Figure 3-14 presents an example of having both the production and test systems on one server. There are two dedicated VIOSs for the production systems and two for the test systems. In a broader context, it shows the wide use of VIOSs and the many scenarios in which they can potentially be used.

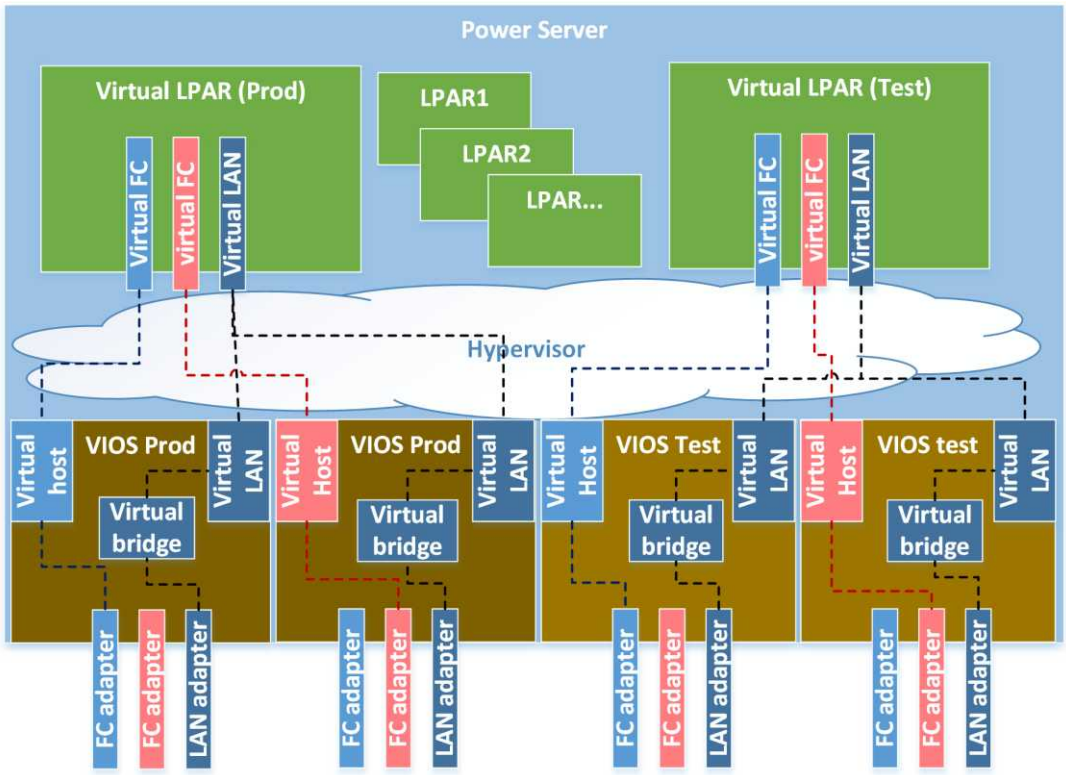

Figure 3-14: Server with four VIOS partitions.

Planning VIOS installations

The Virtual I/O Server installation process itself is much simpler than the installation of the AIX operating system. As the VIOS is a highly specialized system, you do not have too many options during the installation process. However, it is extremely important to properly plan the configuration and determine the physical components on which the installation should take place.

The VIOS can be installed on internal disks connected with the SAS (Serial Attached SCSI) bus as well as disks from the external disk array. Typically, internal drives are selected for the installation. There are several reasons for choosing internal disks for VIOS installations:

- **Isolation of problems with disk arrays and the SAN/LAN network** – Let's imagine a situation in which disk arrays are damaged and we lose access to the operating systems and the VIOS at the same time. Such an event makes diagnostics more difficult and extends the time needed to recover the systems.
- **Restoring the VIOS after disk failure** - Imagine a situation in which the VIOS crashes because of an administrator error. It is necessary to restore the VIOS from the backup. The problem appears when you are selecting the disk on which you need to restore the system. In the case of an internal disk, it is obvious, since it is most likely the only internal bootable disk. If you use a disk from an external storage subsystem, you can select the wrong disks, which were mapped through the VIOS to the LPARs. The restore interface makes it more difficult to

select the appropriate disk, providing only limited information about the disks to which you can restore the system.

For installations with two VIOSs that must operate with HA (High Availability) to each other, you should ensure that the internal disks where the VIOS are installed do not share common parts. They must not be connected through the same disk controller, and in the case of large servers, they must not be located in the same module (CEC - Central Electronics Complex).

An important aspect, especially for multi-VIOS installations, is the selection of the components (physical adapters) that will be virtualized. The configurations to be considered in the case of modular servers are shown in the example given below.

Suppose you have an IBM Power System 870 server. It is a modular server. In this example, we assume you have a server consisting of two modules, without any external I/O components. In this case, the two VIOSs should be in the HA configuration with each other. Here's an example of how you can plan the installation and allocation of the components in order to ensure the highest availability of the server.

Figure 3-15 illustrates some rules that are worth following. Below is a description of those rules. It explains the reasons for their use.

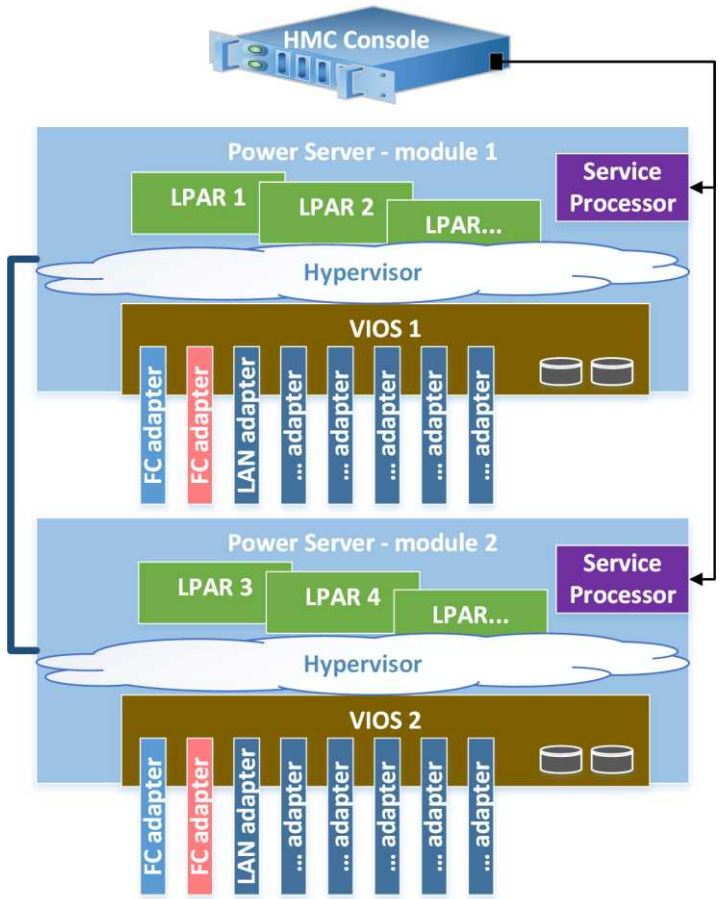

Figure 3-15: VIOS partitions on modular servers.

- **System drives in separate modules** - Each Virtual I/O Server is physically installed in a different server module. Its disks are mirrored. Whenever possible, you can try to make each disk reside on a separate controller.
- **External adapters in the same module as the system disks** – All the external adapters managed by the VIOS are in the same module as the VIOS system disks. It is advisable that the Virtual I/O Server should have a dedicated, high-speed network port when using the *Live Partition Mobility* feature. In this way, the migration processes will be able to proceed quickly and without affecting the network traffic of other systems. Prior to the installation, read the server documentation and place the adapters in the appropriate slots in a way that does not limit their performance.
- **Assign all the VIOS components to a single module** - The following examples explain why this is so important:
    - **There was a "big" failure of one of the modules** - In this case, "big" implies the unavailability of the entire module. As the server consists of two modules, one of them can continue to work. If there were two well-placed VIOSs in the HA configuration, there is a good chance that one of them would keep working (when creating the VIOSs, the hypervisor should take care to allocate memory and processor resources as close to the physical components as possible). It is likely that around half of the LPARs could also survive this failure. Even if the failure caused all the systems to shut down, one Virtual I/O Server and half of the LPARs can be started, while one server module has failed.
    - **There was a "small" failure in one of the modules** - In this case, "small" means the failure, for example, of a processor or memory, which is transparent for the LPARs. The POWER platform has a number of RAS solutions (*Reliability, Availability, and Serviceability*) that allow for the early detection of failures or repeating instructions on another processor in case one of them fails. A small failure, although it did not affect the availability of the systems, may require a service action resulting in the disabling of the module. In the case of a proper configuration, you can attempt to do this without stopping the server, while stopping one of the VIOSs and some LPARs that will not fit on one module.

VIOS installation

Before installing the Virtual I/O Server, please ensure that you have the appropriate licenses. Many POWER server features are purchased and activated separately, including virtualization. You can check which licenses your server has at *https://www-912.ibm.com/pod/pod* after entering the server data. The codes obtained in this way can be used to activate features on the server (if they were not activated earlier). You can verify whether the server features are active from the server properties listed on the HMC. The screen showing the server properties is portrayed in Figure 3-16.

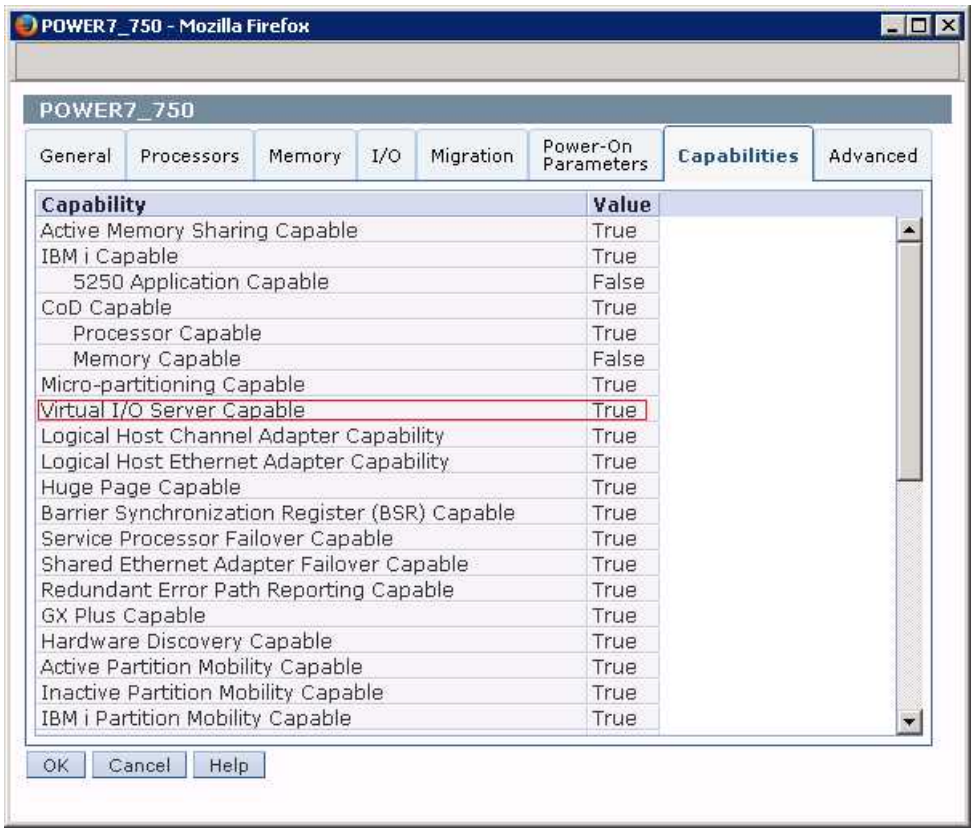

Figure 3-16: HMC - Capabilities, VIOS.

As you can see from Figure, the Virtual I/O Server feature is activated. The subsequent installation steps can be described in several bullet points:

- **Creating an LPAR** - Create an LPAR, indicating that it will act as a VIOS. This is like creating a client LPAR with the AIX system. The difference is that the VIOS must be installed on physical components. When creating a partition, you should pay special attention to the allocation of appropriate physical devices, which should be consistent with the previously created plan. It is a common mistake to set the value too low for the *Maximum virtual adapters*. By default, this value is 10. Leaving it unchanged results in the maximum value being quickly reached and, as a result, leads to the need to restart the VIOS and increase this value.
- **Installation of the VIOS** - You can install the VIOS locally by booting the LPAR from a DVD, either through the HMC (using the NIMOL mechanism - NIM on Linux) or from the *Network Installation Manager* via the LAN. The Network Installation Manager is described in more detail in the chapter on the installation of AIX. The installation process itself does not involve too many options to choose from. The VIOS must comply with strictly defined functions, so it is always a simple installation with only a small number of customization options.
- **Patching** - When installing any software, it is necessary to install the latest patches and constantly check for newer versions. There are installation instructions issued for the VIOS patches, and you should always follow them.
- **Configuration** - This covers many aspects of the system, including the user's configuration,

monitoring, installation of appropriate storage drivers, hardening, configuring a shared network, etc. This topic is extensive. Therefore, it is worth referring to the relevant documentation, although some of the issues will also be discussed later in this chapter. It's still worth considering some initial tips:

- The master user with full rights is known as the *padmin*. He works in a restricted shell. Therefore, he has full configuration rights, but only from the perspective of the commands prepared for him. The commands available to the user can be viewed using the *help* command. They are also visible in the user *.profile* file as aliases:

```
# head -15 /home/padmin/.profile
export PATH=/usr/ios/cli:/usr/ios/utils:/usr/ios/lpm/bin:/usr/ios/oem:/usr/ios/ldw/bin:$HOME

if [ $LOGNAME = "padmin" ]
  then ioscli license -swma
fi

export SHELL=/usr/bin/ksh
alias backup="ioscli backup"
alias restore="ioscli restore"
alias chbdsp="ioscli chbdsp"
alias mvbdsp="ioscli mvbdsp"
alias rmsp="ioscli rmsp"
alias mkrep="ioscli mkrep"
alias chrep="ioscli chrep"
```

- The VIOS is based on the AIX system. You can obtain *root* authorization using the *oem_setup_env* command. After using this command, you can work in the VIOS in the same way as in the AIX operating system. It is often necessary to install storage drivers or monitoring clients.
- Always work on the rights of a dedicated VIOS user using dedicated VIOS commands. The *root* user should only be used when necessary.

Virtualization of storage

One of the main reasons for the existence of the Virtual I/O Server is the virtualization of storage. In the case of the POWER platform, there are many options that you can use for this purpose. You can virtualize storage using:

- VSCSI (Virtual SCSI);
- NPIV (N_Port ID Virtualization); and
- VSCSI and Shared Storage Pool.

A unique feature of this platform is the possibility of not virtualizing a given functionality. You can allocate physical resources to the LPAR, which results in greater bandwidth and performance, but at the expense of lower flexibility.

Further considerations regarding storage will include the key terms related to this issue, which should hence be clarified at the beginning:

- **SCSI - Small Computer System Interface.** This term refers to two elements:
  1. A physical bus that defines the carrier and transmission parameters. In the past, it was

often used in servers. Currently, it has been replaced by other technologies such as Fiber Channel.

2. The SCSI protocol, which defines the way in which devices communicate. To this day, it is a protocol used in communication with storage devices. It is also used by other protocols such as Fiber Channel Protocol, iSCSI, and FCoE.

- **FCP - Fiber Channel Protocol**. The protocol is used in the SAN (Storage Area Network). It is a carrier for the SCSI protocol. With it, servers perform operations on disks presented by storage.
- **LUN - Logical Unit Number**. This name stems from the SCSI terminology. From the server perspective, it is a disk. From the storage perspective, it is a logical structure. It implies that a slice of space of a given size is presented to a given server.
- **SAN - Storage Area Network**. A fast network designed for low latency and high bandwidth. It enables the connection of devices that store data (disk arrays, tape drives, etc.) with servers.
- **SAN Fabric**. A set of connected SAN network devices (switches). When configuring the SAN network within a company, two fabrics are most commonly used. They are physically separated from each other, and each server and each storage device is connected with separate adapters to both fabrics. Such a configuration allows you to build a reliable network.
- **HBA - Host Bus Adapter**. In our case, it is a Fiber Channel adapter connected to the SAN network. It enables the server to access the disks provided by the external storage.

VSCSI (Virtual SCSI)

VSCSI is the oldest form of storage virtualization available through the VIOS. The disks are connected to the LPARs via virtual SCSI channels that connect the Virtual I/O Server with the client LPARs. Such a connection is presented in Figure 3-17.

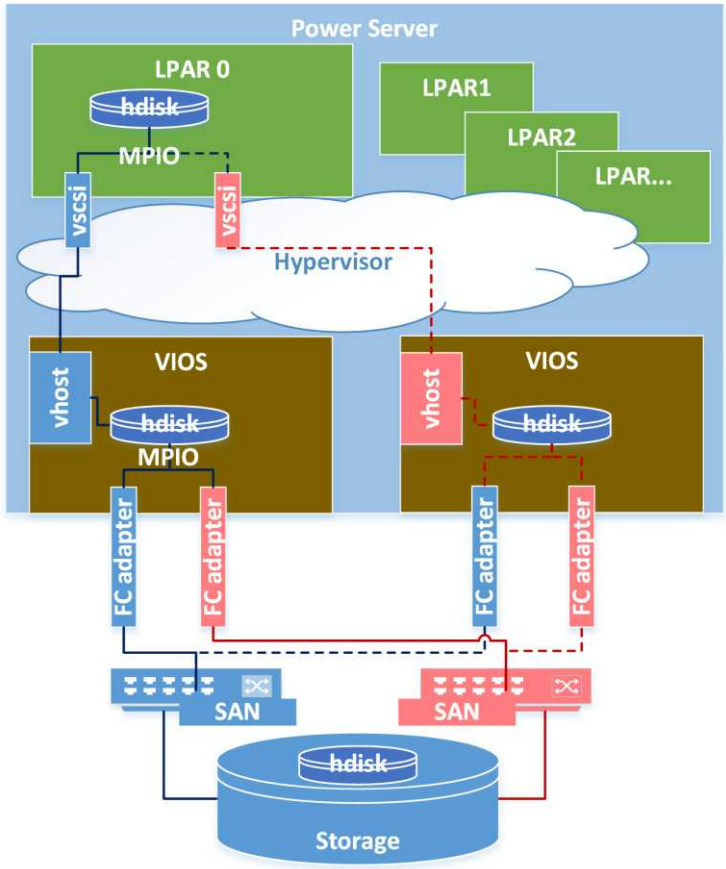

Figure 3-17: Virtual SCSI.

As you can see in the above example, the client partition is connected to two Virtual I/O Servers (there can be more or less of them) via the Virtual SCSI channel. In the above case, one (the same) LUN was presented from the external storage to both VIOSs. It has been mapped to the client LPAR. The partition recognizes that the same disk is mapped by two VIOSs, and it creates a single device with two access paths - one through each Virtual I/O Server. One of the paths is the primary path, while the other one is backup path. The priority of the path determines the VIOS through which the communication with the disk will take place. To balance the load, you can set the priorities of the paths so that one part of the LPAR's disks is served by one VIOS and the other part by the other VIOS.

The above example shows one of the most commonly used cases, in which raw disks (LUNs) are mapped to the client partition. You can also map files or logical volumes created inside the VIOS to the client partition. Further, you can use the Shared Storage Pool, which will be discussed later. The differences in the approaches are presented in Table 3-6, in which the individual columns mean:

- **LUN** - Direct LUN mapping to the LPAR. A LUN is a device that the VIOS has detected as another *hdisk*.
- **SSP** - Shared Storage Pool. Mapping the structure created in the Shared Storage Pool to the client LPAR.
- **LV** - Mapping the logical volume or file created on the VIOS to the client LPAR.

Table 3-6: Differences in SCSI mapping.

Feature	LUN	SSP	LV
Possibility of mapping through many VIOSs (*High Availability*)	YES	YES	NO
The ability to migrate the LPAR "on the fly" to another server - *Live Partition Mobility*	YES	YES	NO
The ability to replicate binary data at the external storage level	YES	NO	NO
The possibility of using snapshots, thin provisioning, and cloning at the VIOS level	NO	YES	NO
Possibility to use snapshots, thin provisioning, and cloning at the external storage level (if supported)	YES	NO*	NO*

* Thin provisioning can be used successfully in all three cases.

With VSCSI, you can create a virtual optical device and image repository on the Virtual I/O Server. In this way, you can maintain a repository of installation software that can be "mounted" at any time and on any LPAR on the server. In such a case, the partition uses the mounted image in a similar way to a physical CD/DVD device.

Sample scenarios of operations with VSCSI

Adding disk space to an LPAR

Let's assume that you have a configuration with two VIOSs. You want to map two additional disks, to be used by the database, to an LPAR while maintaining the high availability solution. Therefore, the disks should be connected through both Virtual I/O Servers.

As a first step, you create a LUN on the external disk array and map it to the appropriate HBAs of both VIOSs. On both VIOSs, you detect disks:

```
$ cfgdev
```

The disks are detected. With many disks, there may be a problem with their identification. In this case, it is best to use the information provided by the storage subsystem software (which will be different for different storage). Alternatively, you can verify devices using the creation dates of the files that represent them in the */dev* directory. In our case, *hdisk4* and *hdisk5* were added on both VIOSs (the numbering may differ on both VIOSs).

In the case of mapping disks over many VIOSs, you should disable the SCSI reservation on each of them. If you do not do so, then in the case of primary Virtual I/O Server failure, the LPAR will not be able to use the alternate access path through the secondary VIOS because the disk was reserved by the primary VIOS. Checking the SCSI reservation:

```
$ lsdev -dev hdisk4 -attr | grep reserve_policy
reserve_policy  single_path                Reserve Policy                  True+
```

```
$ lsdev -dev hdisk5 -attr | grep reserve_policy
reserve_policy  single_path                 Reserve Policy                  True+
```

Disk reservation is enabled, so it should be disabled on both VIOSs:

```
$ chdev -dev hdisk4 -attr reserve_policy=no_reserve
hdisk4 changed

$ chdev -dev hdisk5 -attr reserve_policy=no_reserve
hdisk5 changed
```

Although this is not a necessary step, it is good to set unique identifiers for the disks (PVID) at this time. In this way, you will be able to easily recognize the disk on both VIOSs and LPARs, so you perform:

```
$ chdev -dev hdisk4 -attr pv=yes
$ chdev -dev hdisk5 -attr pv=yes
```

The next step is to map the disks to the LPAR or, more precisely, to the VSCSI controller through which the LPAR is connected to the VIOS. This controller from the VIOS side is represented by a *vhostX* device. So, when you read the slot identifier from the HMC through which the VIOS is connected to the LPAR, you can identify the device on the VIOS side. It has been identified as slot 98. Let's see:

```
$ lsmap -all | grep C98
vhost10         U8233.E8B.062177P-V1-C98                0x00000014
```

Typically, the LPARs are connected to the VIOS with a single controller. So, you can alternatively identify the device on the VIOS side using the LPAR ID. You can read the LPAR ID value from the HMC, which in this case is 20. You convert the decimal value to a hexadecimal, which is 14. Then, you're looking for the right *vhost*:

```
$ lsmap -all | grep  0014
vhost10         U8233.E8B.062177P-V1-C98                0x00000014
```

You do the same on the second VIOS. The device names and disk numbers may vary between the VIOSs, so be sure to check them carefully. Then, on both VIOSs, you map the disks to the *vhost*:

```
$ mkvdev -vdev hdisk4 -vadapter vhost10 -dev aix72_data1
aix72_data1 Available

$ mkvdev -vdev hdisk5 -vadapter vhost10 -dev aix72_data2
aix72_data2 Available
```

The final parameter of the above command indicates the name of the connection. This is an important value for making the configuration clear. It is good practice to include part of the LPAR name here. This will make the verification of the mappings on the VIOS much clearer, which should in turn lead to a reduction in the number of mistakes. Once the mappings are done, you can verify them:

```
$ lsmap -vadapter vhost10
SVSA            Physloc                                     Client Partition ID
--------------- ------------------------------------------- ------------------
vhost10         U8233.E8B.062177P-V1-C98                0x00000014
```

```
VTD                   aix72_data1
Status                Available
LUN                   0x8300000000000000
Backing device        hdisk4
Physloc               U78A0.001.DNWHWGD-P1-C2-T1-W50050768013047ED-L4000000000000
Mirrored              false

VTD                   aix72_data2
Status                Available
LUN                   0x8400000000000000
Backing device        hdisk5
Physloc               U78A0.001.DNWHWGD-P1-C2-T1-W50050768013047ED-L5000000000000
Mirrored              false
# All unnecessary information has been omitted.
```

Once you have a disk mapped by both Virtual I/O Servers, you can proceed to the client partition. For disc detection on the LPAR:

```
# cfgmgr
```

You have detected new disks: *hdisk1* and *hdisk2*. You verify that the disks have redundancy and that they are visible through both VIOSs:

```
# lspath | egrep "hdisk1|hdisk2"
Enabled hdisk1 vscsi0
Enabled hdisk2 vscsi0
Enabled hdisk1 vscsi1
Enabled hdisk2 vscsi1
```

As you can see, each disk has two paths through each VIOS. One of these paths is passive. The result of this is that one of the VIOS serves I/O traffic to both disks, while the other is a backup. You change this situation, balancing the load between the two VIOSs. Each of them must be active for one of the disks as well as a backup for the other disk. You achieve this by changing the priorities of the paths. Initially, both paths have the same priority, so the traffic to both drives is served by the VIOS represented by the device with the lowest number (*vscsi0*):

```
# lspath -AE -l hdisk1  -p vscsi1 -w `lspath -l hdisk1 -p vscsi1 -F'connection'`
priority 1 Priority True

# lspath -AE -l hdisk1  -p vscsi0 -w `lspath -l hdisk1 -p vscsi0 -F'connection'`
priority 1 Priority True
```

You differentiate the priorities:

```
# chpath -l hdisk1 -p vscsi0 -a priority=10
path Changed

# chpath -l hdisk2 -p vscsi1 -a priority=10
path Changed
```

Following this change, the traffic to *hdisk1* will be served by the VIOS represented by the *vscsi1* connection, while the traffic to *hdisk2* will be served by the VIOS represented by the *vscsi0* connection. The disk from the client side still requires some changes. Its parameters typically appear as follows:

```
# lsattr -El hdisk1
PCM             PCM/friend/vscsi Path Control Module        False
algorithm       fail_over        Algorithm                  True
```

```
hcheck_cmd      test_unit_rdy    Health Check Command       True+
hcheck_interval 0                Health Check Interval      True+
hcheck_mode     nonactive        Health Check Mode          True+
max_transfer    0x40000          Maximum TRANSFER Size      True
pvid            none             Physical volume identifier False
queue_depth     3                Queue DEPTH                True
reserve_policy  no_reserve       Reserve Policy             True+
```

In order to improve the performance and reliability of the disks, the following parameters should be changed:

- **hcheck_interval** - The default setting means that the paths to the disks are not checked. By setting the value to 60, you perform an automatic path check every 60 seconds if the path is inactive. This is very important for dual VIOS configurations. With this parameter, after rebooting one of the VIOSs, the LPAR will check the path leading through it and bring it back to work. Without it, restarting the VIOSs one after the other (e.g., in case of an update) will cause the LPAR to crash, since it will lose access to the disks. If you change this parameter, for example, to 60, the path is automatically verified, and a path will be available as soon as the VIOS starts again after the service.
- **queue_depth** - The depth of the queue to the disk. The default setting of three significantly limits the performance of I/O operations. It is worth equalizing this value with the setting that a given disk has on the VIOS (i.e., usually 20–30). More information about this parameter is included in the section on performance considerations.

Change the following parameters on the disks:

```
# chdev -l hdisk1 -a hcheck_interval=60
hdisk1 changed

# chdev -l hdisk2 -a hcheck_interval=60
hdisk2 changed

# chdev -l hdisk1 -a queue_depth=20
hdisk1 changed

# chdev -l hdisk2 -a queue_depth=20
hdisk2 changed
```

In addition, you should change the *vscsi* interface settings:

- **vscsi_err_recov** - Set by default as *delayed_fail*. A change to *fast_fail* will result in faster switching to the backup paths in the event of a failure, thereby increasing the chance that the crash will be unnoticeable from the user's point of view.
- **vscsi_path_to** - The *vscsi* timeout setting. Setting the parameter to 30 will result in the faster occurrence of a timeout in case of failure and thus less of a delay in using the backup path.

```
# lsattr -El vscsi0   # The default VSCSI parameters to be changed.
rw_timeout      0           Virtual SCSI Read/Write Command Timeout True
vscsi_err_recov delayed_fail N/A                                    True
vscsi_path_to   0           Virtual SCSI Path Timeout               True

# chdev -l vscsi0 -a vscsi_err_recov=fast_fail -P   # Change of parameters.
vscsi0 changed

# chdev -l vscsi0 -a vscsi_path_to=30 -P
```

```
vscsi0 changed
```

After the restart, the LPAR will gain two new, properly configured disks. You can then proceed with further configuration according to the original plan.

Configuring the software repository on the VIOS

It can prove very troublesome to install software from DVDs. The difficulty is caused by the need to go to the server room in order to place the disc in the drive. The solution to this problem is the VIOS functionality available under the names VIOS Media Repository and Virtual Media Library. It allows you to create a repository of DVD images on the Virtual I/O Server. The Images can then be used by any LPAR on the server using a virtual optical device.

In this scenario, you create a repository on the Virtual I/O Server and load the AIX 7.2 installation image there.

We create a 4 GB repository in the *rootvg* group. You can choose another dedicated volume group:

```
$ mkrep -sp rootvg -size 4G
Virtual Media Repository Created
Repository created within "VMLibrary" logical volume
```

We verify the repository. An empty repository has appeared, and its physical location is the file system */var/vio/VMLibrary*:

```
$ lsrep
Size(mb) Free(mb) Parent Pool         Parent Size      Parent Free
    4079     4079 rootvg                   81792            43904

$ df -k | grep VMLibrary
/dev/VMLibrary     4194304   4176952    1%        4     1% /var/vio/VMLibrary
```

The next step is to create a virtual disk from AIX 7.2 in the repository. Create a *.iso* file from the physical media and copy it to the VIOS. Then, create a virtual disk:

```
$ mkvopt -name AIX72TL0SP2DVD1OF2 -file /home/padmin/AIX_V7.2_TL_72002_DVD1OF2.iso -ro
```

The indicated file has been copied to the repository and registered under the given name as read-only media:

```
$ lsrep
Size(mb) Free(mb) Parent Pool         Parent Size      Parent Free
    4079      921 rootvg                   81792            43904

Name                                                File Size Optical         Access
AIX72TL0SP2DVD1OF2                                       3158 None            ro

$ ls -l /var/vio/VMLibrary
total 6466112
-r--------    1 root     staff    3310649344 Sep 29 14:22 AIX72TL0SP2DVD1OF2
drwxr-xr-x    2 root     system         256 Sep 29 14:09 lost+found
```

If it becomes necessary to add more images to the repository, you can enlarge it at any time.

```
$ chrep -size 1G   # Enlarges the repository by 1 GB.
```

Mapping the image from the repository to the LPAR

An LPAR can only use the repository if it has a virtual optical device. Therefore, you need to identify the *vhost* device corresponding to the LPAR on which the installation process will be performed (the process of identifying the *vhost* device is described in more detail in the scenario concerning the addition of disk space). In this case, it is a *vhost7* device.

Create a virtual optical device for the LPAR identified by the *vhost7*:

```
$ mkvdev -fbo -vadapter vhost7
vtopt0 Available
```

The LPAR has a device that is empty:

```
$ lsmap -vadapter vhost7
SVSA            Physloc                                     Client Partition ID
--------------- ------------------------------------------- ------------------
vhost7          U8233.E8B.062177P-V2-C23                    0x00000010

VTD                   vtopt0
Status                Available
LUN                   0x8100000000000000
Backing device
Physloc
Mirrored              N/A
```

Load the AIX 7.2 installation image to the virtual device:

```
$ loadopt -vtd vtopt0 -disk  AIX72TL0SP2DVD1OF2
```

The image is loaded:

```
$ lsmap -vadapter vhost7
SVSA            Physloc                                     Client Partition ID
--------------- ------------------------------------------- ------------------
vhost7          U8233.E8B.062177P-V2-C23                    0x00000010

VTD                   vtopt0
Status                Available
LUN                   0x8100000000000000
Backing device        /var/vio/VMLibrary/AIX72TL0SP2DVD1OF2
Physloc
Mirrored              N/A

$ lsvopt
VTD             Media                                           Size(mb)
vtopt0          AIX72TL0SP2DVD1OF2                                 3158
```

From now on, you can use the drive in the same way as you would use its physical equivalent.

Performance considerations

As a disk space virtualization method, VSCSI is considered to be less efficient than NPIV. The performance is the first argument for choosing NPIV when confronted with VSCSI. Although

theoretically you can agree with this, in practice the performance differences may be small. The myth of a big difference stems from the fact that the NPIV configuration built on the default settings will work efficiently, while the VSCSI configuration built on the default settings will limit performance. The problem is that the administrator often leaves parameters with default settings instead of optimizing them. The following information will show you what aspects you should pay attention to when configuring disk space virtualization based on VSCSI.

Disk queues

The queue is a very important parameter. The LPAR can simultaneously run as many operations on the specified LUN as the length of the queue on the disk. If the queue has three elements, the system can only run three operations simultaneously, and the rest will have to wait their turn.

The first problem encountered concerns the length of the queue, which from the LPAR side is three by default, while from the VIOS side it is usually 20–30 (it may depend on the external storage and its drivers).

```
# lsattr -El hdisk1 | grep queue_depth   # The default queue length from the LPAR level.
queue_depth     3               Queue DEPTH                 True

# lsattr -El hdisk1 | grep queue_depth   # The default queue length from the VIOS level.
queue_depth     20              Queue DEPTH                 True
```

Therefore, if you are configuring disk space by mapping the LUN directly from the VIOS to the LPAR, you should coordinate these parameters. The value 20–30 is a universal value and is hence suitable in most cases. Increasing the queue above this value can have a positive effect in some cases, although it depends on the performance characteristics of the application and storage.

When changing the queue settings, keep in mind that they are relevant for the entire transmission path. When changing the parameters of the queue, you have to take this into consideration, since the bottleneck may be found in any of the following components:

- **HBA of an external storage** - There are a defined number of operations that a given port can handle at a given time. Typically, the queue sizes on external storage ports are much larger than those on hosts, although the number of hosts that communicate with a given storage port can be large. However, it is usually not possible to influence the size of this parameter.
- **HBA of a host** - The VIOS ports in our case. Each port has a parent queue to the queues of individual disks/LUNs. By creating a very careful configuration, you can attempt to adjust the queue length on all the disks mapped to a given port up to the length of the port queue. Nevertheless, these parameters are usually configured in a disproportion, considering the fact that the queues for disks are rarely saturated at 100% and, even then, it does not apply to all systems at the same time.
- **Disk on a VIOS** - The size of the queue, usually 20–30, is often determined by the external disk array drivers.
- **Disk on an LPAR** - The size of the queue, three by default, is insufficient in terms of performance and inconsistent with the size of the queue on the same disk on the VIOS.

The queues on the transmission path and other important parameters are shown in Figure 3-18.

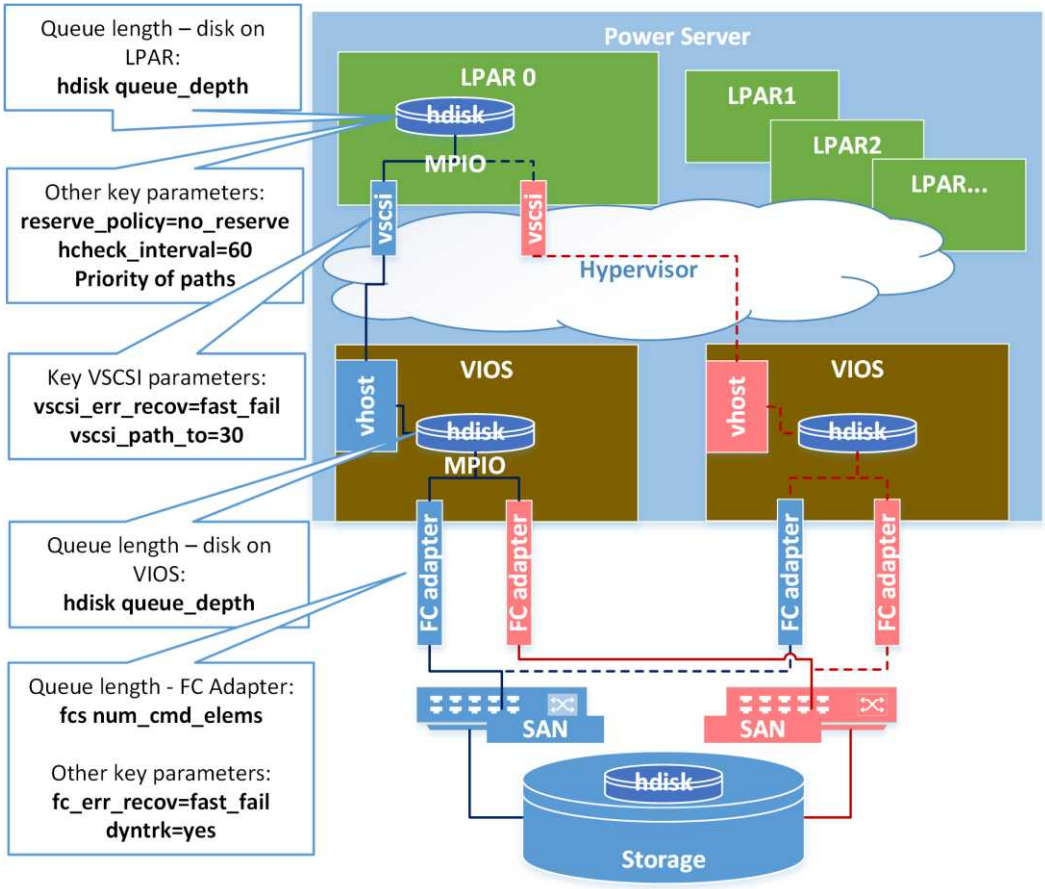

Figure 3-18: VSCSI - Queue length and other key parameters.

The balancing of traffic on two VIOSs

Having two VIOSs on the server is a fairly typical configuration from the high availability perspective. In this case, each disk operates through one VIOS, while the other VIOS maintain the backup paths used in the event of the failure of the first one. From a performance point of view, the problem is that regardless of the number of disks assigned to the LPAR, each disk by default operates through the same VIOS.

If the LPAR has several disks, then you should differentiate the priorities of the individual paths so that some of them operate through the first VIOS, and some through the second. The method of implementation is described in one of the scenarios in this chapter. In addition, when you create logical volumes in a volume group that consists of several disks, you must apply an allocation policy that stripes data across all the disks. If the volume group is constructed of small allocation units (small physical partitions), then you will achieve effective striping between the disks. When you change the path priorities, you will also achieve effective striping between the VIOSs.

Other aspects that potentially point to the weakness of VSCSI include the default parameters related to system availability, especially in the case of a configuration composed of two VIOSs. The default parameters that should be changed in this situation:

- ***reserve_policy*** - Disk parameter on the VIOS. In the case of mapping disks over many VIOSs, it should be set to *no_reserve*. By default, it is set to *single_path*.

```
# chdev -dev hdiskX -attr reserve_policy=no_reserve
```

- ***hcheck_interval*** - Disk parameter on the LPAR. The default value is *0*, which implies no path checking. The recommended value is *60*, which requires the verification of the paths every 60 seconds

```
# chdev -l hdisk0 -a hcheck_interval=60 -P
```

The other parameters to which you should pay attention are related to the operation of the SAN network rather than to VSCSI. However, it is important to change the FC ports settings (fscsiX devices). The parameters to change are:

- ***fc_err_recov=fast_fail*** - Allows the driver to quickly respond to problems on one of the paths. The controller disables this path, thereby enabling work on the remaining active ones.

```
# chdev -dev fscsiX -perm -attr fc_err_recov=fast_fail
```

- ***dyntrk=yes*** - Makes the driver aware of and resistant to potential changes in the SAN architecture.

```
# chdev -dev fscsiX -attr dyntrk=yes
```

NPIV (N_Port ID Virtualization)

NPIV is a standard for the virtualization of access to SAN networks. It is a feature that occurs on many hardware platforms, and it works in a similar way on both the x86 platform and the POWER platform. In order to explain how it works, it is worth briefly mentioning the origin of this feature in the SAN.

The Storage Area Network allows you to share the resources connected to it. Through this sharing, a single disk array, a single tape drive, or any device connected there can be used by many servers. By connecting devices to each other in a SAN network, you create zones that can be created in several ways. The most popular and generally most effective way is to use WWN (*World Wide Name*) addresses.

Each adapter port has its own WWPN (*World Wide Port Name*) address. This is a 64-bit address, for example:

```
# lscfg -vl fcs0 | grep Network
        Network Address.............10000000C99ACBF4
```

When you perform configurations in the SAN, on servers, and on disk arrays, you use these addresses. However, they have a key disadvantage. They are long, so they do not fit into the philosophy of the SAN network, the assumption of which is to minimize delays and minimize the transmission of unnecessary information. Thus, the network itself dynamically translates 64-bit WWN addresses into 24-bit addresses known as FCID (*Fiber Channel ID*). They consist of three parts:

- **Domain** - An 8-bit switch ID in the network.
- **Area** - An 8-bit port ID in the network (it can also use an additional bit from the part that is intended for the Node).
- **Node** - An 8-bit node ID. This part of the address is used when using devices in a loop.

Due to the increasingly rare use of loops in the SAN, the element of addressing described as a Node is unused. This, typically empty, part of the address allows the use of NPIV. With it, a larger number of addresses can be assigned to a single physical port. Thus, from the point of view of the SAN, the physical port address usually differs from the virtual port address in terms of the last eight bits. Examples of addressing:

- 020200 (hexadecimal) - An example of a 24-bit physical port address.
- 020203 (hexadecimal) - An example of a 24-bit virtual port address.

The advantage of virtualizing access to a SAN network using NPIV is the ability to work in a similar way to if the system was connected to a physical adapter. Therefore, you can use advanced drivers for external storage, use tape drives in the SAN, or use advanced cluster solutions, which often require direct communication with the external storage.

NPIV in AIX

For the NPIV configuration in the AIX operating system, several conditions must be met:

- You must have HBA adapters that support virtualization. All the adapters from 8 Gb/s should have this functionality, although it is worth checking in the device documentation.
- You must have virtualization licenses and at least one Virtual I/O Server.
- The SAN should support this solution. Typically, this feature is enabled on specific ports on the SAN switch.

Virtualization, as in the case of VSCSI, is carried out by the VIOS. It must have physical HBAs that will be virtualized. It is worth considering the use of high availability mechanisms in the form of dual VIOS installations; otherwise, the shut down or failure of a single Virtual I/O Server will result in the loss of the LPAR's access to the SAN.

In the case of VSCSI, disk array drivers were installed on the VIOS, while the LPAR used the native VSCSI drivers. When using NPIV, the situation is different. The Virtual I/O Server does not require any additional drivers, since it only transfers specific traffic to the LPAR. However, the LPAR must be able to work with the disk array, and it usually requires the installation of drivers to support the disk array. The advantage of this mode of operation is that from the VIOS level, the configurations are performed once. After that, device/disk mapping from the SAN to the LPAR does not require any action on the VIOS.

The way in which NPIV works on the POWER platform is illustrated in Figure 3-19.

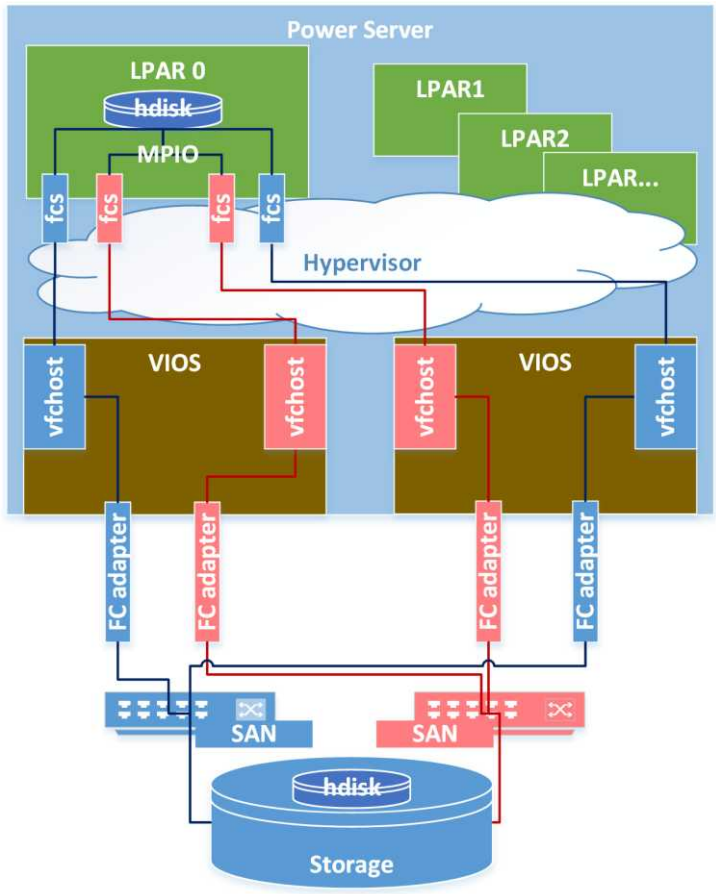

Figure 3-19: NPIV - Two VIOS partitions.

It is worth noting that queues also play a big role in this situation as well. Although the queues for the detected disks are quite large by default, you should pay attention to the queues for the virtual and physical adapters and their association. You should keep in mind that all virtual adapters, although they have their own queues, also use the queue of the physical adapter to which they are assigned on the VIOS. Thus, in the case of multiple virtual adapters assigned to one physical adapter, you should consider increasing the physical adapter queue on the VIOS or reducing the queues of virtual adapters on the LPARs.

```
# lsattr -El fcs0 | grep num_cmd_elems # A queue that you can modify on the physical and virtual
adapters.
num_cmd_elems 500        Maximum number of COMMANDS to queue to the adapter True
```

Examples of NPIV scenarios

Connecting an LPAR to the SAN using NPIV

In this scenario, we have one VIOS named 5437v1. We connect LPAR SB5437_AIX72 to the SAN

network with the preservation of the possible redundancy (i.e., connecting to two different fabrics).

We create two virtual fibre channel adapters for the LPAR (Figure 3-20) and two virtual fibre channel adapters for the VIOS (Figure 3-21):

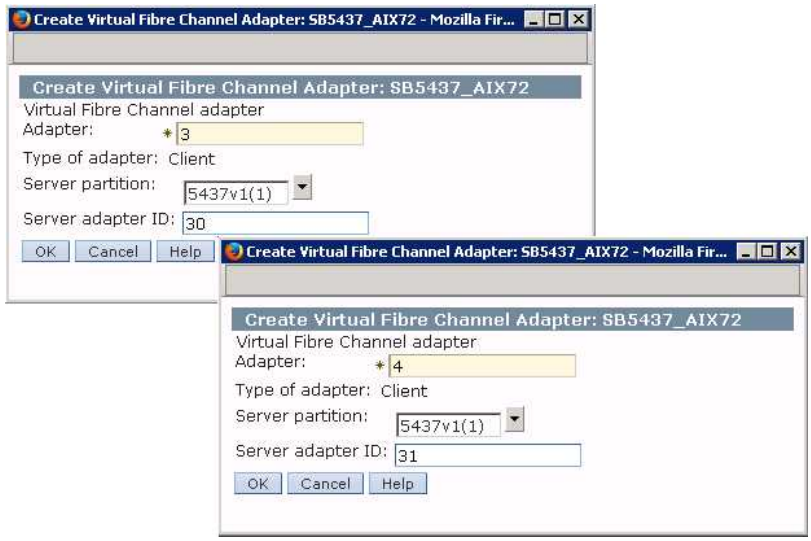

Figure 3-20: HMC - Creation of the Virtual Fibre Channel Client Adapter - LPAR.

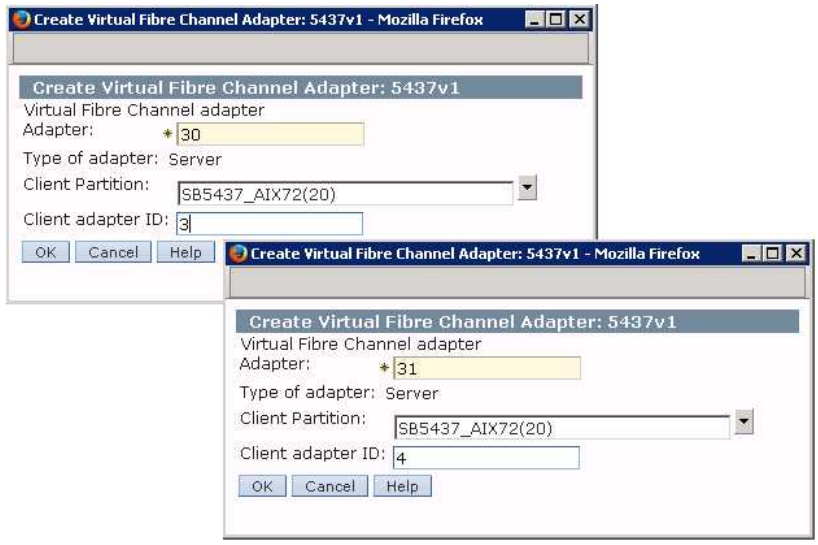

Figure 3-21: HMC - Creation of the Virtual Fibre Channel Server Adapter - VIOS.

After adding the virtual adapters that make up the connection between the LPAR and the VIOS, we detect the created devices.

The *fcs0* and *fcs1* devices are detected on the LPAR:

```
# cfgmgr
```

```
# lsdev -Cc adapter | grep fcs
fcs0   Available C3-T1 Virtual Fibre Channel Client Adapter
fcs1   Available C4-T1 Virtual Fibre Channel Client Adapter
```

The *vfchost0* and *vfchost1* devices are detected on the VIOS:

```
$ cfgdev
$ lsmap -all -npiv | grep -e "C30"   # We check the device corresponding to our adapter number: C30.
Vfchost0     U8233.E8B.062177P-V1-C30               20
$ lsmap -all -npiv | grep -e "C31"   # We check the device corresponding to our adapter number: C31.
Vfchost1     U8233.E8B.062177P-V1-C31               20
```

In the next step, we map the vfhostX device to a specific physical FC adapter on the VIOS. We should know which adapters are connected to which fabrics. Using this knowledge, we can provide redundancy, thereby enabling the LPAR to work in two independent SAN fabrics. In our case, we have two FC adapters that we can use for mapping. Each of the physical adapters can support 64 virtual adapters.

```
$ lsnports
name             physloc                        fabric tports aports swwpns  awwpns
fcs0             U78A0.001.DNWHWGD-P1-C2-T1          1     64     64   2048    2036
fcs1             U78A0.001.DNWHWGD-P1-C2-T2          1     64     64   2048    2036
```

We perform the mappings:

```
$ vfcmap -vadapter vfchost0 -fcp fcs0
$ vfcmap -vadapter vfchost1 -fcp fcs1
```

We then check the mappings. As you can see, the connection has the status *LOGGED_IN*, which means that the adapters have logged into the SAN network correctly.

```
$ lsmap -vadapter vfchost0 -npiv
Name          Physloc                            ClntID ClntName       ClntOS
------------- ---------------------------------- ------ -------------- -------
Vfchost0      U8233.E8B.062177P-V1-C30               20 SB5437_AIX72   AIX

Status:LOGGED_IN
FC name:fcs0                    FC loc code:U78A0.001.DNWHWGD-P1-C2-T1
Ports logged in:1
Flags:a<LOGGED_IN,STRIP_MERGE>
VFC client name:fcs0            VFC client DRC:U8233.E8B.062177P-V20-C3

$ lsmap -vadapter vfchost1 -npiv
Name          Physloc                            ClntID ClntName       ClntOS
------------- ---------------------------------- ------ -------------- -------
vfchost1      U8233.E8B.062177P-V1-C31               20 SB5437_AIX72   AIX

Status:LOGGED_IN
FC name:fcs1                    FC loc code:U78A0.001.DNWHWGD-P1-C2-T2
Ports logged in:1
Flags:a<LOGGED_IN,STRIP_MERGE>
VFC client name:fcs1            VFC client DRC:U8233.E8B.062177P-V20-C4
```

From now on, the WWPN addresses assigned to our virtual adapters should appear in the SAN. You can conduct the process of creating appropriate zones as well as mapping specific disks and other devices to our adapters. If you want to check the WWN address assigned to the virtual adapters, you can do so from the client LPAR side or from the HMC:

```
# lscfg -vl fcs0 | grep Network   # Checking from the LPAR side.
        Network Address.............C0507602DD6E0204
```

You can check the WWN addresses from the HMC side (Figure 3-22):

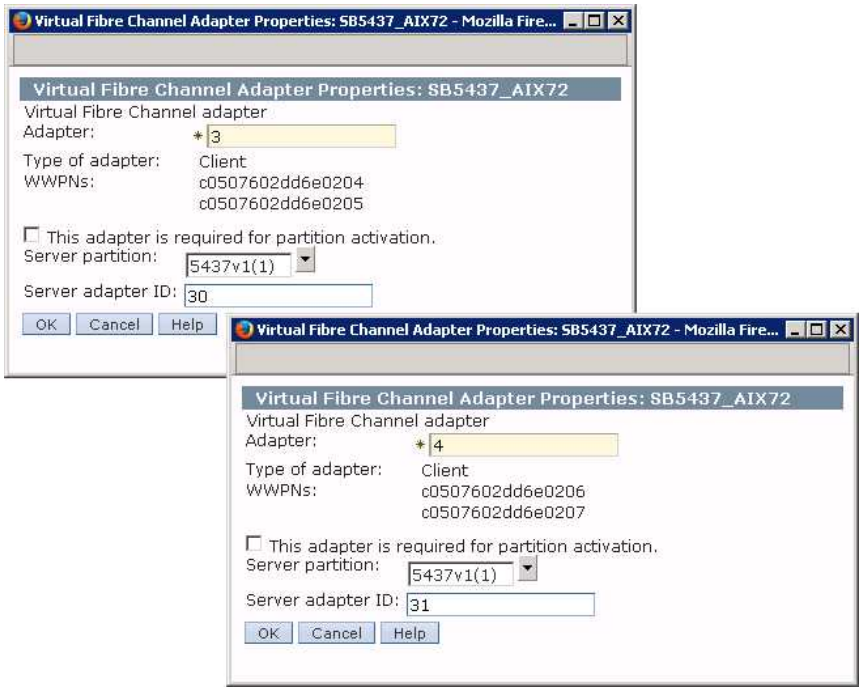

Figure 3-22: Checking the WWN addresses assigned to the virtual adapters.

On the console, you can see two WWN addresses assigned to one adapter. The second address (the backup) is not visible in the SAN. It exists to support the Live Partition Mobility (LPM) process. In the case of LPM migration to another server, the backup address becomes visible on the migrated LPAR, and the primary address becomes the backup. When planning to migrate, you must use both addresses to create zones in the SAN. You can find more information about Live Partition Mobility later on in this chapter.

NPIV vs VSCSI

Nowadays, there is a perception that NPIV should be used instead of VSCSI wherever possible. This view is often based on the belief that newer technology will always work better for organizations. Meanwhile, the use of both NPIV and VSCSI could be a good choice for a given system depending on the way the organization operates, the scale of operations, or the individual needs of a given system.

To facilitate a better understanding of the characteristics of a given approach, the approaches were collected together in Table 3.7 along with the assessment and a description of what it implies. In the case of VSCSI, let's assume that the VIOS is used only for directly mapping a given (whole) disk to a particular LPAR.

Table 3-7: Comparison of NPIV and VSCSI.

Feature	VSCSI Evaluation	NPIV Evaluation
Performance	*Very good after additional configuration*	*Very good by default*
NPIV - You have virtual HBAs that are equivalent to physical adapters. As a result, you can use dedicated drivers on the LPAR to manage the multiple disk access paths provided with the external storage. After applying the appropriate drivers, the traffic is intelligently distributed between all the paths, which optimizes performance. Usually, the default parameters assigned to the disks (e.g., queue size) are correct in terms of performance; therefore, without any additional configuration, you will achieve the expected performance.		
VSCSI - Installation of external storage drivers is performed on the VIOS. The LPAR uses the standard VSCSI drivers. By default, the parameters assigned to the LPAR disks are usually inefficient. The efficiency can be very close to that of NPIV, so long as you make an additional configuration, which is described in the chapter on VSCSI. This configuration, from the point of view of the client's LPAR, includes: • Modification of the priorities of the disk paths; • Changing the length of the disk queues; and • An appropriate volume group configuration that provides stripping between disks and between paths.		
The amount of configuration work	*Large on the side of VIOS and AIX*	*Average on the storage side*
NPIV - During configuration, each storage administrator must: • Create zones (usually in two fabrics) for active WWN addresses; • Create zones (usually in two fabrics) for inactive WWN addresses, if you use the LPM feature; and • Create an appropriate configuration in the context of the new WWN addresses on the storage. The Virtual I/O Server administrator must perform the initial configuration on the VIOS and the HMC. Then, all the disks can be mapped directly from the storage to the operating system without the need for additional configuration on the VIOS.		
VSCSI - When configuring the system, the storage administrator does not have to create zones, since they were created during the initial VIOS configuration. For the same reason, he does not have to perform storage configuration in the context of the new WWN addresses. In this case, the VIOS administrator must detect the new LUNs and, if necessary, change the parameters for the SCSI reservations and performance. Then, he must map them to the LPAR. The operating system administrator must also perform some additional activities: • Change the performance and availability parameters (queues, health check, etc.). • Change the priorities of some disk paths to balance the load between the two VIOSs.		
Management	*More work on the VIOS*	*More work on storage*
NPIV - From the point of view of further system management, activities such as adding, changing, and removing disks from the storage itself are transparent for the VIOS. When you map disks from another		

storage to the operating system, you may have to install new drivers or even add additional virtual adapters. This may occur due to the impossibility of working several drivers with the same adapters. In this configuration, there are less likely to be "ghost" disks. After removing the LPAR, the WWN addresses become inactive, and it is easier to find unused disks. The LPAR removal requires additional cleaning work, such as deleting relevant zones and storage configurations.
VSCSI - Every action related to adding, changing, and removing disks requires intervention on the VIOS. However, there is no need to maintain the storage drivers in the operating system, since they are maintained on the VIOS. Attaching disks from another storage is transparent for the LPAR, because it operates on storage using the standard VSCSI drivers. For this configuration, there are more likely to be "ghost" disks still mapped to the VIOS after deleting the LPAR. The removal of the LPAR does not require the modification of zones or changes on the storage side regarding the WWN addresses.

Tape drives, and storage control	*Lack of or difficult cooperation*	*Good cooperation*
NPIV has all the features of a physical connection to the SAN, so it can be used for direct cooperation with tape drives and disk arrays. Such direct cooperation with disk arrays can be used in advanced configurations, for example, to automate the operation of clusters between multiple locations or to control data replication.		
Due to indirect mappings by the VIOS, ***VSCSI*** is not able to work directly with tape drives. An exception is seen in the case of the drives connected via the physical SCSI interface directly to the VIOS. Often, we cannot control data replication in this way.		

SSP (Shared Storage Pool)

Disk space mapping to LPARs via NPIV, or LUNs mapped directly to LPARs via VSCSI, has one major disadvantage. This disadvantage is the need to perform actions on external storage (VSCSI) and in the SAN (NPIV) when creating the LPARs or modifying the disk space. This can prove uncomfortable, especially in large companies, where the role of the system administrator is separate from that of the storage administrator. Indeed, there are situations in which two people from different teams have different approaches to a given activity.

To overcome the above issue and enable the administrators of operating systems and the Power platform to operate independently, the Shared Storage Pool concept was created. SSPs are created on the Virtual I/O Server. You can compare them to volume groups, which have been expanded with new features and can be active on many servers at the same time. The operation of a SSP is illustrated in Figure 3-23.

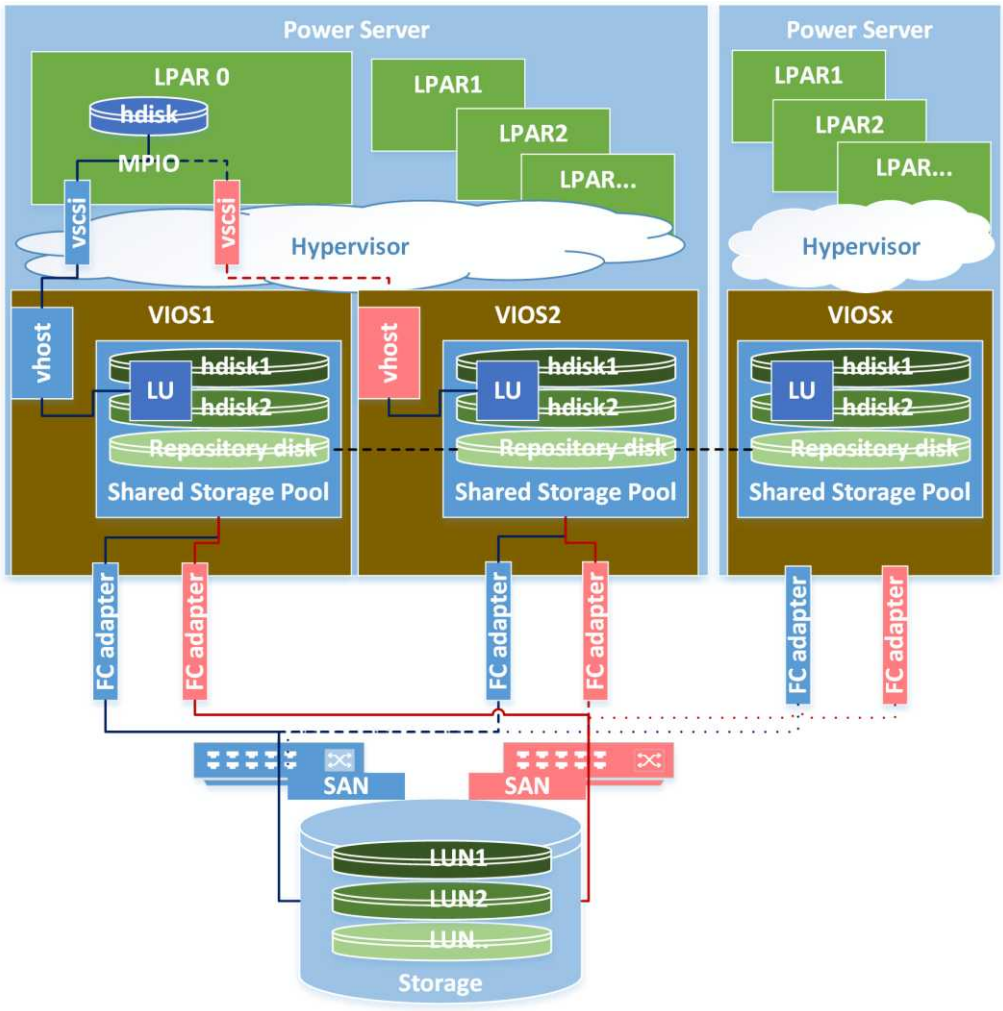

Figure 3-23: Shared Storage Pools.

On disks in a SSP, you can create so-called Logical Units (LU), which are presented to the LPARs via VSCSI and seen there as standard disks (*hdisk*). By creating this structure, you gain additional features that you can use on the VIOS, such as *Thin Provisioning*, or the ability to create snapshots. In addition, the space assigned to the SSP is fully managed by the Power platform administrator, which simplifies the process of creating and modifying LPARs.

A single SSP can be used by many VIOS partitions. It works through the CAA cluster mechanisms (Cluster Aware AIX) and RSCT. The installation uses the repository disk, whose task is to maintain an alternative path of information exchange between the cluster nodes (VIOS partitions) in the event of a LAN failure.

The Shared Storage Pools' functionality is constantly evolving. Their current limitations are detailed in Table 3-8:

Table 3-8: Shared Storage Pools - Limitations.

Feature	Maximum value
The number of VIOS partitions in the cluster	16
The number of LUNs that you can use to create a pool	1024
The number of LUs created in the pool	8192
The number of LPARs per one VIOS	200
The size of the LUNs from which you can build a pool	16 TB
The size of the pool	512 TB
The size of the disk (LU) mapped to the LPAR	4 TB

The disadvantage of this solution is the inability to use the advanced features of external storage systems that support disaster recovery solutions, such as synchronous or asynchronous replication. Therefore, the ideal application of SSPs concerns test and development systems as well as production systems that do not use replication solutions at the level of external storage.

Network virtualization

Network virtualization is another key element of a well-functioning virtualization. On the Power platform, there are several ways to virtualize access to the external network:

- **Shared Ethernet Adapter (SEA)** - A classic virtualization method based on the Virtual I/O Server. It is a method specific to the Power platform as well as the one most commonly used on it.

- **Single Root I/O Virtualization (SR-IOV)** - A relatively new method widely used in IT and available on various hardware platforms. It can be used independently of the Virtual I/O Server. SR-IOV is a standard that allows the virtualization of the adapter with minimal hypervisor participation and without the need for the Virtual I/O Server, which results in higher transmission performance. You can only virtualize the network using this standard if you have specialized PCIe adapters. You can get up to 64 logical ports from one adapter. These ports can be allocated to individual LPARs. An important extension of this method on the Power platform is the possibility of further virtualization using the VIOS. You can create vNIC adapters that, while retaining the SR-IOV performance gains, enable it to be managed by the VIOS. This, in turn, gives you the opportunity to use advanced virtualization features such as Live Partition Mobility together with SR-IOV.

- **Integrated Virtual Ethernet (IVE)** - A virtualization method specific to the Power platform. It is also known as the Host Ethernet Adapter (HEA). It is available with some of the older server models (prior to Power 8). This method was created to minimize delays and maximize bandwidth, while providing the possibility of virtualization. It works in a similar way to SR-IOV.

Issues related to the functionality of a *Shared Ethernet Adapter* will now be discussed in greater detail.

Shared Ethernet Adapter (SEA)

A Shared Ethernet Adapter is a native method of virtualizing access to the external network on the Power platform. It requires the Virtual I/O Server. It enables the sharing of physical LAN adapters by multiple LPARs. It also allows multiple LPARs to work through individual physical adapters.

An SEA supports many VLANs. With it, you can create scalable and highly available solutions. The hypervisor embedded in the machine's firmware and VIOS is used with this virtualization method.

The hypervisor provides the functionality of the internal virtual LAN switches built into the server. There can be many virtual switches in a single server. They behave in the same way as physical switches. They work in the second layer of the ISO OSI model, and they allow you to create 4094 VLANs. Each virtual network adapter has a MAC address assigned to it, and it is located in the VLAN you specified. As virtual switches behave analogously to physical switches, LPARs that have virtual adapters connected to the same virtual switch and in the same VLAN can communicate with each other inside the server via a hypervisor. In this case, network communication between the virtual adapters is performed by copying the memory by the hypervisor.

LPARs that have virtual adapters connected to different virtual switches or that are in the different VLANs can communicate with each other through the external world. For example, via an external LAN switch using the VIOS with a configured SEA.

The described functionalities are shown in Figure 3-24.

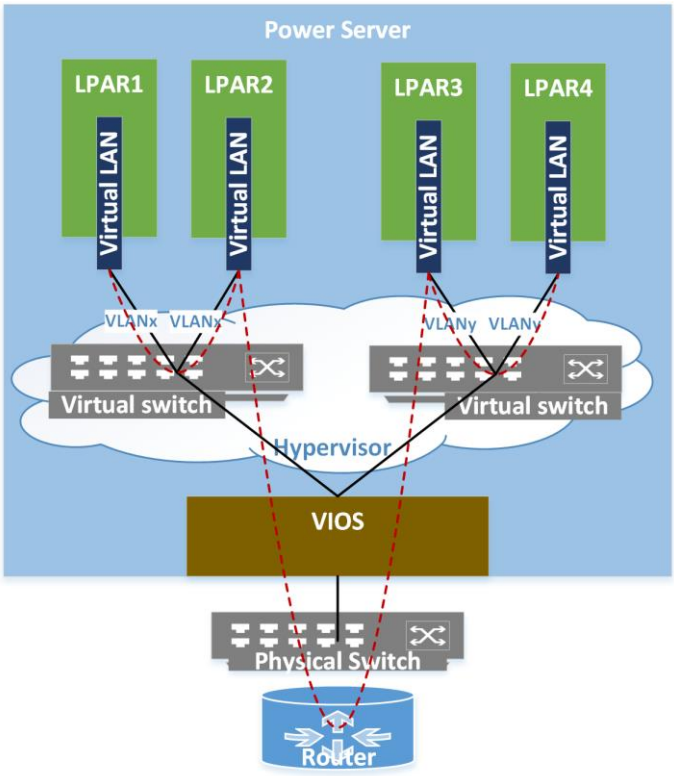

Figure 3-24: Virtual Network - Virtual Switches.

Virtual I/O Servers are the key to providing communication with the outside world. They act as a bridge between the network inside the server and the outside world. By using VIOSs, you can create SEA adapters that are a bridge between the internal and external worlds. There may be multiple SEA adapters. Each of them can support a single VLAN or many VLANs, and they can use single or aggregated physical links. A typical solution of this type is presented in Figure 3-25.

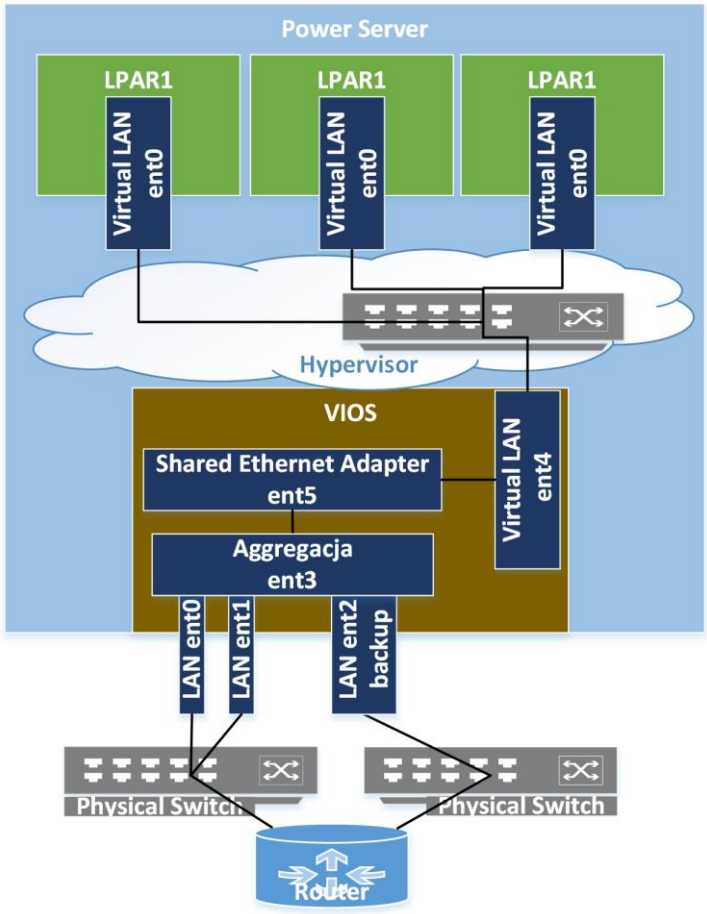

Figure 3-25: Shared Ethernet Adapter.

In addition, you can ensure high availability by creating two SEA adapters on two VIOS partitions and connecting them with a "heartbeat." Such a configuration will also ensure constant communication with the external world in the event of the failure of one of the VIOSs' LPARs. It can also ensure the load distribution between both VIOSs using the *sharing* mode. In this case, some of the VLANs are served by one VIOS, and some by the other. A configuration with two Virtual I/O servers is shown in Figure 3-26:

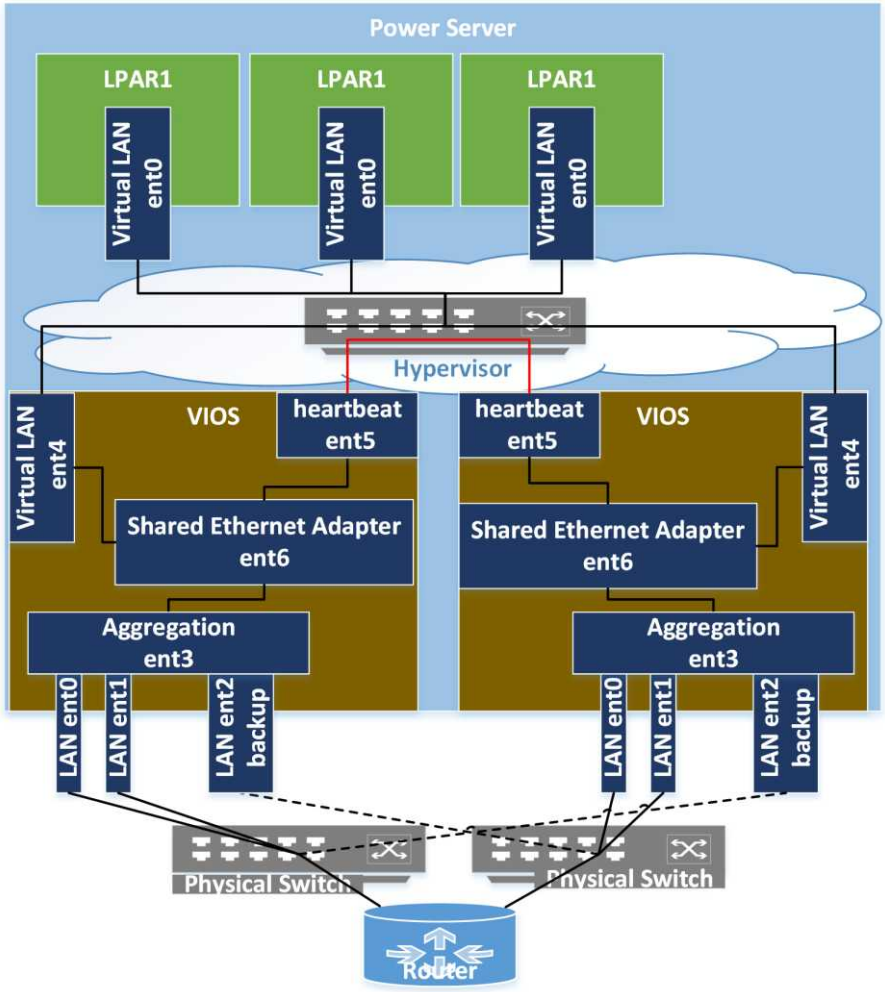

Figure 3-26: Shared Ethernet Adapter - Two VIOS partitions.

SEA configuration scenario

The following scenario will demonstrate how to configure the VIOS pair (VIOS1 and VIOS2) in order to support the VLANs numbered 22, 23, 24, and 25. The configuration is intended to provide tolerance to the failure of a single VIOS as well as network traffic load distribution between both Virtual I/O Servers.

On each VIOS, we have a single physical adapter, *ent0*, which is configured as a trunk and supports VLANs 22, 23, 24, and 25. If we want to provide redundancy at the level of a single VIOS, several adapters can be combined into a single communication channel (*EtherChannel*) and then used to create a *Shared Ethernet Adapter*.

On the first VIOS, we create three adapters that will be detected as *ent1*, *ent2*, and *ent3*. The creation of adapters on the HMC is illustrated in Figure 3-27:

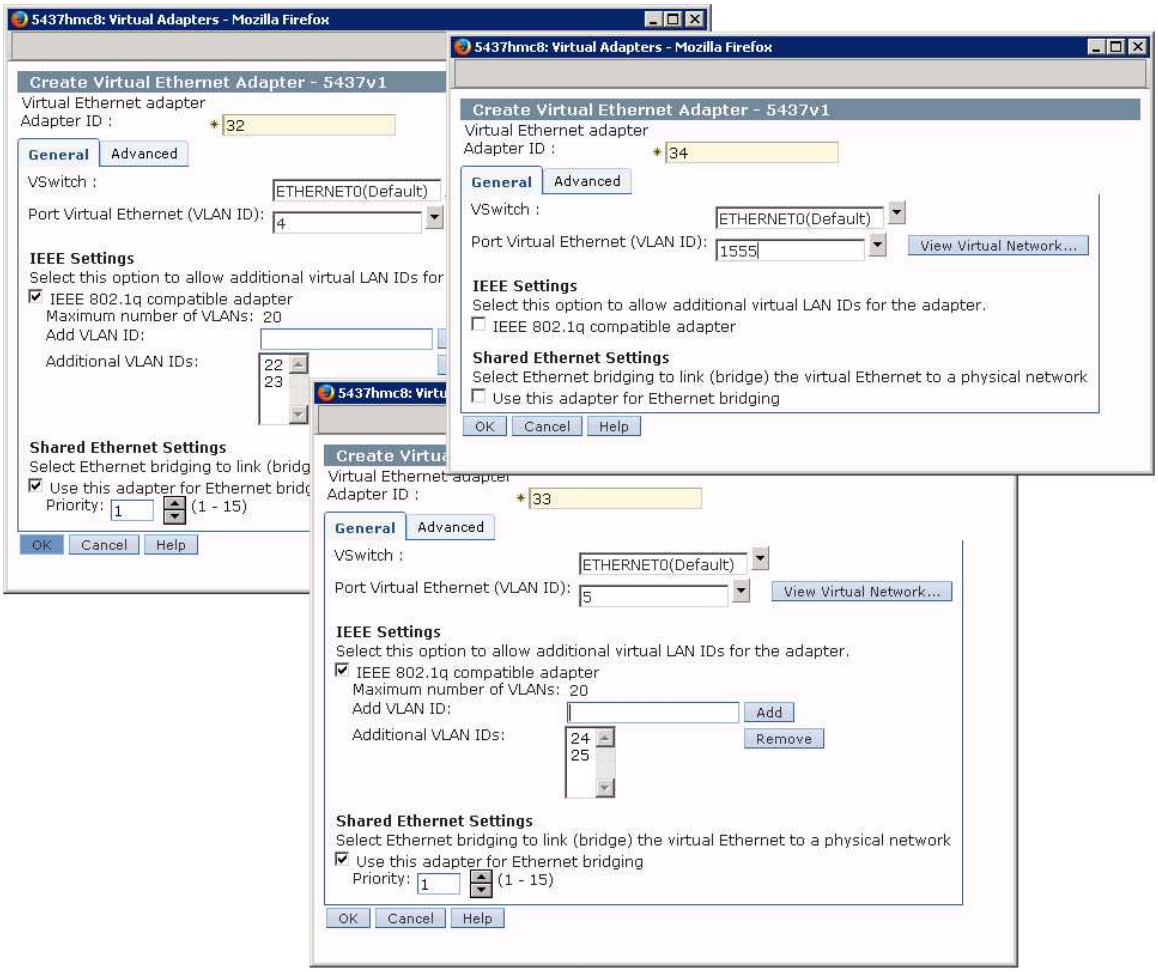

Figure 3-27: HMC - VIOS1 - Creating interfaces for an SEA.

- ***ent1*** - The adapter with ID 32 supports VLANs 22 and 23. The "*Port Virtual Ethernet (VLAN ID):*" must have a unique value. It is irrelevant, but it must be unique and point to a VLAN that is not used on the server. The adapter is marked as *bridge* and it has priority *1*.
- ***ent2*** - The adapter with ID 33 supports VLANs 24 and 25. The "*Port Virtual Ethernet (VLAN ID):*" must have a unique value. It is irrelevant, but it must be unique and point to a VLAN that is not used on the server. The adapter is marked as *bridge* and it has priority *1*.
- ***ent3*** - The adapter with ID 34 is a classic virtual adapter. It is in the unique VLAN 1555, which we chose to build a control channel between the SEA on VIOS1 and the SEA on VIOS2.

If we wanted to build a Shared Ethernet Adapter in the HA configuration (*Active/Passive*), rather than two virtual adapters supporting two VLANs each, we would have created one that supports all four VLANs. However, in the case of building an SEA in shared mode, it is necessary to split the VLANs into several adapters, so that each VIOS can work with some of these VLANs.

On the second VIOS, we create three adapters that will be detected as *ent1*, *ent2*, and *ent3*. The creation

of adapters on the HMC is presented in Figure 3-28.

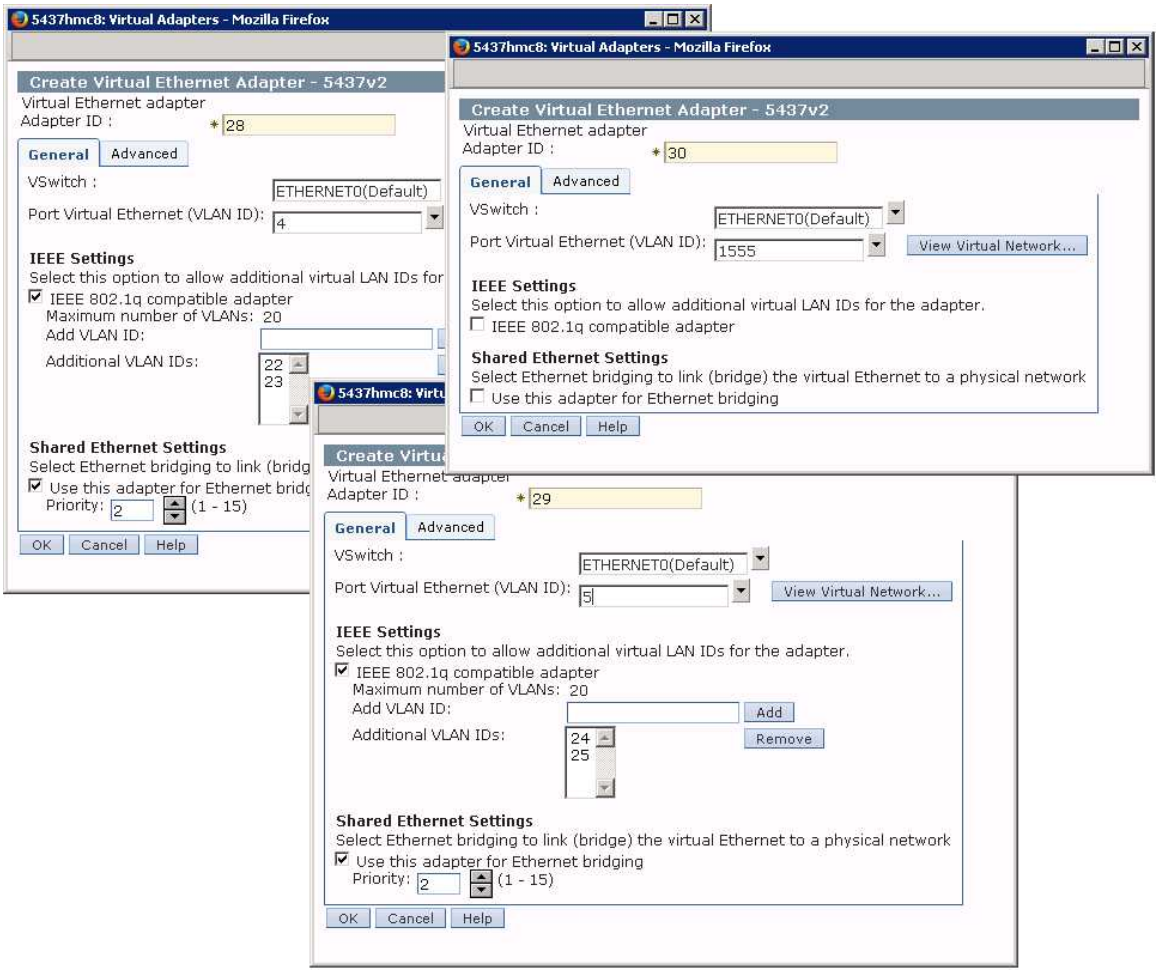

Figure 3-28: HMC - VIOS2 - Creating interfaces for an SEA.

- ***ent1*** - The adapter with ID 28 supports VLANs 22 and 23. It is equivalent to the *ent1* adapter on the first VIOS. The only difference is its priority of *2*. This means that the adapter on VIOS1 has a higher priority and will support network traffic to the first VLAN on the list, while VIOS2 will support network traffic to the second VLAN on the list. Both VIOS partitions will maintain backup paths in case of the failure of one of them.
- ***ent2*** - The adapter with ID 29 supports VLANs 24 and 25. It is the equivalent of the *ent2* adapter on the first VIOS, and the only difference is its priority of *2*.
- ***ent3*** - The adapter with ID 30 is a classic virtual adapter. It is in the unique VLAN 1555, which we chose to build a control channel between the SEA on VIOS1 and the SEA on VIOS2.

On the first VIOS, we create a *Shared Ethernet Adapter* in *sharing* mode:

```
# mkvdev -sea ent0 -vadapter ent1,ent2 -default ent1 -defaultid 4 -attr ha_mode=sharing
ctl_chan=ent3
```

On the second VIOS, we also create a *Shared Ethernet Adapter* in *sharing* mode:

```
# mkvdev -sea ent0 -vadapter ent1,ent2 -default ent1 -defaultid 4 -attr ha_mode=sharing
ctl_chan=ent3
```

The configuration of both VIOS partitions means creating a bridge between the physical adapter (*ent0*) and the virtual adapters (*ent1* and *ent2*). The *ent3* adapter in both cases is a control channel whose task is to diagnose the failure of one of the SEAs. In the event of such a failure, the second will take over. The *default* and *defaultid* values are required, although they are not relevant for SEA operations.

Default - Specifies the adapter through which untagged Ethernet frames are to be transmitted. Untagged frames are those that do not have a VLAN tag. Generally, we do not expect such frames and we do not want to handle them.

Defaultid - If an untagged Ethernet frame appears on the port, it will be tagged with the VLAN ID specified there. Generally, we do not want to handle untagged frames, so the configuration we create will ignore by redirecting to an empty VLAN.

The created configuration is presented in Figure 3-29.

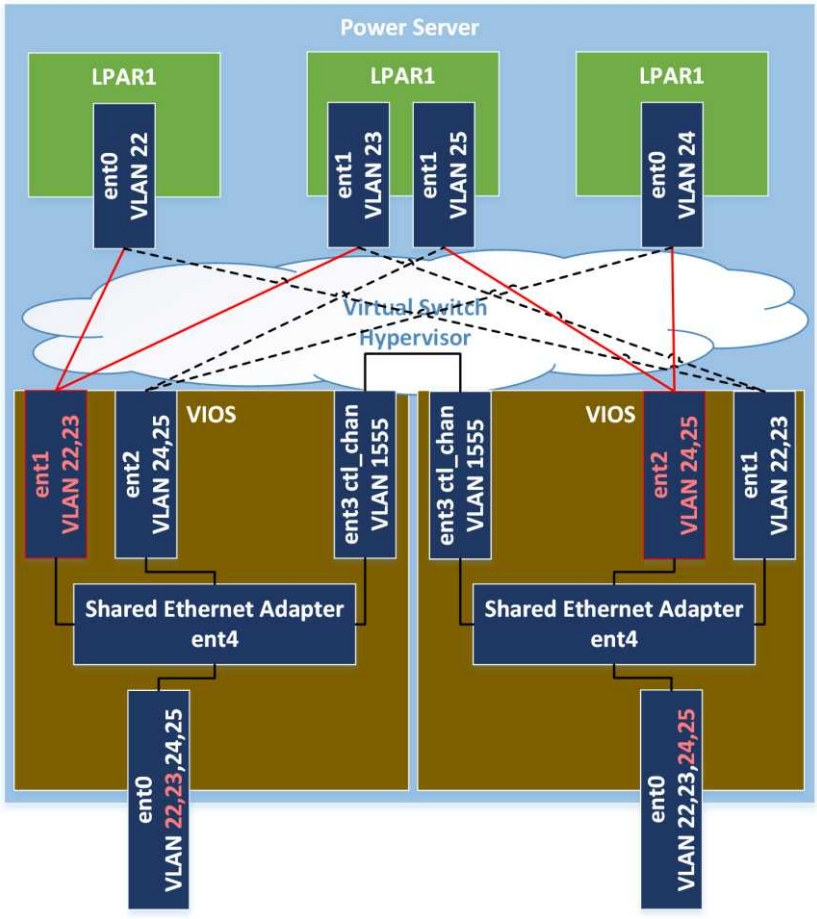

Figure 3-29: Shared Ethernet Adapter - Sharing mode.

Performance

A *Shared Ethernet Adapter* is a very convenient solution. It allows you to configure high availability and move part of the performance configuration to the Virtual I/O Server layer. As a result, from the point of view of the LPAR, you can have a single virtual network adapter and hence not worry about configuring the high availability. The simple configuration from the LPAR perspective is the advantage of a Shared Ethernet Adapter. However, SEAs do have some performance limitations.

An SEA's performance in most cases is sufficient for many applications. However, several facts should be considered:

1. **The problem with using the full speed of the physical adapter**

This is not a problem directly related to the SEA itself, but rather to the transmission performed by the virtual adapters. The transmission between two virtual adapters is not able to run at the full speed of 10 GB/s high-speed physical adapters, although along with newer servers, its speed is increasingly approaching those values. A hypervisor that supports internal transmission is responsible for this means of working. The transmission speed between virtual adapters can be increased by using large frames (by increasing the MTU parameter, by using Jumbo Frames on the Ethernet network). Using large frames, you can optimize the work of the hypervisor, since it is easier to handle a small number of large frames than a large number of small frames. Regardless of the size of the frame, the part of the work that the hypervisor must perform remains constant.

2. **Load of the VIOS processors**

Network traffic management via a *Shared Ethernet Adapter* is one of the most resource-intensive operations performed by the Virtual I/O Server. At the beginning of virtualization on the Power5 platform, the transmission of 1 GB by an SEA utilized the entire Power5 core. Since then, the hardware has accelerated many times, and the algorithms have been improved. However, you should continue keep in mind the possible CPU load.

When using an SEA and virtual networks, you should consider the following tips:

1. **Make sure that the systems are well placed and addressed**

Via the appropriate placement of systems on servers, you can highly optimize network transmissions. By placing many LPARs that communicate with each other on one server and in one VLAN, you will ensure low response times as well as reduce the workload of the SEA and the external network, since communication will not go outside the server. By placing two LPARs that communicate with each other on one server, but in other VLANs, you will cause them to communicate with each other through an external switch. It is possible that such network traffic will go in and out through the same interface.

2. **Use large frames**

Large Ethernet frames (*Jumbo Frames*) accelerate the transmission of large amounts of data, while still relieving the SEA. The number of frames processed has a significant impact on the SEA load. Processing the same amount of data in fewer frames will generate a lower VIOS load.

Jumbo Frames should be enabled at the level of the physical interfaces and the SEA itself. Regardless

of this parameter, it is also worth using large MTU values at the level of the LPARs themselves, which can significantly increase the transmission efficiency between virtual adapters (perhaps even several times over). It is also worth enabling the *largesend* parameter on the network interfaces and the Shared Ethernet Adapter. Enabling this option causes large frames (64 KB) to be transmitted between the virtual adapters on the server. The transmission to the SEA also takes place with a frame of this size. The SEA can offload the CPU by passing the task to the physical network adapter. The task in this case is to fragment the large frame into smaller ones in order to transmit them to the external network.

3. **Consider other types of network access**

It is safe to say that an SEA is suitable for 90% of applications. In some cases, especially for systems that use very high transmission speeds, you should consider the use of SR-IOV or a physical adapter. This solution will result in a lower server load (SEA) and increased transmission performance. The use of a physical adapter also allows you to transfer work, related to building and controlling the transmission, to the hardware layer of the adapter (so-called offloading), which relieves both the VIOS and the client LPAR.

LPAR

As mentioned at the beginning of this chapter, the LPAR has many synonyms, including logical partition, partition, micro-partition, and SPLPAR (*Shared Processor Logical partition*). Although some of them are used to symbolize a refreshed form of the approach to virtualization on the Power platform, colloquially, these names all mean the same.

The LPAR is a logical part of the server that consists of specific resources:

- **Processor resources** - Available in various modes and in the various conditions described above.
- **Memory** - Part of the server's memory. It is more or less virtualized depending on the chosen virtualization method.
- **Virtual adapters** - Allow access to the LAN, disk space, or tape drives.
- **Physical adapters** - A unique virtualization feature on the POWER platform that allows virtual systems to use physical hardware.

All of the above components were described in detail in individual sections of this chapter.

Below, we focus on the operations that can be performed on LPARs as well as on reading the LPAR parameters from the operating system level.

Creating an LPAR

There are different approaches to creating partitions. In organizations in which a lot of this type of work is performed, the automated approach is most commonly used. You can use dedicated tools such as PowerVC. You can also automate individual parts of the tasks by creating your own scripts. Even in

the case of the full automation of creating systems, it is sometimes worth going through the process manually, so as not to lose detailed knowledge about its steps and capabilities. Detailed knowledge is especially important if you run into problems and need to engage in troubleshooting.

The manual process for creating an LPAR based on the graphical form of the HMC interface is as follows:

From the console menu, choose the option to create an LPAR. In our case, this is an AIX/Linux LPAR. We also have the option of creating LPARs for the VIOS and IBM i. The first window that appears to us is displayed in Figure 3-30.

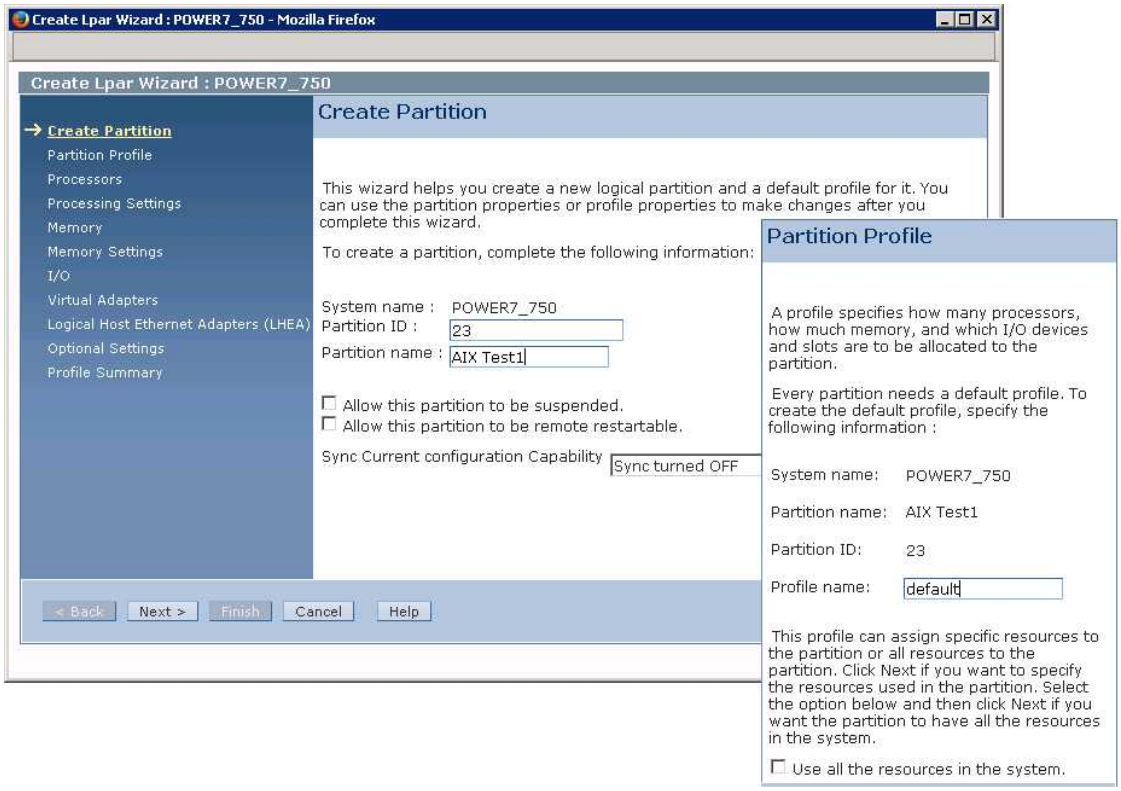

Figure 3-30: HMC - Creating an LPAR.

When creating an LPAR for AIX/Linux, we have several options to choose from in the first window:

- ***Partition ID*** - A unique identifier at the level of a single server. The console allocates the first free number. The ID number usually does not matter for the work of an LPAR.
- ***Partition name*** - The name given by the administrator. It is usually a good idea to use a transparent naming convention that will help you to identify the partition's functions.
- ***Allow this partition to be suspended*** - The option to freeze the LPAR. The function requires a proper VIOS configuration for operation. It should have a dedicated disk space for which it will be able to dump the memory of the frozen partition. The LPAR must also have the appropriate configuration, for example, it must be fully virtualized. There are also other restrictions on this feature.

- ***Allow this partition to be remote restartable*** - This option allows you to run the LPAR on another server, for example, in the event of a primary server failure. However, this option should not be considered a high availability solution. Such an operation has many restrictions regarding the configuration of the LPAR itself, as well as the state in which the source server is in. The total failure of the source server, including the unavailability of service processors, makes it impossible to run the LPAR on another server using this feature.
- ***Sync Current Configuration Capability*** - The option for maintaining consistency between the current LPAR configuration and its configuration as saved in the profile. A common problem when working with partitions is the inconsistency of these configurations. Such inconsistency often results from the fact that it is necessary to save changes to the profile after they have been implemented in the current configuration. If you do not take care of this action and the current configuration is divergent from the profile, then you can accidentally run the LPAR with a different configuration than that with which it was closed. By enabling synchronization, you ensure the consistency of the profile with the current configuration.
- ***Profile name*** - The profile name assigned by the administrator. The name usually does not matter, unless you want to use several configuration profiles for the LPAR. In this situation, their name should reflect the purpose of creation.

In the following steps, we configure the processor settings (Figure 3-31). We have two modes to choose from: *Shared* and *Dedicated*. The modes and their parameters are described in more detail in the chapter concerning processor virtualization.

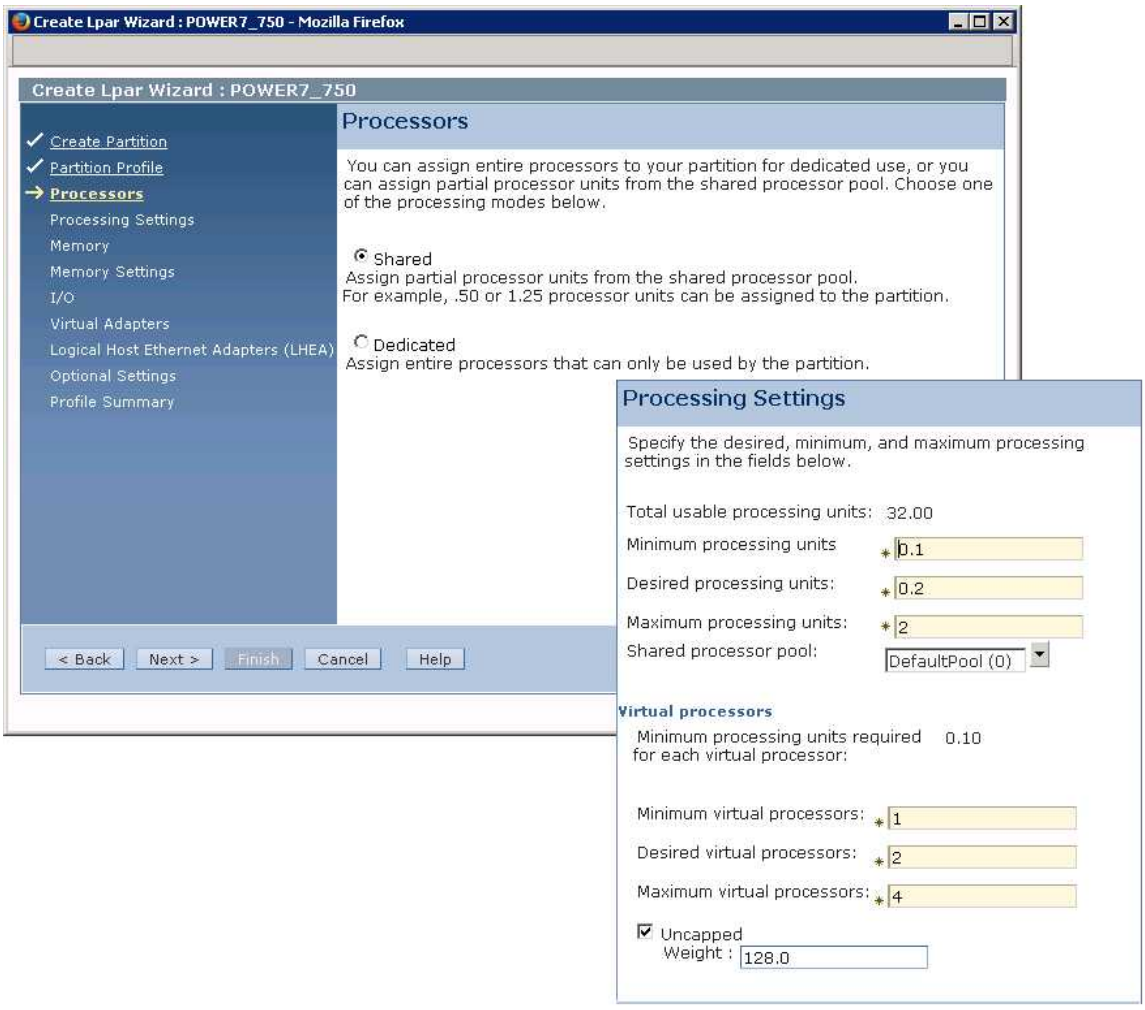

Figure 3-31: HMC - Creation of an LPAR - Processors.

The next steps concern the parameters related to memory (Figure 3-32). These settings, as in the case of the processor, come in two variants: *Shared* and *Dedicated*. The operating modes and their parameters are described in the chapter concerning memory virtualization.

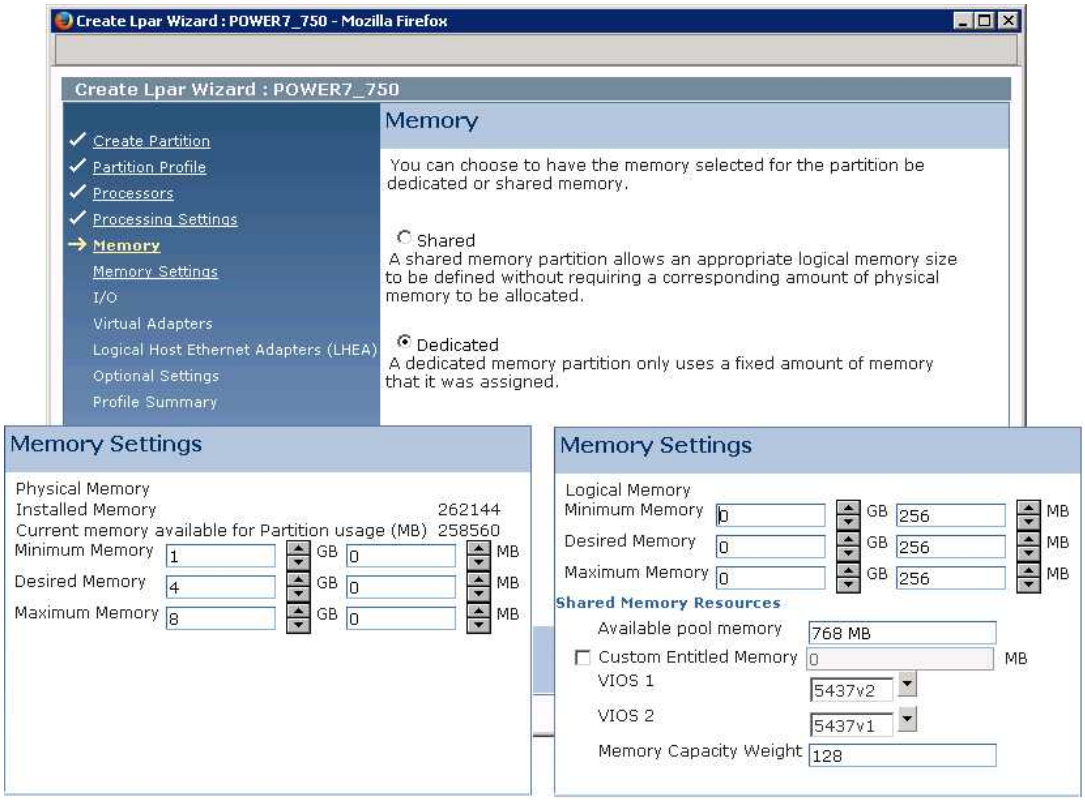

Figure 3-32: HMC - Creation of an LPAR - Memory.

The next step is to add physical components to the LPAR. Any adapters available on the server can be added to the configuration. However, you should remember that using physical components activates certain limitations in the LPAR operation. These limitations are the inability to use many advanced features, including the migration of the system "on the fly" to another server (*Live Partition Mobility*), the ability to freeze the system state and let it later resume its work, and the ability to use the *remote restart* function.

Regardless of whether or not physical components have been added, we create virtual adapters. There are four types of adapters available here:

- ***Serial Adapter*** - By default, there are two adapters of this type in the system. The first one is used to log into the system through a virtual terminal from the HMC. The second is used for internal LPAR communication with the server/hypervisor. You can also connect any two LPARs on the server through virtual serial adapters.
- ***Fiber Channel Adapter*** - A virtual device that allows access to SAN networks using NPIV technology. It is described in more detail in the section on storage virtualization. It requires the creation of a client adapter on the LPAR (the adapter is represented by the fcsX device) as well as the server adapter on the VIOS (the adapter is represented by the vfchostX device).
- ***SCSI Adapter*** - A virtual SCSI device that provides access to disks through the VIOS. It is described in more detail in the section on storage virtualization. It requires the creation of a client adapter on the LPAR (the adapter is represented by the vscsiX device) as well as the server adapter on the VIOS (the adapter is represented by the vhostX device).

- ***Ethernet Adapter*** - A virtual Ethernet adapter that allows network transmission inside the server via a hypervisor. Transmission to outside the server is performed via the Virtual I/O Server and the *Shared Ethernet Adapter* configured on it. For more information, see the section on network virtualization.

Figure 3-33 demonstrates how to create virtual SCSI client adapters and the virtual Ethernet adapter. The SCSI adapters point to two different VIOSs (*Server Partition* option in the figure) as well as to adapter number 155 (*Server Adapter ID* option) on each of the VIOS. The adapters on every VIOS must be created independently and then paired with client adapters. The virtual Ethernet adapter allows communication in VLAN 120, and it works on the default virtual switch. In terms of the adapter, you have several advanced options for managing the assigned MAC address:

- ***MAC Address*** - The option has an *Auto-Assigned* status by default. This implies automatic MAC address assignment. By selecting the *Override* option, you change the status to *User-Defined* and you can then assign your own MAC address to the adapter.
- ***Maximum Quality of Service (QoS)*** - This allows you to define the QoS for network traffic. Levels zero to seven are available, with level seven having the highest priority.
- ***Allow all O/S Defined MAC Addresses*** - The default value. The MAC address is automatically assigned to the adapter by the hypervisor. From the operating system level, you can change this address.
- ***Deny all O/S Defined MAC Addresses*** - In this case, you cannot change the MAC address that was automatically assigned.
- ***Specify Allowable O/S Defined MAC Address*** - This allows you to define up to four MAC addresses allowed on the interface.

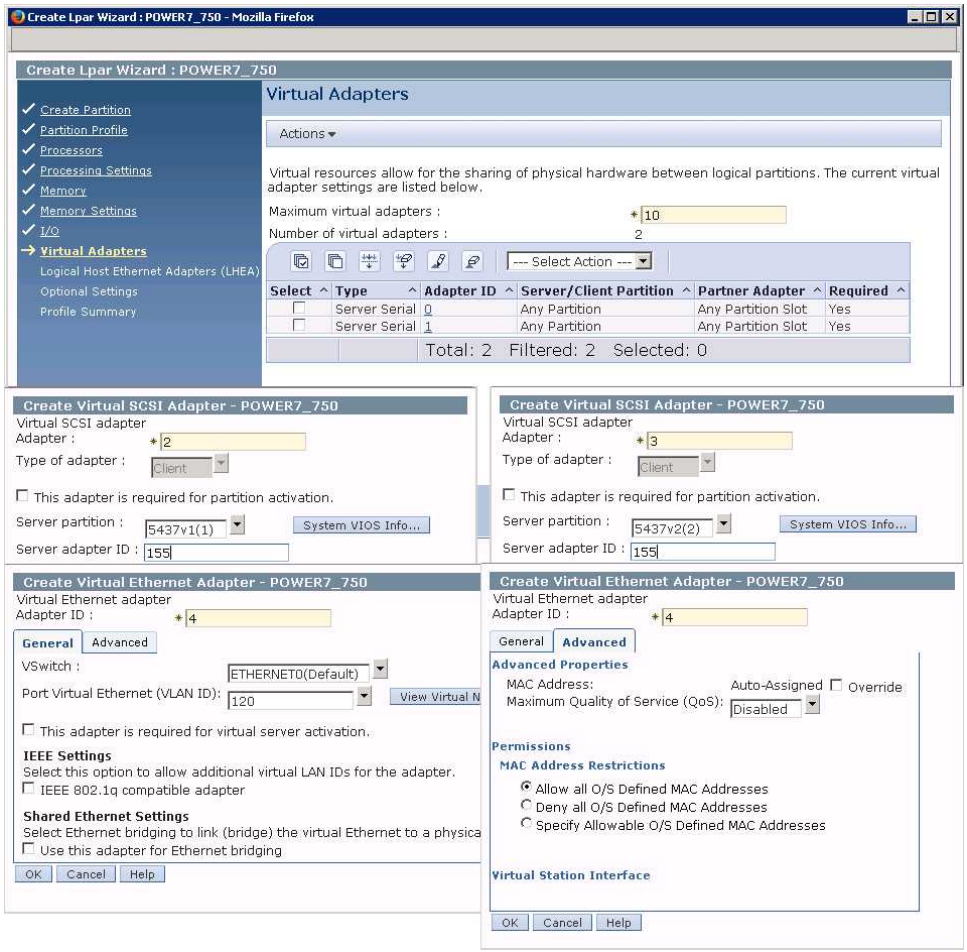

Figure 3-33: HMC - Creation of an LPAR - Virtual adapters.

During the next step, there are Host Ethernet Adapter configuration options available on some servers. Then, we have several optional settings, as shown in Figure 3-34:

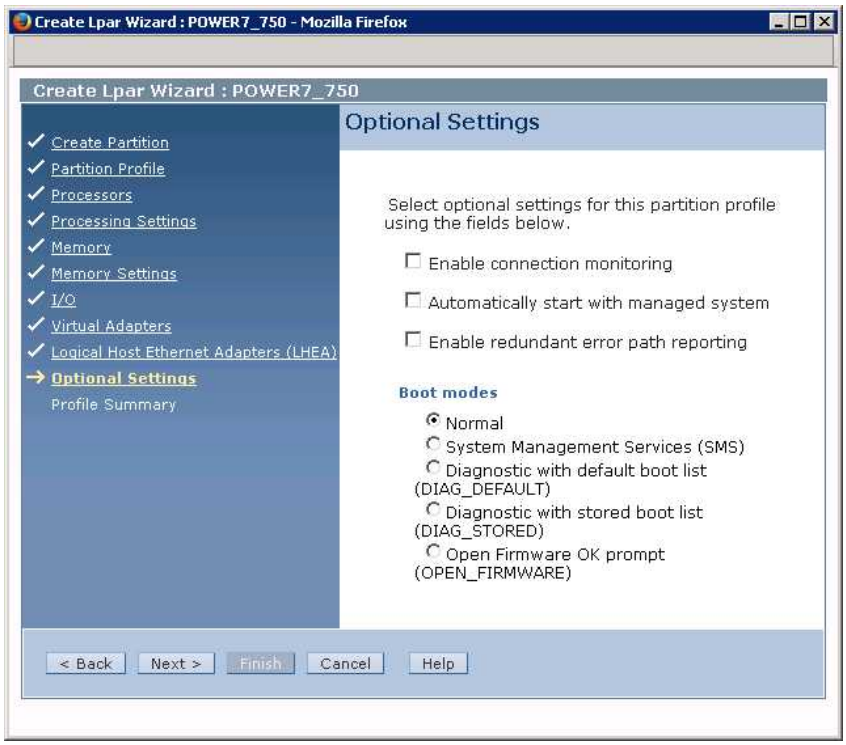

Figure 3-34: HMC - Creation of an LPAR - Optional settings.

- ***Enable connection monitoring*** - This option enables the monitoring of the connection between the HMC and the LPAR. If there are problems with the connection, an event is generated in the console log.
- ***Automatically start with manager system*** - With this option selected, the LPAR will automatically start after starting the server.
- ***Enable redundant error path reporting*** - The disabled option causes the LPAR to report to the console only those errors that affect it. When enabled, it causes the LPAR to report errors regarding the entire server to the console.
- ***Boot Modes*** - The mode in which the LPAR should be started by default. Usually, the *Normal* mode is used, unless it is necessary to carry out diagnostics or facilitate the installation of the operating system by stopping the LPAR activation in the *SMS* mode.

In the next step, you can verify your configuration and create an LPAR. The partition is ready to map the disks and install the operating system using the Network Installation Manager server or a classic physical or virtual media with the operating system.

Operations to be performed on an LPAR

All the defined LPAR parameters can be changed from the management console. Some of them can be changed "on the fly", that is, while the system is running, while others require the reading of a new profile. To read the new profile, you have to perform the following procedure: change the profile, close the LPAR, and start the LPAR from the modified profile.

The operations that can be carried out as well as the way in which they are performed are detailed in Table 3-9.

Table 3-9: Ways of changing the parameters of virtualization.

Operation	Way of Carrying Out	Comments
VCPU - add/remove	On the fly	On the fly, in the range between the *minimum* and *maximum virtual processors*.
CPU - add/remove	On the fly	On the fly, in the range between the *minimum* and *maximum processing units*.
Memory - add/remove	On the fly	On the fly, in the range between the *minimum* and *maximum memory*.
Change the *minimum/maximum* value	Change of profile	Changing any of the values specified as *maximum* and *minimum* requires changing the profile: closing the LPAR and then starting the LPAR from the modified profile.
Capped/Uncapped - switching modes	On the fly	
Weight - change of priority while in uncapped mode	On the fly	
Change *Shared/Dedicated* modes	Change of profile	
AME - turn on/off	Change of profile	AME (*Active Memory Expansion*) requires a system restart. During booting, the system creates dedicated structures for the AME.
AME - change the factor	On the fly	On the fly, when AME is switched on.
AMS – add/remove from poll	Change of profile	
Virtual SCSI Adapter - add/remove	On the fly	On the fly, removal after deconfiguration from the operating system.
Virtual Ethernet Adapter - add/remove	On the fly	On the fly, removal after deconfiguration from the operating system.
Virtual Fiber Channel Adapter - add/remove	On the fly	On the fly, removal after deconfiguration from the operating system.
Shared Processor Pool - add/remove CPU	On the fly	
LPAR - change the processor pool	On the fly	

Verification of the LPAR parameters from AIX

You can check the parameters of a given LPAR from the management console as well as from the operating system level. From the AIX system, you can do this using the *lparstat* command. An example of running this command, with a description of the most important parameters, follows:

```
# lparstat -i
Node Name                                  : AIX72c2
Partition Name                             : AIX Test1
Partition Number                           : 23   # ID LPAR-a.
Type                                       : Shared-SMT-4   # SMT is on, four threads per core.
Mode                                       : Uncapped   # Processor mode.
Entitled Capacity                          : 2.00   # CPU power guaranteed.
Partition Group-ID                         : 32790
Shared Pool ID                             : 0   # The LPAR is in the default pool.
Online Virtual CPUs                        : 2   # The system shows two cores in SMT-4 mode. Thus,
you will see eight threads from the operating system level.
Maximum Virtual CPUs                       : 2
Minimum Virtual CPUs                       : 1
Online Memory                              : 2048 MB
Maximum Memory                             : 2048 MB    # The maximum memory amount is equal to the
current amount. To add more to the LPAR, a reboot is required.
Minimum Memory                             : 256 MB
Variable Capacity Weight                   : 128
Minimum Capacity                           : 0.10
Maximum Capacity                           : 2.00
Capacity Increment                         : 0.01
Maximum Physical CPUs in system            : 32   # The number of processors in the server.
Active Physical CPUs in system             : 32   # The number of active processors. Some processors
can be activated later, for example, in the Capacity on Demand option.
Active CPUs in Pool                        : 28
Shared Physical CPUs in system             : 28   # The LPAR is in the default pool, the size of
which represents all the server's processor resources.
Maximum Capacity of Pool                   : 2800   # The size of the pool is expressed in 100 units
per one core. The power of the processor can be allocated with this gradation.
Entitled Capacity of Pool                  : 1610   # The processor power that is assigned to all
the LPARs in the pool in the Entitled Capacity parameter.
Unallocated Capacity                       : 0.00
Physical CPU Percentage                    : 100.00%
Unallocated Weight                         : 0
Memory Mode                                : Dedicated   # Memory allocation mode. Below are a
number of parameters regarding the allocation in Shared mode.
Total I/O Memory Entitlement               : -
Variable Memory Capacity Weight            : -
Memory Pool ID                             : -
Physical Memory in the Pool                : -
Hypervisor Page Size                       : -
Unallocated Variable Memory Capacity Weight: -
Unallocated I/O Memory entitlement         : -
Memory Group ID of LPAR                    : -
Desired Virtual CPUs                       : 2
Desired Memory                             : 2048 MB
Desired Variable Capacity Weight           : 128
Desired Capacity                           : 2.00
Target Memory Expansion Factor             : -   # Parameters related to memory compression.
Target Memory Expansion Size               : -
Power Saving Mode                          : Disabled
Sub Processor Mode                         : -
```

You can also use the *lparstat* command with other parameters for current LPAR monitoring.

Advanced PowerVM features - Live Partition Mobility

Live Partition Mobility (LPM) is a feature of the Power platform and PowerVM virtualization. It allows you to move partitions between servers. Similar feature exists with other virtualization providers, including Vmware (vMotion) or Microsoft (Hyper-V Live Migration).

The systems moved via Live Partition Mobility can be active and provide services (Active LPM) as well as disabled (Inactive LPM). Moving an active system between servers is transparent for both users and the services running there. People who are using the system during such a migration should not notice this operation in any way. LPARs moved between servers may contain the AIX, Linux, or IBM i operating systems. You can use this feature to achieve several goals:

- **Reduction of system unavailability** - The server sometimes requires some unavailability. You need to shut it down in order to install a new firmware. Sometimes, you must perform such an action due to the failure of one of its components. With LPM, there is no need to disable the system, since for the time needed for repair, you can migrate it to another server, and then migrate it back after doing the necessary work.
- **Reduction of the administrator's workload** - The periodic shutdown of the system can be a very burdensome operation, especially in the case of an organization with a lot of IT systems. The key issue here is not the technical complexity, but rather the need to coordinate such an operation between many business units. In some organizations, shutting down one system can trigger an event chain in the form of having to shut down and restart subsequent systems. This renders it necessary to make many arrangements as well as to provide support by many groups of administrators. The LPM operation, being transparent to the user, allows the reduction of the number of system restarts required and thus reduces the workload within the organization.
- **Load balancing between servers** - The load on individual systems tends to increase steadily. The use of LPM allows the load to be distributed between individual servers, thereby balancing the load on the entire environment.
- **Easy migration to new servers** - The unique feature of the Power platform is the ability to handle LPARs in compatibility mode with an earlier POWER processor. When you buy a new server with a new processor, you have the option of migrating LPARs to it without having to shut them down. Such LPARs will operate in the mode of compatibility with the older processor model until you decide to restart them and change the operating mode.
- **Energy saving** - With the ability to transfer systems without unavailability, you can periodically consolidate them on fewer servers and then shut down the unnecessary servers. You can do so during periods characterized by a lower load on the platform, such as holidays, weekends, or nights.

The migration process

The system migration can be initiated from the HMC from both the GUI and the command line. Multiple migrations can be carried out simultaneously, and the current values for Power8 servers are 16 simultaneous migrations per server and eight per VIOS. Migration is similar in both the active and inactive modes, except for one key phase. This phase involves migrating the current LPAR state to another server. To perform the migration, you need:

- A management console: HMC or IVM. It is worth noting that it is not possible to migrate between systems managed by consoles of various types. The consoles should also be trusted in relation to each other (using an SSH keys exchange).
- Virtual I/O Servers on the source and target servers, which should be marked as "mover service partition." The VIOS plays a key role in synchronizing the LPAR state between the servers. The VIOS partitions on the source and destination sides must have access to the same disk resources assigned to the LPAR. It is possible to migrate in configurations with one and two VIOS LPARs per server.
- A fully virtualized LPAR that does not have any physical components.

The migration process itself can be divided into several phases:

- **Validation** - Verification of the available resources on the server side (physical resources, such as memory, processor, etc.). Verification of the firmware version, processor type, etc. Validation is performed on the operating system and on the VIOS. The target VIOS must support the VLANs used by the source VIOS. The target VIOS must also have access to the same disk resources as the source. If the disks are mapped by VSCSI, they must be visible to both VIOSs with the same parameters. A common reason for a validation failure is the incompatibility of the disk configurations in the context of parameters that are rarely noticed (for example, the *max_transfer* parameter).
- **Creating a new LPAR** - A new LPAR is created on the target server. The new LPAR is the same as the old one, including the device numbers. The numbering of devices is usually irrelevant to the operation of the system. However, it is important for administrators, who it helps to be better oriented within the environment. Important parameters, such as the MAC addresses, are moved to the interfaces of the new LPAR. An important change occurs when the LPAR is connected to the SAN using NPIV. When creating a *Virtual Fiber Channel* adapter, we defined two WWN addresses, one of which was active and the other inactive. The newly created LPAR will have the same two addresses, but in reverse order. Both LPARs (old and new) will have simultaneous access to the same disks for a short period of time. An example of WWN addressing is given in Figure 3-35.

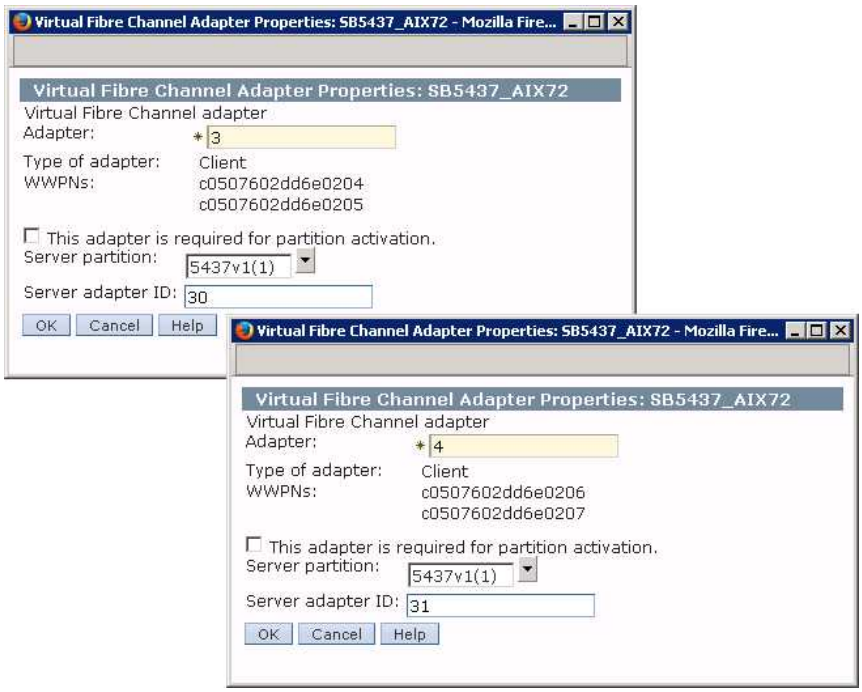

Figure 3-35: HMC - Double WWN addresses.

- **Creation of resources for the LPAR** - On the VIOS side (or VIOSs - if we migrate the LPAR in a configuration with two VIOSs), appropriate resources are created to support the LPAR. These are virtual resources that reflect the configuration on the source VIOS side, for example, *vhost* (support for VSCSI disks) and *vfchost* (support for NPIV disks). Wherever possible, the names of the devices are preserved.

- **LPAR state migration** - This phase only occurs when the active LPAR is migrated. The source Virtual I/O Server with the "*mover service partition*" status sends all the information about the status of the transferred LPAR to the destination VIOS over the LAN, such as:
    - The memory contents of partition;
    - The processor status;
    - The state of virtual adapters; and
    - Other information.

  The transferred LPAR still works while rewriting its state, and it still changes part of its memory. Any changes made by the LPAR are also recorded and synchronized with the destination server until full synchronization. This mode of operation requires efficient network communication between the VIOS partitions. An inefficient network may lead to a situation in which the LPAR will be quicker at changing the contents of its memory than the VIOS at sending it over the LAN. In this case, the migration process will have no chance of success. Another important aspect concerns the security of data sent between the VIOS partitions. You can guarantee security by building a secure tunnel between the Virtual I/O Servers.

  After synchronizing the LPAR state, the source LPAR is frozen and the work is switched to the new LPAR. This activity is usually unnoticeable for applications or users; however, by

enabling advanced monitoring, you are able to capture the short-term extension of the LPAR response times.

- **Removal of the old LPAR and release of resources** - Subsequent phases involve cleaning the configuration on the source server. The old LPAR and its devices on the VIOS servers are removed.

The main requirements for an LPM

When compared with the operation of the generally understood functions of Live Migration in other popular virtualizers that operate on the x86 platform, an LPM may appear burdened by many limitations. If you take a closer look at those limitations, you may notice that they are caused by the wider configuration capabilities of the PowerVM environment. By creating very common virtual system configurations, you will usually not encounter limitations.

The main conditions that need to be met in order to use *Live Partition Mobility* (*Active Migration*) are described below:

- **License** - The LPM option is available in the enterprise version of PowerVM.

- **Fully virtual LPAR** - To carry out the migration in the *active* mode, the LPAR must be fully virtual.

- **Its disks must be mapped by VSCSI or NPIV**. It must use an SEA to communicate with the external network. You can, of course, migrate LPARs that use HEA adapters, but only if those adapters are paired with a virtual Ethernet adapter. The result of this migration is the unconfiguration of the HEA adapter from the pair, and the further operation of the operating system on the virtual adapter. The latest change is the ability to migrate LPARs that use SR-IOV, provided that they do not use SR-IOV directly, but instead use it through a virtual vNIC adapter.

- **Servers' compatibility** - The servers between which migration occurs must be compatible with one another:
    - The *Logical Memory Block Size* parameter specifies the minimum size of the memory unit that can be allocated to the LPAR. It must be the same on both servers in order to perform the LPM operation in the *active* mode. This is important because changing this parameter at a later point leads to the need to restart the entire server.
    - Compatibility at the firmware level must be maintained. The dependencies are not particularly restrictive, but if you have servers that differ in terms of the firmware version for many years, then you can expect problems. Prior to migration, verify the compatibility matrix: *Firmware support matrix for partition mobility*.
    - You can migrate between servers with Power 6, 7, and 8 processors. Migration to a server with a newer version of the processor does not usually cause problems. After the migration, the LPAR runs in compatibility mode with the older version of the CPU. Migrating from a newer version of the processor to an older version will only succeed if the LPAR is operating in processor compatibility mode supported by the older server. When running LPARs on servers, it is important to be aware of whether

and to which servers they will be migrated to run them in the appropriate compatibility mode.

- **Communication** - The management console, Virtual I/O Servers, and LPAR participate in the migration process. To run the migration, it is necessary to:
  - Ensure communication via the *Resource Monitoring and Control* (RMC) subsystem between the console and all the participants in the process. By default, this communication exists because it is needed to manage systems through the console, although it can be unstable, for example, due to network problems.
  - Ensure efficient communication with the Virtual I/O Server that participates in the migration process.
  - Ensure a trusted configuration between the consoles (if the servers between which communication occurs are managed by different consoles). Trusted communication takes place via the SSH protocol, and it requires the prior exchange of keys between consoles.

- **Access to the same resources through the VIOSs** - The Virtual I/O Servers on both servers must provide access to the same resources:
  - Provide communication via an SEA for the VLANs in which the LPAR is located.
  - Provide access to the same disk devices configured in the same way when the LPAR works through VSCSI.
  - Provide the option to run additional NPIV adapters when the LPAR uses this technology.

Additionally, the VIOS partitions that are to perform the LPAR state transfer operation must be marked as *mover service partition*, as shown in Figure 3-36.

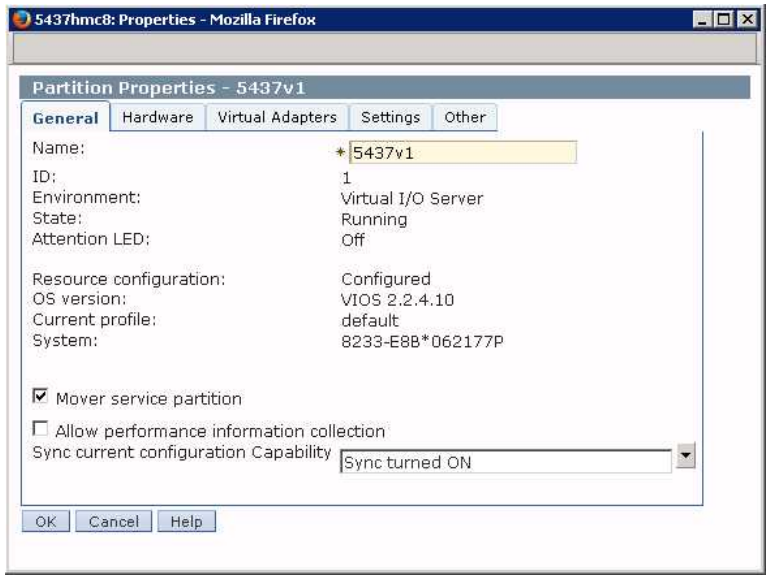

Figure 3-36: HMC - Mover service partition.

Before you start your first migration, it is a good idea to read the relevant documentation. It is also worth carrying out the validation from the HMC. The validation process checks the dependencies in

the context of the defined migration and lists any potential incompatibilities. An example of the validation process is shown in Figure 3-37.

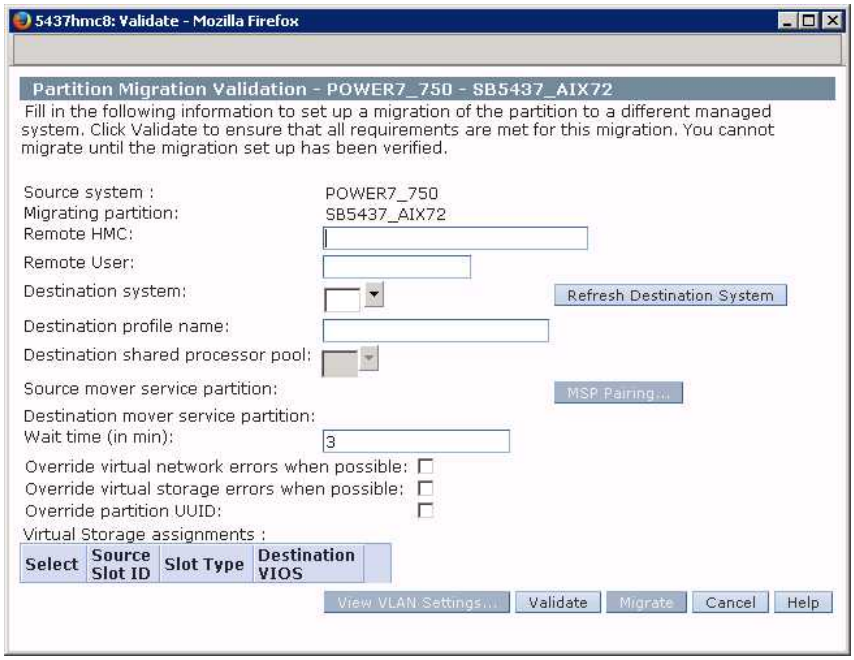

Figure 3-37: HMC - Partition migration validation.

In the case of LPAR migration in *inactive* mode, some of these requirements do not have to be fulfilled.

Suspend and resume

The operation of suspending or freezing the system is very popular with many types of virtualizers. It is most commonly used in the case of the virtualization of personal computers. Then, you typically need to run a virtual machine with a different operating system or other functionality. The most effective method in this regard is the resumption of a previously suspended system.

Suspending a partition in a POWER platform involves saving its current state in the disk space and then releasing the resources that it uses. The released resources are the operational memory and the processor computing power. All the resources used by the suspended partitions can be used for any purpose. The partition is suspended even after the entire server has been restarted, and it can be resumed at any time.

The LPAR is resumed from the moment at which it was suspended. This is made possible by restoring the operating memory, the state of the virtual processors and devices, and using data previously stored in the disk space. All the launched applications resume their operation at exactly the point where they were suspended, in most cases without being aware of the fact that they have been suspended. Nevertheless, the suspension can cause some problems in a large environment where many systems work together. Such problems include the expiration of sessions between systems, which may lead to difficulties in subsequent communication.

Purpose of use and requirements

This feature can be used in several typical situations:

- **Maintenance** - Maintenance may necessitate stopping the server. In this case, you can suspend the systems running on the server and then resume them after performing the necessary actions. The systems will not be available, but they will resume operation faster and you will not have to restart the entire application.

- **Reducing the server load and required licenses** - Each system has its own work cycle. Test system groups represent a special case. There may be many of them within a company, but at the same time only one or a few of them actually work. In this case, the non-working systems can be suspended and, as a result, a number of systems will be permanently shut down. Hence, you need a smaller number of servers to support the environment. In addition, systems with the same licensing profile can run in the *Shared Processor Pool*. If a number of systems are permanently suspended, fewer processors and licenses for the product will be needed.

- **Energy management** - The suspension of some systems lets you temporarily shut down some servers. It provides financial savings and reduces the impact on the environment.

- **Faster migration (LPM)** - You can speed up the migration of systems with a large amount of memory by suspending them first. Thanks to this, the LPAR will not modify the memory during the migration, and it will not generate additional network transmissions.

There are a number of basic requirements that need to be met in order to use the *suspend/resume* feature:

- You must have a PowerVM virtualization license in the standard or enterprise version.

- The partition should have the feature enabled. The relevant option is *Allow this partition to be suspended*. It should only contain virtual devices. This is the same requirement as that for using *Live Partition Mobility*.

- The VIOS must have a dedicated disk space for storing the LPAR state, which is known as the *Reserved Storage Device Pool*. For each suspended LPAR, there should be a separate disk or logical volume in the pool equal to the memory of the partition to be *suspended/resumed*, plus 10% for additional data.

- The operating system, console, and VIOS must all be in versions that support this feature. As the feature is not new, all relatively modern installations should satisfy this condition.

Activity scenarios

Creation of a Reserved Storage Device Pool

In the scenario below, we want to create a *Reserved Storage Device Pool* in order to use the previously unused *suspend/resume* feature. We want this pool to be served by two VIOS partitions.

As a first step, we mask a single LUN to two VIOS partitions and then detect it as hdisk6. In the next step, we create a *Reserved Storage Device Pool* (Figure 3-38):

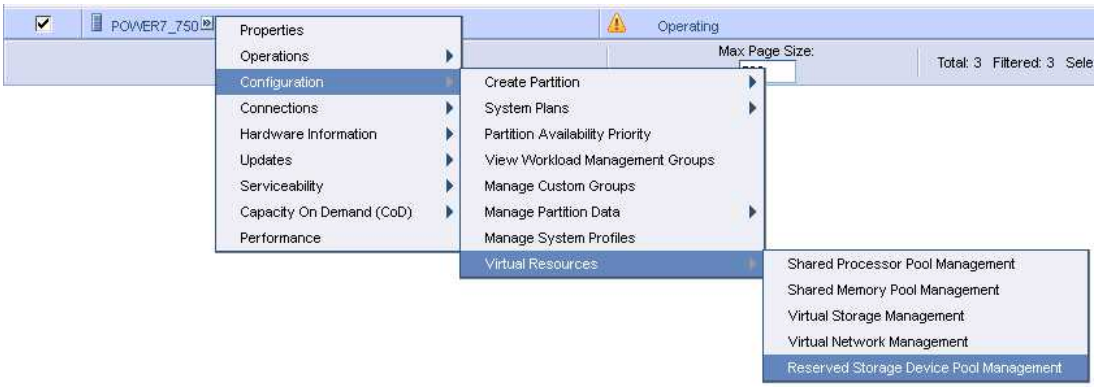

Figure 3-38: HMC - Creation of a Reserved Storage Device Pool.

A window appears in which you can select the devices (disks) that will be assigned to the pool (Figure 3-39):

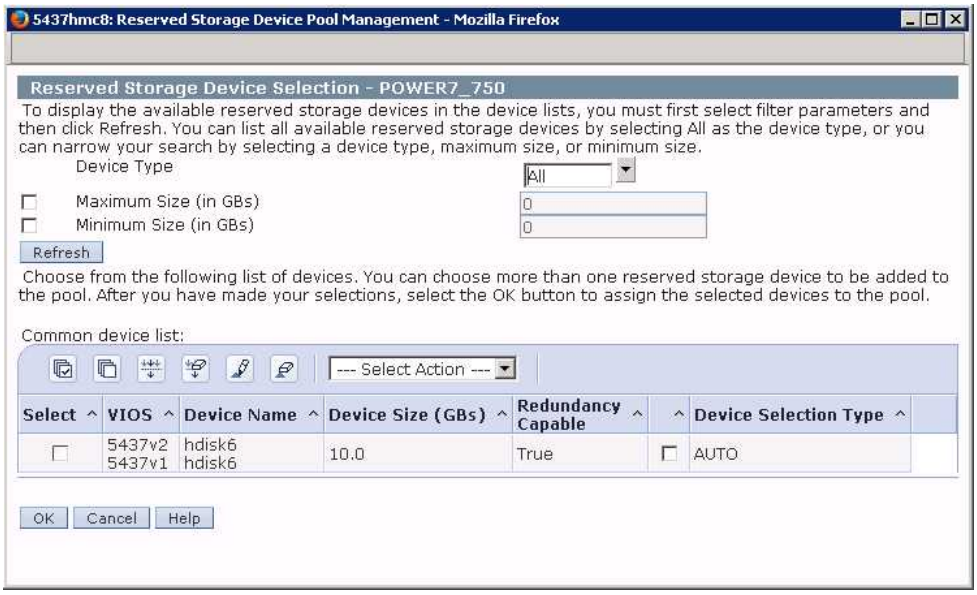

Figure 3-39: HMC - Creation of a Reserved Storage Device Pool (continued).

After selecting the disk, we create a *Reserved Storage Device Pool*, and we can use the *suspend/resume* feature for a single LPAR with a memory of less than 10 GB. If you want to suspend a larger number of LPARs, then you must add the appropriate number of disks, one per LPAR, with a size that reflects the memory size of the given LPAR + 10%.

LPAR suspension

We decided to suspend the LPAR. Before doing so, however, we need to change its configuration in order to allow such an operation. In the partition properties menu, select the option *Allow this partition to be suspended* (Figure 3-40):

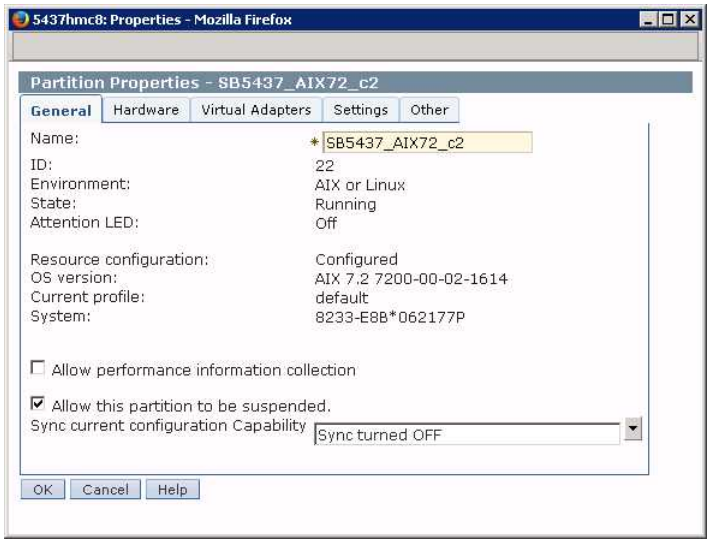

Figure 3-40: HMC - Allow this partition to be suspended.

From the partition menu, we start the *suspend* action (Figure 3-41):

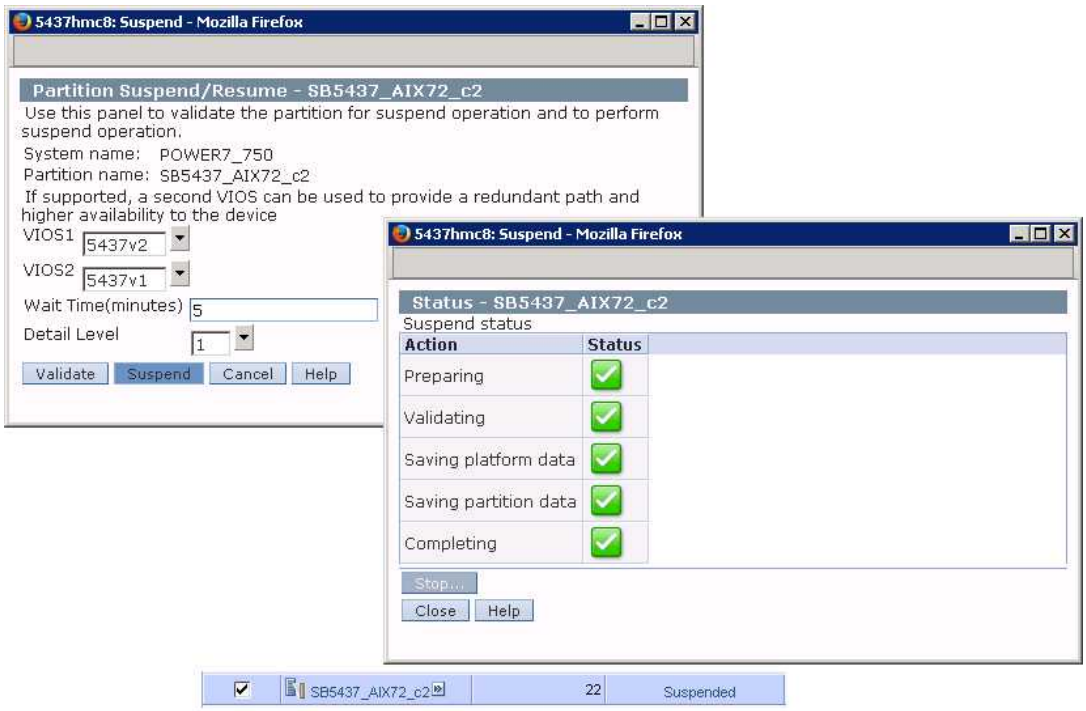

Figure 3-41: HMC - Partition suspended.

We can choose the VIOS that will handle the operation. In our case, the disk is detected on both VIOSs and we have a redundancy. If one of the VIOS partitions is unavailable, we still have the option of resuming the partition. After pressing the *Suspend* button, the work will be suspended after all the process phases marked in green have been actioned. The partition is suspended, and it will remain in this state after restarting the entire server.

Resuming a partition requires only the selection of the *resume* option from the menu.

Dynamic Platform Optimizer (DPO)

Large servers that support multiple processors have a complicated structure. They consist of many sockets, each of which can consist of many cores. Each socket has memory banks assigned to it. The entire server's memory is seen by all the sockets of a given server; however, the time required to access the memory may be different in different cases. The processor that operates on the memory that is closest to it will process more efficiently than the one that operates on the memory assigned to another socket. This situation is depicted in Figure 3-42.

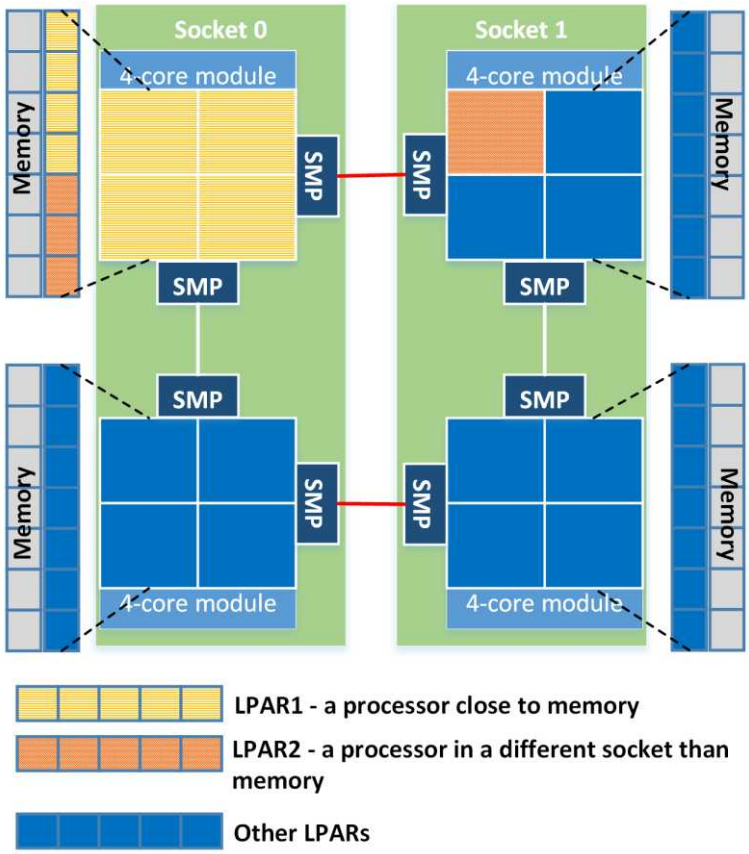

Figure 3-42: Dynamic Platform Optimizer - CPU and memory.

When creating new LPARs or when adding resources to an existing LPAR, the hypervisor ensures that the distribution of resources on the server is optimal. However, this is not always possible. In the case of the frequent deletion and creation of LPARs, the memory can be fragmented. When creating a new LPAR, the hypervisor may have no way out and hence allocate the memory on many sockets. A similar situation may occur in the case of expanding the LPARs' memory. It is clear that a well-utilized server on which many operations were made to create, delete, and resize LPARs does not work optimally.

In the past, you could fix this problem in only one way, namely by shutting down all the partitions on the server, deleting the information about the resources assigned to them, and running them in order starting from the most important one. Starting the LPARs reallocated the physical memory to each of them. Unfortunately, constant work on the server (i.e., creating, deleting, and changing LPAR resources) caused the server to transition to a non-optimal work state. The remedy for this is found in the feature known as *Dynamic Platform Optimizer*. This feature makes it possible to improve the allocation of memory and CPU resources for individual LPARs without the need to shut them down. This usually involves rewriting the memory contents of some LPARs to a different location, which allows the processors to work on the memory in the closest neighborhood. An example of this optimization process is presented in Figure 3-43.

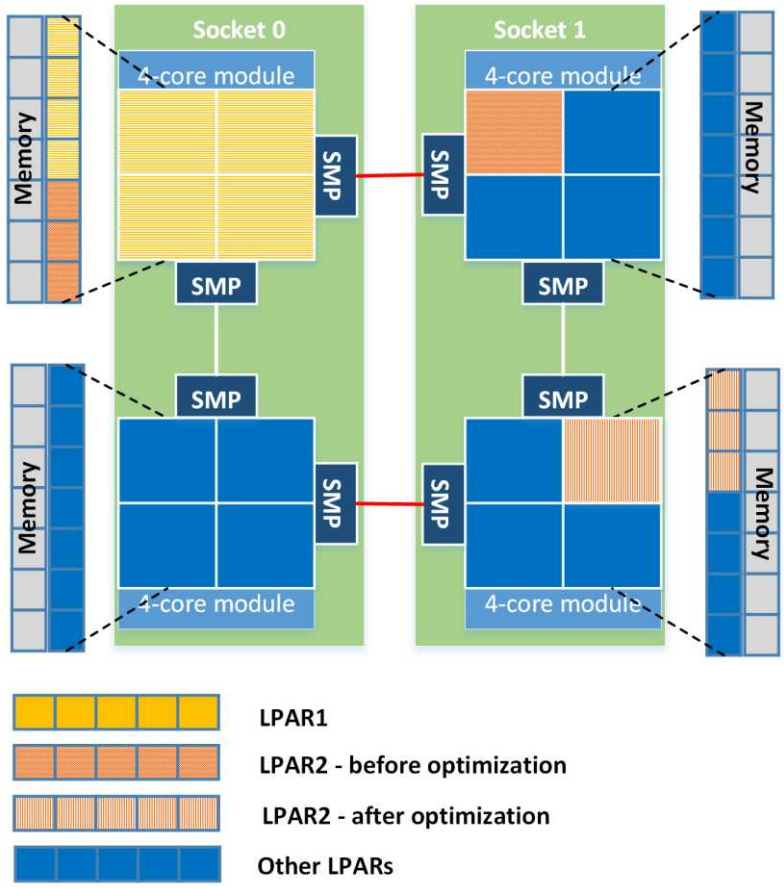

Figure 3-43: Dynamic Platform Optimizer - Before and after optimization.

When using the DPO, make sure that the operating systems affected by this feature supports it. If the systems does not support the DPO's functionality, the change may actually degrade performance. This is because the operating system itself is trying to optimize its work in terms of the resources it has available. If the system does not support the DPO, it will not realize that there has been a change, and it will use outdated information to optimize its work. This situation can be remedied by restarting the given LPAR, which will refresh the information concerning the platform. You can also exclude this type of LPAR from the optimization process.

The machine's firmware is responsible for the optimization process itself. Operations in the context of the DPO are started from the HMC using the following commands.

- *lsmemopt* - Calculation of profits in the context of optimization as well as verification of the current status.

- *optmem* - Start or stop the optimization of a given server. Optimization does not have to be performed at the entire server level. Some LPARs can be excluded from this process.

When you plan to use this feature, check the requirements of the server firmware version, the HMC, or the operating system version. Since the operations require the copying of the memory inside the server,

they should be planned during a period of small load.

Dynamic System Optimizer (DSO)

The operating system is fully aware of the hardware on which it operates, its architecture, and the resources that have been allocated to it. This represents a major advantage of PowerVM virtualization over competitive x86 architecture solutions. Due to this awareness, the operating system can optimize its own work. Optimization concerns many aspects, and it can even enable efficient system operation with large memory fragmentation. The method of its operation is best illustrated using a simple example, as shown in Figure 3-44.

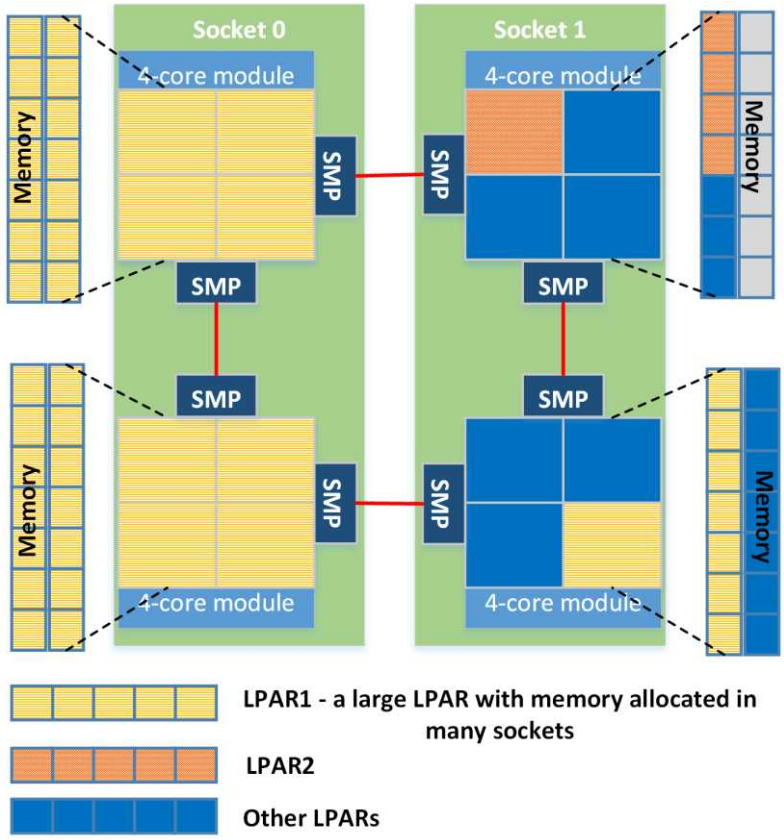

Figure 3-44: LPAR placed on several sockets.

In the above figure, we have a system (LPAR1) whose memory is distributed on the server, and it is located in the neighborhood of several different sockets. The operating system will try to optimize its work in such a way that the processes operating on the memory closest to *socket 0* will be run on the processors located in *socket 0*, while the processes that operate on the memory closest to *socket 1* will be run on the processors located in *socket 1*.

Some optimizations, such as the one described above, work automatically without our awareness. However, it is worth noting the functionality of the *Dynamic System Optimizer*. It is disabled by default.

This functionality may affect how the system works in the context of the server's cache memory.

The DSO subsystem is responsible for the operation of this feature:

```
# lssrc -s aso
Subsystem         Group            PID          Status
 aso                               4587920      active
```

To manage this feature, you can use the *asoo* command, which has several configuration options.

```
# asoo -FL
NAME                      CUR    DEF    BOOT   MIN    MAX    UNIT           TYPE
     DEPENDENCIES
--------------------------------------------------------------------------------
aso_active                0      0      0      0      1      boolean          D
--------------------------------------------------------------------------------
debug_level               -1     -1     -1     -1     9      numeric          D
--------------------------------------------------------------------------------
##Restricted tunables
--------------------------------------------------------------------------------
aggressive_cache_affinity 1      1      1      0      1      boolean          D
--------------------------------------------------------------------------------
aggressive_cache_opt_utilisation
                          1000   1000   1000   1      2000   1/1000th cores   D
--------------------------------------------------------------------------------
allow_fp_placement        1      1      1      0      1      boolean          D
--------------------------------------------------------------------------------
allow_sub_srad_placement  1      1      1      0      1      boolean          D
--------------------------------------------------------------------------------
cache_affinity            1      1      1      0      1      boolean          D
# All unnecessary information has been omitted.
```

There are a number of basic optimizations that you can use when using the DSO:

- **Cache Affinity** - This moves the threads of a particular processing closer to each other, thereby focusing them in a specific processing domain, for example, one corresponding to neighboring cores. This can increase the number of hits in the cache and thus speed up the processing. Optimization brings good results where the processing is multithreaded. The affinity cache can also work in aggressive mode. This involves the consolidation of processing on fewer cores so as to achieve greater cache accuracy. An aggressive mode can be used when the system statistics indicate that this approach will be beneficial. In practice, it may be more about limiting the processing to a given socket. If a single socket has, for example, eight cores and the processing is done on nine, it means more frequent reaching to the memory located at another socket and thus reducing the hits in the cache. In this case, processing on eight cores within one socket can prove more efficient than processing on nine cores in two sockets.

- **Memory Affinity** - The functionality can work if the processing is optimized by using the *cache affinity*. It moves the hot pages of the process's private memory to the processing domain. This action takes place when the algorithm that manages this functionality "finds" that such a move can bring benefits to the processing. This only applies to multithreaded processing.

- **Large Page Optimization** - This allows the system to automatically use 16 MB memory pages. This feature is particularly useful for software that operates on large memory areas, such as large databases. Memory pages with a size of 16 MB may have been used before, but this required the explicit allocation of a certain number of pages. This caused some resources to be

wasted, and it could cause dangerous system behavior. If you declared a specific number of 16 MB pages, but the data buffers (e.g., databases) did not fit there, then 4 KB and 64 KB pages were allocated. This could result in the use of the paging space as well as slowing down the system while the allocated 16 MB pages were unused.

Software licensing in the PowerVM environment

PowerVM and AIX represent an environment that enables you to run almost any type of software. The software often has a commercial character, so it requires a license. The available licensing methods differ, and they may concern the number of users, parallel sessions, cores, sockets, or other units invented for the needs of a given product.

By examining the licensing models of different software vendors, you can make one fundamental mistake. This mistake lies in the fact that users are guided by logic, not by facts about the policy of a given product. It should be noted that licensing does not have to be logical. It is a marketing model of a given vendor, which is intended to achieve the required financial impact and enable a "fight" against competing products. Therefore, it is not a permanent model. The vendor can dynamically change it, thereby optimizing profits or increasing the competitiveness of the product.

The different ways of licensing and optimizing the use of licenses will be discussed based on the example of two significant vendors, namely IBM and Oracle. The following guidelines should be confirmed with the software vendor prior to use, since the licensing model can change dynamically, while some software vendors leave space for interpretation.

IBM - PVU licensing

IBM uses many licensing methods depending on the product. In this chapter, we will focus on PVU (*Processor Value Unit*) licensing. A large part of IBM's software is subject to licensing of this type, for example, software from the WebSphere family, databases (DB2, Informix), and many others.

PVU is concerned with assigning an appropriate value to a particular type of core. In the case of the Power platform, the values range from 70 PVU per core for small servers to 120 PVU per core for the largest and most efficient servers. A very useful feature of this licensing method is the full understanding of virtualization on the Power platform. It supports the licensing of processor fractions (*sub-capacity*) and a *Shared Processor Pool*, and it allows you to use the Live Partition Mobility functionality without any restrictions.

If you use IBM software on virtual platforms, you must use the license control software - ILMT (IBM License Metric Tool). This software is provided free of charge by IBM. The ILMT client must be installed on all virtual machines that use IBM licenses and are subject to PVU licensing. As users, we are required to create a report about the utilization of licenses once a quarter and maintain such reports for two years in the event of an audit. The need to use this software is an additional duty, but correct operating procedures provide full control over the utilization of a license and limit the potential for interpretation by the vendor. The rules for the use of licenses are clear, and as a customer you see their utilization. If you notice any unfavorable changes to the use of the license, you can immediately respond.

In order to optimize the use of licenses within organizations that have multiple installations with the same software, it is worth following a few rules:

1. **Install similar software sets on all LPARs**. In addition, wherever possible, distribute different software to different partitions. This approach opens up opportunities to optimize the use of licenses by applying the methods detailed in the following points.

2. **Use the uncapped type configuration**. Only this configuration allows the optimal use of all server CPU resources. In addition, it allows you to optimize the use of the licenses of software installed there, if you combine its effects with the information contained in point three below. It is worth using this mode because of the typical performance characteristics of systems. Normally, no system uses the maximum CPU power assigned to it and, even if it does, it uses the maximum power for only a limited period of time. Thanks to such working characteristics, a given server with production systems can have at least twice as many virtual as physical processors without a significant negative impact on performance. Test servers have this factor to a much higher degree. Such an approach opens up the possibilities of optimizing the utilization of licenses.

3. **Group the LPARs with specific software**. When the LPARs are divided in terms of the installed software, you can group them on dedicated servers or in a *Shared Processor Pool*, thereby limiting the number of licenses needed.

4. **Consider what server model to use**. Depending on the server model, the number of license units needed for a single core can range from 70 to 120. It is therefore worth choosing servers for specific solutions.

Examples of calculating the number of necessary licenses and their optimization

In the following examples, we assume that all the LPARs have the same IBM product or products licensed under the PVU rules.

1. CPU fractions

The following example (Table 3-10) shows several LPARs configured on one server. Each of the LPARs has a fraction of the processor computing power and operates in *Capped* mode. A single partition is not able to use more than the indicated fraction, while the maximum power used by all the LPARs is one core. In total, all the LPARs can use one CPU, so the number of licenses needed to cover one core ranges from 70 to 120 PVU depending on the type of server.

Table 3-10: Licensing, IBM, Capped mode.

	Virtual Processors	Processing Units	
	Desired	Desired	Cap/Uncap
LPAR1	1	0.3	*Capped*
LPAR2	1	0.3	*Capped*
LPAR3	1	0.4	*Capped*
Suma	***3***	***1.0***	

Demand for licenses: One core (70–120 PVU).
Tips: Try to group these types of systems on a single server.

The next example (Table 3-11) shows the same LPARs but operating in uncapped mode. In this case, each LPAR can use the entire core, so together they can utilize up to three cores; hence, that number of licenses is needed.

Table 3-11: Licensing, IBM, Uncapped mode.

	Virtual Processors	Processing Units	
	Desired	Desired	Cap/Uncap
LPAR1	1	0.3	*Uncapped*
LPAR2	1	0.3	*Uncapped*
LPAR3	1	0.4	*Uncapped*
Suma	***3***	***1.0***	

Demand for licenses: Three cores.
Tips: In this case, you can consider using the Shared Processor Pool.

2. Shared Processor Pool

The following example (Table 3-12) shows six LPARs with the same software. In order to optimize the use of the licenses, the partitions were assigned to a processor pool with a size of 13 CPU, which means that the same number of licenses is needed. Despite limiting the power sum of all the LPARs to 13 cores, each partition can individually use such power as indicated by the number of virtual processors owned.

Table 3-12: Licensing, IBM, Shared Processor Pool.

	Virtual Processors	Processing Units		
	Desired	Desired	Cap/Uncap	Pool
LPAR1	2	1.2	*Uncapped*	
LPAR2	4	2	*Uncapped*	
LPAR3	6	4	*Uncapped*	
LPAR4	3	1	*Uncapped*	***13 CPU***
LPAR5	4	2	*Uncapped*	
LPAR6	4	2.5	*Uncapped*	
Suma	***23***	***12.7***		

Demand for licenses: 13 cores.
Tips: If you do not want to limit the power of the LPARs, then you should determine the size of the pool based on the load data of all the partitions in the pool, while remembering to leave a margin of safety. Another method is to close the LPARs in a large pool and then gradually decrease it based on the observation of its load.

3. **Live Partition Mobility**

There are no special restrictions when using this functionality. The maximum number of processors that a given LPAR utilizes at a given time is important. A good and viable solution is to have dedicated processor pools on two servers, between which the LPARs can migrate. When considering the large scale, these can be dedicated servers.

Oracle - Core Processor Licensing Factor

Oracle, like other software vendors, has many methods for licensing its products. Even for a single product, the licensing method depends on the edition of the software and its purpose. The case of development applications is different, as is the case for small and large companies. For large organizations, licenses are often associated with a specified number of cores. The number of cores in this case should not be taken literally, since they are converted using the *Core Processor Licensing Factor*. The Power platform has the least favorable factor in the case of Oracle software. At the time of writing this book, the factor is one, while on the x86 and SPARC platform it is usually 0.5. Despite the unfavorable license factor on the Power platform, you can benefit from using Oracle software due to the high performance of the platform itself, as well as the good consolidation methods.

The verification of the utilized Oracle licenses on the Power platform is not easy. It involves two aspects:

1. The license policy for the Power platform is unclear. It leaves some room for interpretation. Therefore, some virtual configurations regarding Oracle software should be agreed with the software vendor prior to purchase.
2. In the case of IBM, we use the ILMT tool, which, despite its inconvenience, allows us to determine the status and control our licenses. In the case of Oracle software, the situation is more difficult, since Oracle does not provide such a tool.

To optimize the use of Oracle licenses on the POWER platform, similar rules apply as for IBM software. The rules are described in more detail in the section above concerning IBM, although a reminder is given below along with an indication of the differences:

1. Install similar software sets on the LPARs.
2. Use the *Uncapped* type configuration.
3. Group the LPARs with specific software.
4. Use the highest performance server models. Oracle licenses the software on the Power platform uniformly, regardless of the server performance. Thus, more efficient servers will allow you to use fewer licenses.

Examples of calculating the number of necessary licenses and their optimization

In the following examples, we assume that all the LPARs have the same Oracle product licensed under the *Core Processor Licensing Factor*. It is recommended that more complicated configurations are agreed with the vendor due to the imprecise nature of the license.

1. Uncapped mode

Similar to other vendors, Oracle understands the Uncapped configuration. The three LPARs in the example below (Table 3-13) will utilize licenses for three cores regardless of the other LPAR parameters.

Table 3-13: Licensing, Oracle, Uncapped mode.

	Virtual Processors			Processing Units				
	Min.	**Desired**	**Max.**	**Min.**	**Desired**	**Max.**	**Cap/Uncap**	**Weight**
LPAR1	1	*1*	4	0.1	*0.5*	4	*Uncapped*	64
LPAR2	1	*1*	4	0.1	*0.5*	4	*Uncapped*	64
LPAR3	1	*1*	4	0.1	*0.5*	4	*Uncapped*	128
		3			***1.5***			

Demand for licenses: Three cores.
Tips: In this case, you can consider using the *Shared Processor Pool*.

2. Shared Processor Pool

We can use processor pools to optimize the number of licenses required. In the following case (Table 3-14), we have four LPARs with four active cores. We will need a license for eight cores due to the assignment to a processor pool of this size.

Table 3-14: Licensing, Oracle, Shared Processor Pool.

	Virtual Processors			Processing Units					
	Min.	**Desired**	**Max.**	**Min.**	**Desired**	**Max.**	**Cap/Uncap**	**Weight**	**Pool**
LPAR1	1	*4*	4	0.1	*1*	4	*Uncapped*	64	***8 CPU***
LPAR2	1	*4*	4	0.1	*1*	4	*Uncapped*	64	
LPAR3	1	*4*	4	0.1	*1*	4	*Uncapped*	128	
LPAR4	1	*4*	4	0.1	*1*	4	*Uncapped*	128	
		16			***4***				

Demand for licenses: Eight cores.
Tips: If you do not want to limit the power of the LPARs, then the size of the pool should be determined based on the load data of all the partitions in the pool, while remembering to leave a safety margin. Another method is to close the LPARs in a large pool and then gradually decrease it based on the observation of its load.

3. Live Partition Mobility

The use of the mechanisms known as *Live Migration* is very expensive in the case of Oracle software. In terms of Power servers, the use of this mechanism forces you to cover all the source and destination servers with licenses, regardless of the size of the migrated LPAR. When you have two 64-core servers and migrate between them one LPAR with one virtual processor, you need a license for 128 cores. Migration between processor pools will not help here, since they do not change the licensing approach. This approach may seem to be devoid of logic, but as I explained earlier, licensing has goals other than logic.

Chapter 4. Installation in Physical and Virtual Environments

The only servers we can install AIX on are servers from the IBM Power Systems family. These servers use IBM POWER processors, so AIX works on uniform hardware manufactured by one manufacturer. This situation has its good and bad sides. On the one hand, we are doomed to the monopoly of one supplier, while on the other hand, the close co-operation of the operating system with the hardware brings the benefits of a high-stability solution and the full use of server performance.

You can install AIX in several ways, for example, using installation DVDs, tape backups, or over the network – using the *Network Installation Management* (NIM). Another relatively new method is to use the PowerVC environment based on OpenStack and install from the system images prepared there. In larger environments, the use of PowerVC is becoming increasingly popular. It is to be expected that this will become a necessary solution for organizations using IBM Power Systems.

This chapter describes the traditional means of installation using operating system media. Nevertheless, it is worth mentioning the other installation methods:

- **Network Installation Manager (NIM)** - This allows you to install the operating system on non-DVD servers. It also allows simultaneous system installation on multiple servers. To use this method, you must first configure one of your AIX systems to be the Network Installation Manager server. The great advantage of this method is the ability to automate the installation process. We traditionally go through several screens, providing the necessary parameters each time. For installations using the Network Installation Manager, we can skip those steps. This requires prior preparation of a parametric file describing the selected options. We are also able to include a script to run after the installation of the operating system, which provides us with very wide customization possibilities. With the NIM, we can install the system from previously prepared images (i.e., backups created using the *mksysb* command). This approach allows the early preparation of the operating system image in accordance with the standards in the organization and thus saves the work done after each installation.
- **Installation from system backup media** - This is very similar to installing from AIX DVDs. The process of creating a system backup (*mksysb* command) builds a structure on the target media that allows it to run a reduced version of the system. After booting from such media, you can restore the backup or install the new system. Installation may be performed on other servers than that on which the backup was made. To perform such an installation, one more condition must be met. The backup should come from the operating system installed with the *Enable System Backups* option set to *install any system*. This option puts all the possible drivers for the devices that may exist on the server into the system. This has the additional advantage that typically we do not need to install additional drivers when we modify hardware resources, since they are all already on the system. Another advantage is that the operating system can be properly prepared prior to the backup so that it can be used to install new systems. It can have all the necessary settings according to the standards of the organization, which greatly reduces the configuration work that must be performed after the fresh installation of the system.
- **PowerVC** - Installation in this case looks more like the cloning of the prepared system image. This is a very effective method, since it is automatic and reduces the workload compared to the above two methods. After properly preparing the system, such an activity does not need to be performed by the administrator or even a human. PowerVC provides a REST API that can be used to control it from higher layers, for example, from an orchestrator.

All three of the above-mentioned methods may use system images for installation. If you want to use images, you must usually prepare them first. To prepare them for the first time, you need to perform a classic system installation from the installation media.

Types of installation

AIX, whether installed from DVDs or from a system backup, provides three types of installations for different purposes:

- **New and Complete Overwrite Installation**

This destroys all the data stored on the system drives (drives selected for the system installation). It is mainly used for installing new servers, although it is also used to completely refresh the system.

It is important to keep in mind that if the server has separate system disks and separate data disks located in different volume groups, the information on the data disks can be easily and quickly made available back on the system (*importvg* command). The disk structures will be discussed in later chapters.

- **Migration Installation**

This type of installation is used to upgrade from an older version of the operating system to a newer version. It is selected by default when running the server from a media that contains a more recent version of the system than that installed on the server.

Migration keeps all the disk structures and installed software unchanged, since only files in the */tmp* directory are deleted. In the case of migration, some system settings cannot be preserved due to changes in subsequent system versions. In this case, we will be notified of any changes that have taken place after migration.

- **Preservation Installation**

This type of installation is used when you need to reinstall the system, but at the same time need to preserve the user data located in the *rootvg* volume group. This is a complete system installation that does not destroy the file systems intended to store user files, but instead destroys all the file systems intended to store the operating system files. A list of the deleted and saved file systems is presented in Table 4-1 below:

Table 4-1: Preservation Installation – File systems.

Removed file systems	Preserved file systems
/(root)	/home
/usr	/opt
/var	All other file systems and logical volumes created after the system installation
/tmp	

Prior to this type of installation, you can specify the files in the removable file systems that should be preserved. You can use the */etc/preserve.list* file to list the files. By default, AIX preserves */etc/filesystems*, */etc/swapspaces*, */etc/resolv.conf*, and */var/adm/ras/raspertune*. The names of these files are listed in */etc/preserve.list*. This file can be extended before installation with additional files that should be preserved.

After this type of installation, you need to configure the system: that is, create appropriate users, user groups, and configure the resources. Typically, you also need to reinstall the applications running on the server, or at least configure them from the operating system.

Boot order

Be sure to set the appropriate boot order, which is the order in which the server checks the device from which it can boot the operating system. There are two ways to do this:

- **Changing the boot order using the *bootlist* command** - The approach to use if the system is already installed, but you need to reinstall or migrate to a newer version.

  Before making the change, you can check the current boot order:

```
# bootlist -m normal -o
hdisk0 blv=hd5 pathid=0
cd0
```

In the above example, "normal" boot mode is selected, which the server uses by default. There is also a "service" mode whose boot order checks and changes in the same way using the keyword "service" rather than the keyword "normal." You can force the server to start using the service mode by pressing the *6* key while it is starting.

If the order is not appropriate, you can change it using the following command:

```
# bootlist -m normal cd0 hdisk0
```

- **Changing the boot order via SMS (*System Management Services*) mode** - You can access this mode from the console just before the startup process for the operating system. Below is a brief description of how to change the boot order in this way.

1. Enter SMS mode: Start the server and wait for the welcome screen to appear on the console. The machine will start to write the words "*memory keyboard network scsi speaker*." At this point, press the *1* key to enter the menu (Figure 4-1):

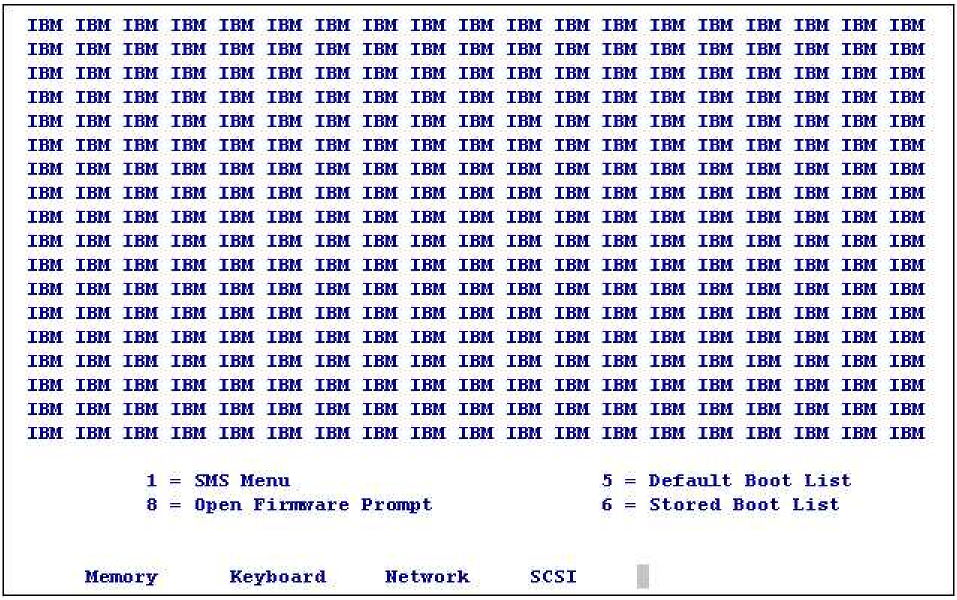

Figure 4-1: Enter SMS mode.

2. In SMS mode: Select "*Select Boot Options*" and then "*Select Install/Boot Device*." The next menu is intuitive enough that it does not require a detailed description. It is worth mentioning that from this level, we can also choose to boot from the network using BOOTP. In this way, we can preconfigure the system, providing information about the IP address of the system and setting the IP address of the NIM server.

3. After leaving SMS mode, the server will start the system from the first drive on the modified boot list. For a basic system installation, this is usually a DVD drive. Once the installation program has run, you can perform the installation.

Sometimes, the system may not want to run from a DVD. The media may be defective. In addition to the damaged media, the likely cause of such behavior may be an old installation version of the system that does not have the necessary drivers for the resources of the (usually new) server model. The term "old version" does not apply in this case to the AIX version, but rather to the specific installation DVD. It may also be necessary to update your server's firmware in order to install a newer version of your system on older hardware.

Installation process

The server running from the installation DVD reads a limited version of the kernel that supports the rest of the installation process. The first interaction of the system with the installer concerns the questions of the default console and the language that the system should use during installation. After answering these questions, the main configuration window is available, which includes five options (Figure 4-2):

- **Start Install Now with Default Settings** - When this option is selected, the default settings are displayed. Confirm the settings or return to the previous menu by entering "*99*."
- **Change/Show Installation Settings and Install** - Allows you to change the parameters of the installed system. Since this is the most commonly used option, this will be discussed further later in this chapter.
- **Start Maintenance Mode for System Recovery** - An option that provides extensive system repair capabilities. This will be covered in greater detail in the backup recovery section.
- **Make Additional Disks Available** - Allows you to configure disks when using iSCSI or release the SCSI reservation on a specific drive.
- **Select Storage Adapters** - Select the adapter to which the disks you are installing the system on are connected to.

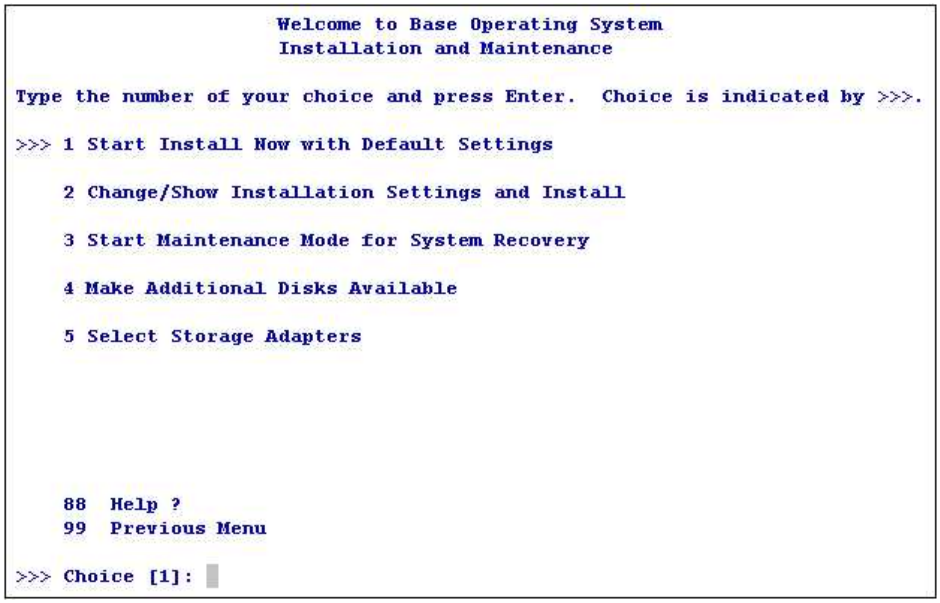

Figure 4-2: Operating system installation - Main screen.

Change/Show Installation Settings and Install

When you select this option, the following screen for selecting the installation parameters appears on the console (Figure 4-3).

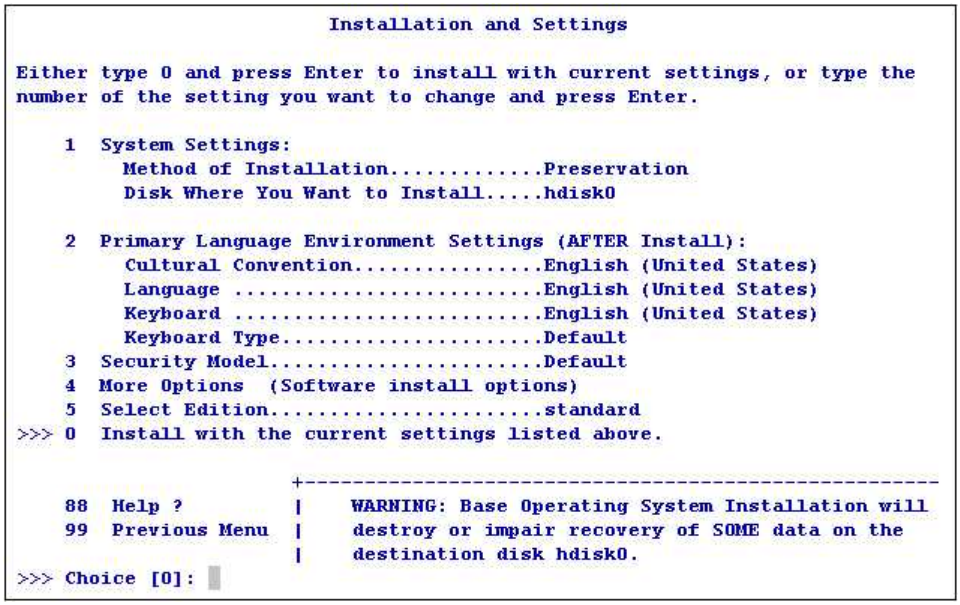

Figure 4-3: Operating system installation - Settings.

Meaning of individual parameters:

- **Method of Installation** - Specifies the type of installation to be performed. The installation types and their meaning were described earlier in this chapter.
- **Disk Where You Want to Install** - Allows you to select the disks on which you want to install the operating system. You can select multiple disks to install it on. In that case, the system will be distributed among the disks. Although this will ensure an even distribution of load between the selected disks, it does not provide any protection against failure. Therefore, you usually choose to install on one disk, and then when it is finished, you create a mirror, if necessary.
- **Primary Language Environment Settings (AFTER Install)** - The parameters of this option apply to the language settings. Choosing English is often a better choice than selecting a different language. Messages presented in a language other than English may be more difficult to interpret and less readable. In addition, when using English, it is easier to work with a vendor support line or use the associated documentation. The above parameters can be changed after the system installation.
- **Security Model** - Allows you to choose additional security options, such as *Trusted AIX*, *BAS and EAL4 + Configuration Install*, *LAS / EAL4 + Configuration Install*, and *Secure by Default*. The options are discussed in the Security chapter.
- **Select Edition** - Allows you to choose the operating system edition. The system is available in standard and enterprise editions. The basic functionality of the system does not differ between the editions, but the enterprise version includes additional licenses on a broad spectrum of

external tools, such as PowerVC, Tivoli Monitoring, and IBM BigFix Lifecycle. The selected edition can be changed during system operation using the *chedition* command or the *smitty editions* menu. The main purpose of this option is to point to the version of the system for the *IBM License Metric Tool (ILMT)* used to measure and report on the IBM license usage.

- **More Options** - This option provides some additional installation parameters, including:
  - **Enable System Backups to Install any system** - The system installed with the option set to *yes* will contain all the possible drivers for the known system devices. Due to this, from the system backup made on this server, you will be able to install the system on another server. This installation is performed in the same way as when installing from installation media. Refer to the Backup chapter for more information.
  - **OpenSSH Client Software and OpenSSH Server Software** - Installs filesets with OpenSSH. Usually, we want to use secure methods of accessing the system, so we choose this option.
  - **Graphics Software** - Installs the graphical environment.
  - **System Management Client Software** - Installs Java software and additional tools.

The contents of the packages selected from the installation menu can be thoroughly verified from the installed system. These packages are grouped into a so-called bundle. By running *smitty install_bundle* and selecting the appropriate package, you can see what kind of files it contains. We can find such packages as:

- **AllDevicesKernels** - Used when you select *Enable System Backups to install any system*.
- **"openssh_client" and "openssh_server"** - Used when you select *OpenSSH Client Software* and *OpenSSH Server Software*.
- **Graphics** - Used when you select *Graphics Software*.
- **SystemMgmtClient** - Used when you select *System Management Client Software*.

Installation Assistant

Once installed, the system automatically starts the tool for its basic configuration. This is known as the *Installation Assistant*, which is shown in Figure 4-4.

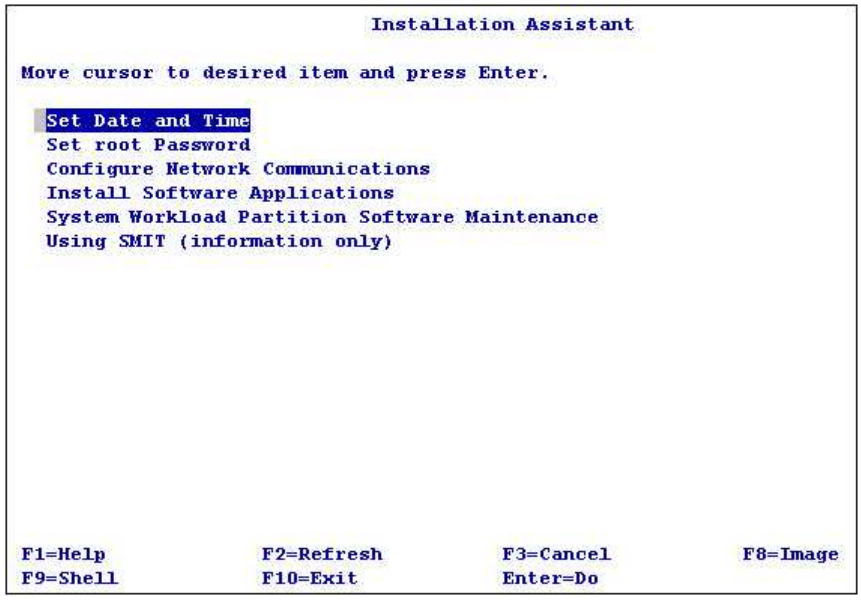

Figure 4-4: Installation Assistant.

The *Installation Assistant* is part of SMIT. It can be started at any time from the operating system using the *smit assist* command. The *Installation Assistant* allows you to quickly configure basic system parameters such as:

- The date, time, and time zone;
- The root user password; and
- The IP address and basic network parameters.

The tool also allows you to install additional applications and manage WPARs (Workload Partitions).

Once you have installed the operating system, you can find a *rootvg* volume group, a series of logical volumes and the file systems located on them. The list of file systems and logical volumes is shown in Table 4-2.

Table 4-2: System status after installation.

Logical Volume Name	Type	File System
hd1	jfs2	/home
hd2	jfs2	/usr
hd3	jfs2	/tmp
hd4	jfs2	/
hd5	boot	
hd6	paging	
hd8	jfs2log	
hd9var	jfs2	/var
hd10op	jfs2	/opt
lg_dumplv	sysdump	
livedump	jfs2	/var/adm/ras/livedump

The above terms, for example, the volume group and the logical volume, will be explained later in this book.

Chapter 5. Maintenance of the Operating System

The maintenance of the operating system relies heavily on patch installations, system component updates, and the installation of additional components in response to demand.

In order to be able to describe these processes clearly and comprehensively, we should familiarize ourselves with the concepts associated with them:

- **Fileset** - The smallest unit that can be installed on AIX. It contains all the files needed to accomplish specific tasks. It is usually the whole product or a functionally separate part of the product. Sample filesets:
  - *bos.rte.cron* - Files needed by a *cron* tool to work.
  - *bos.net.nfs.client* - Files needed by an NFS client to work.
- **Bundle (Bundle of multiple filesets)** - Many filesets collected together into larger groups based on their role. Sample bundles:
  - *Alt_Disk_Install* - A collection of filesets, which contains all the tools needed to clone the system.
  - *App-Dev* - A collection of filesets, which contains the filesets needed by developers working in the AIX operating system.
- **PMR (Problem Management Record)** - A problem reported to IBM technical support. Such a problem is assigned an appropriate identifier and priority, a record is created for it, and work on its solution is begun. The result of the work on the problem can be an APAR.
- **APAR (Authorized Program Analysis Report)** - Report generated due to work on the reported problem (PMR). If IBM support confirms that the problem encountered by the customer lies in the AIX layer, an APAR is created. For a given APAR, a workaround, Interim Fix, or a new version of the fileset can be developed. If the same problem occurs within several Technology Levels, several APARs will be created.
- **iFixes (Interim Fixes)** - Temporary fixes for critical issues that require a quick response. IBM tests these types of patches, although it does not go through full regression testing.
- **Security iFixes** - A special subset of temporary fixes that include security fixes. In this case, the patches also do not go through full regression testing, so they can potentially negatively affect stability. Nevertheless, due to the very limited number of changes in such iFixes, there is only a limited risk of instability.
- **Technology Level (TL)** - A large package of patches that can support new hardware and software enhancements in the operating system. This kind of package is issued once a year. Each TL can be patched by installing another Service Pack. IBM releases patches to several TLs simultaneously, so installing current patches does not necessarily change the TL of the system. Prior to the introduction of the definition of "Technology Level," a similar package was known as the "Maintenance Level" (ML). The ML shortcut still appears in certain commands.
- **Service Pack (SP)** - A patch package that is installed in a specific TL that may support new hardware. An SP is usually published twice a year for each supported TL. The patches that are grouped within the SP should have only a minimal impact on system functionality.

Information regarding the installed version of AIX along with the TL and SP versions can be obtained by using the *oslevel* command:

```
# oslevel -s
7200-00-02-1614
```

Meaning of individual parts:

- *7200* - AIX version, in this case 7.2.
- *00* - Technology Level, in this case 0.
- *02* - Service Pack, in this case 2.
- *1614* - Build Sequence Identifier, which indicates the year and week of publication. In this case, 2016 and the 14th week.

Installing and updating filesets

Both updating the system and installing additional components can be performed in a traditional or a safe way, which allows you to roll back your changes.

- **Safe way** - Old files are stored in */usr/lpp* when new filesets are loaded. The filesets added or updated in this way have a status of APPLIED in the system. The only issue in this case is the use of large amounts of space in the */usr* file system for old files. To overcome this issue after a specified period of time, newly uploaded or updated filesets must be either committed or rejected. The first possibility is to delete old versions of the files and deprive them of their ability to be restored. Then, the status of the fileset will change to COMMITTED. The second option will restore the old version of the fileset, which means rolling back the patches. The safe way should always be used when we are not sure about the correct functioning of the components being updated or installed. In order to use the safe way of updating, we should select the following options in the SMIT menu:
  - *"COMMIT software updates?" = "No"* - Patches will not be approved and we can roll it back later.
  - *"SAVE replaced files?" = "Yes"* - Files replaced during patching will be preserved for potential rollback in the future.
- **Traditional way** - Old filesets are replaced with new ones. The installed or updated filesets have the status *COMMITED*. This prevents a simple rollback to the previous filesets. This installation method should only be used if you are sure that the installed or updated components will work correctly.

The choice of method is important when updating. The new files we added can be removed, although restoring them to an earlier version can cause more problems and force us to restore the system from the backup.

As with almost every operation on the AIX operating system, this can be done both with the SMIT tool and from the command line. The following section shows how to do this using the SMIT tool. However, it is important to know what commands to use when using only the command line:

- *installp* - Installs filesets and bundles.
- *instfix* - Installs and checks for installed fixes.
- *emgr* - Installs and checks interim fixes.

- *lslpp* - Displays information about the installed components.
- *lppchk* - Verifies the installation of the fileset.

Examples of using the lslpp command:

Checking the fileset installed on your system:

```
# lslpp -l   # „-l” displays the installed files on the screen.
  Fileset                      Level  State      Description
  ----------------------------------------------------------------------------
Path: /usr/lib/objrepos
  ICU4C.rte                  7.2.0.0  COMMITTED  International Components for
                                                 Unicode
  Java7_64.jre             7.0.0.320  COMMITTED  Java SDK 64-bit Java Runtime
                                                 Environment
  Java7_64.sdk             7.0.0.320  COMMITTED  Java SDK 64-bit Development
                                                 Kit
# All unnecessary information has been omitted.
```

Checking the contents of a given fileset:

```
# lslpp -f bos.rte.cron
  Fileset               File
-------------------------------------------------------------------------------
Path: /usr/lib/objrepos
  bos.rte.cron 7.2.0.0  /usr/bin/at
                        /usr/bin/atq
                        /usr/bin/atrm
                        /usr/bin/batch
                        /usr/bin/cronadm
                        /usr/bin/crontab
                        /usr/sbin/cron
                        /usr/lib/cron -> /var/adm/cron

Path: /etc/objrepos
  bos.rte.cron 7.2.0.0  /etc/cronlog.conf
                        /var/adm/cron
                        /var/adm/cron/at.deny
                        /var/adm/cron/cron.deny
                        /var/adm/cron/log
                        /var/adm/cron/queuedefs
                        /var/spool/cron
                        /var/spool/cron/atjobs
                        /var/spool/cron/crontabs
                        /var/spool/cron/crontabs/adm
                        /var/spool/cron/crontabs/root
                        /var/spool/cron/crontabs/sys
```

Patches

There are errors in the AIX operating system, as there are in every piece of software. They may cause system malfunction in certain situations. They may also violate system security. Such errors occur in any developing software; therefore, one of the most important tasks of the system administrator is to ensure that errors are eliminated as soon as possible.

The system can be patched by installing a patch group (TL, SP) that is periodically provided by IBM. It can also be patched by loading single patches (as a new version of fileset) not yet included in the TL or SP. It may be the case that the problem detected is so serious that you need to fix it immediately. If this happens, you can use the iFix patch. This patch fixes the problem temporarily, and it is replaced at a later point by a Service Pack or by a specific patch on the fileset that caused the problem. It is important to keep in mind that iFix patches do not go through full regression tests. Before using iFix patches, it is worth checking whether there is a workaround for the problem. If such a workaround exists, then we can eliminate the problem by changing the system parameters so that we can wait for a patch that has gone through full regression tests.

It is best practice to ensure that your system has the latest but proven patches. It is imperative to have a test environment that will allow you to verify the behavior of the patches in your application environment. The more similar the test environment is to the production environment, the more reliable the test will be.

IBM maintains several TLs and releases patches for them all. The current information can always be verified on the "Fix Central" pages: *https://www.ibm.com/support/fixcentral/*. At the time of writing this book, only the base version of AIX 7.2 is available, while in AIX 7.1, the patches are issued in parallel for TL3 and TL4, as shown in Figure 5-1.

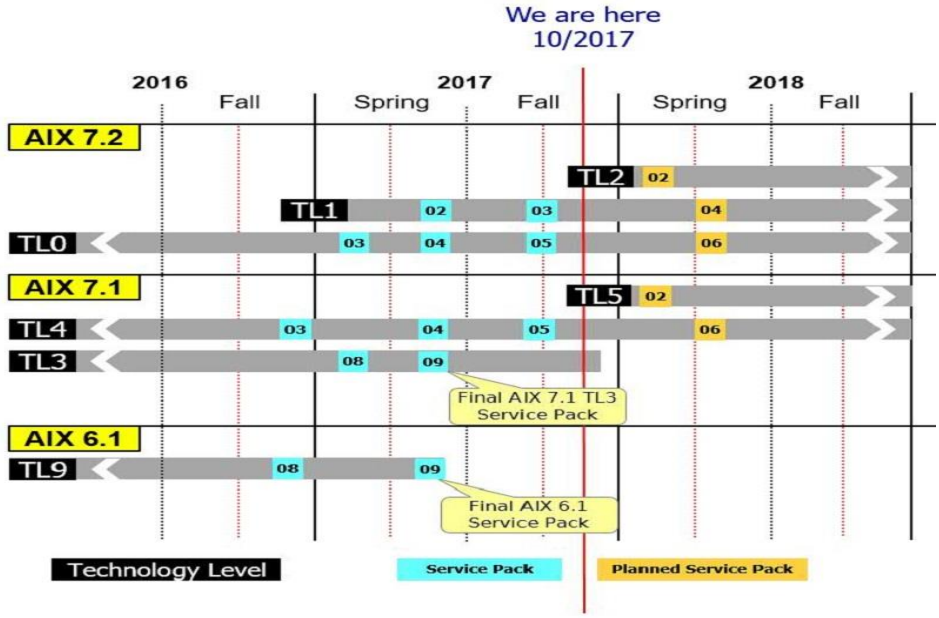

Figure 5-1: IBM Fix Central - TL and SP.

The Technology Levels bring new functionality. If we do not need to use the new functionality, it is a good idea to remain on the previous TL, at least until the first Service Pack is released. The AIX Technology Level is released approximately once a year, while the Service Packs are released for every TL approximately twice a year. You can obtain information about the installed TLs and SPs by using the *instfix* command:

```
# instfix -i | grep ML   # Displays TLs, formerly called Maintenance Levels (ML).
    All filesets for 7.2.0.0_AIX_ML were found.
    All filesets for 7200-00_AIX_ML were found.

# instfix -i | grep SP   # Displays the installed Service Packs.
    All filesets for 72-00-011543_SP were found.
    All filesets for 72-00-021614_SP were found.
```

The above example for AIX 7.2 shows that two Service Packs are correctly installed. There is no Technology Level installed, so the system indicates that the current TL is 0 (base level). To confirm the above information in a less detailed format, you can use the *oslevel* command:

```
# oslevel -s
7200-00-02-1614
```

You should keep track of all the patches that appear. This is important for the security and stability of the system. One way to keep up to date with the patches is to subscribe to this information in the *My Notifications* feature in the IBM Support Portal: *https://support.ibm.com/*.

SUMA (Service Update Management Assistant)

The SUMA tool was designed to release administrators from the need to actively check and download patches from IBM servers. It allows you to automate the download:

- Technology Levels.
- Service Packs.
- Single Fixes.
- All the latest fixes.

This tool has the ability to automatically perform certain tasks at a given time. This way, you can schedule patch downloads for hours when the server and the network are subject to the least load. Communication with the patch server takes place through HTTPS, and the files can be downloaded using HTTP or HTTPS.

If you want to use this tool but you do not have direct access to the internet, you can use a proxy server. Proxy access configurations can be made using the SMIT menu (*smitty srv_conn*).

The patches that the SUMA downloads are stored in the */usr/sys/inst.images* directory by default.

Below, I will describe the capabilities of the SUMA tool. The description will be based on the capabilities of the SMIT menu, but it is important to keep in mind that the same steps can be performed from the command line.

The main SMIT menu for the SUMA (*smit suma*) allows you to:

- *Download Updates Now (Easy)*;
- *Custom/Automated Downloads (Advanced)* and
- *Configure SUMA*.

Configure the SUMA

In most cases, you can use the SUMA tool without any additional configuration. The tool performs tasks based on its predefined configuration. Sometimes, however, the predefined settings are not appropriate for your system; therefore, you should be aware of the configuration options.

You can configure the tool using two menus: *Base Configuration* and *Task Defaults*. You can perform a global configuration using the *smit suma_config_base* or the *sum -c* command. The global parameters that you can configure are given in Figure 5-2 (the following figure shows their default values).

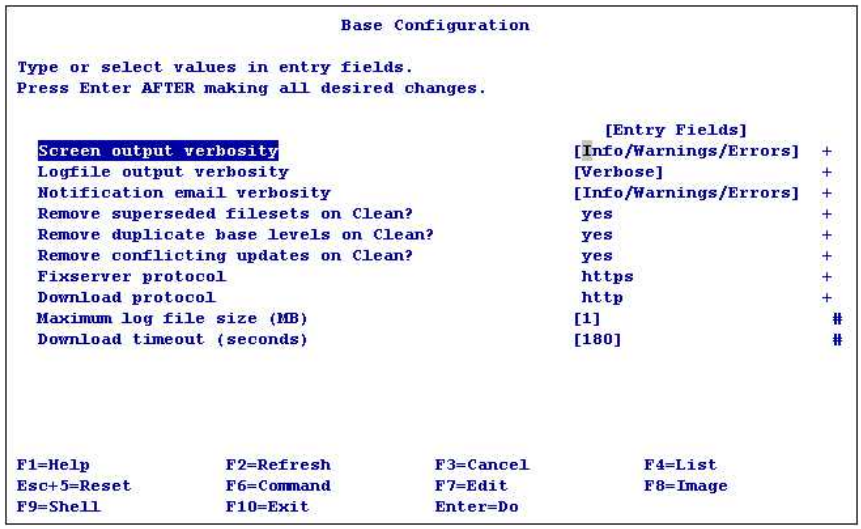

Figure 5-2: smit suma_config_base - The basic parameters of the SUMA tool.

Meaning of the parameters:

Screen output verbosity, Logfile output verbosity, and Notification email verbosity - Specify the detail of the task log. They specify the detail of the information displayed on the screen and recorded in the log file (*/var/adm/ras/suma.log*) or sent by e-mail after the scheduled task. The values that these options may have and the information that is associated with a particular value are specified below:

- *Silent* - No information at all.
- *Errors*.
- *Warnings and Errors*.
- *Info/Warnings/Errors*.

- *Verbose* - Errors, warnings, and other information at a higher level of detail than *Info/Warnings/Errors*.
- *Debug* - A full set of information. This value is not used for normal operation, but it is helpful in solving problems related to downloading patches.

Remove superseded filesets on Clean?, Remove duplicate base levels on Clean?, and Remove conflicting updates on Clean? - Concerns tasks that have a defined *Download and Clean* action. These settings instruct the SUMA to:

- Remove older patches from the download directory, if newer versions are downloaded;
- Remove duplicates; and
- Remove conflicting updates.

Fixserver protocol - The protocol that is used to communicate with the patch server. The only option available is HTTPS. Historically, it was also possible to use HTTP.

Download protocol - The protocol that is used to retrieve patches from the patch server. You can choose between HTTP and HTTPS.

Maximum log file size (MB) - The maximum size that SUMA log files can reach (*/var/adm/ras/suma.log* and *sum_log.log*). When a log file reaches this size, the *.bak* extension is added to its name, and a new file is created in its place. If there is already a file with the same name, it is overwritten.

Download timeout (seconds) - Specifies the idle time of the connection, after which it is interrupted.

To configure tasks in the SUMA tool, you can use the *View/Change SUMA Task Defaults* menu. You can call the menu by running the *smit suma_task_defaults* command. You can also use the *suma -D* command to configure the task without the SMIT tool. You can set the default parameters that will be used to create new tasks. This means that if you do not specify otherwise during creation, the created jobs will use the parameters set here. The configuration options are shown in Figure 5-3 (the default values are displayed).

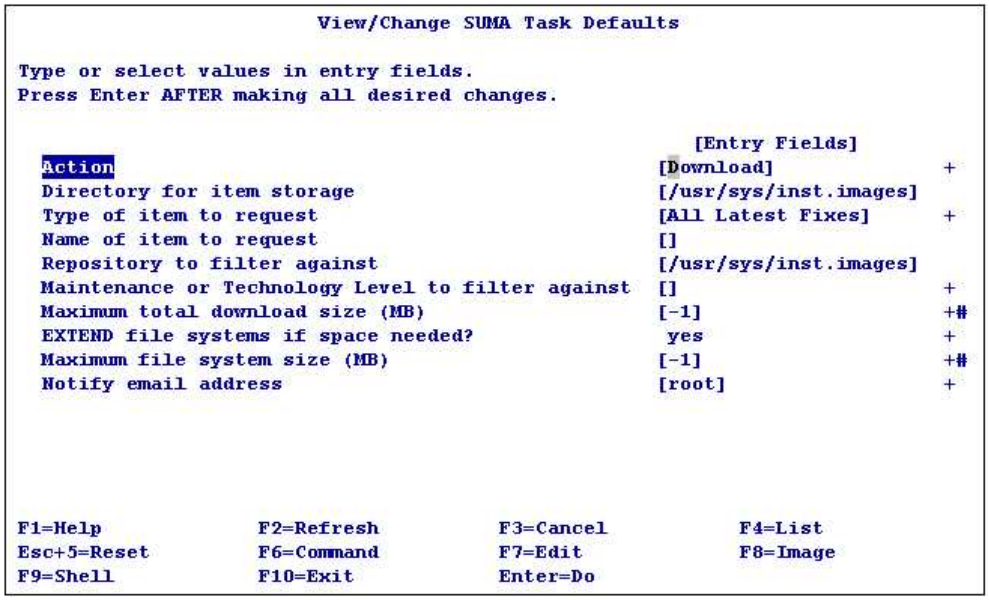

Figure 5-3: smit suma_task_defaults - Parameters of defined tasks.

The most important parameters:

- ***Action*** - The action to be performed by the task. You have the following options.
  - ***Preview*** - Performs a simulation of the patches download task. As a result, you know what will happen during the download, which files will be downloaded, and which will not.
  - ***Download*** - Retrieves the patches and writes it to the location specified in the *Directory for item storage* parameter.
  - ***Download and Clean*** - Retrieves the patches, saves them to the location specified in the *Directory for item storage* parameter, and removes any unnecessary files. The global parameters determine whether to perform cleaning and what files to delete:
    - *Remove superseded filesets on Clean?*
    - *Remove duplicate base levels on Clean?*
    - *Remove conflicting updates on Clean?*
- ***Directory for item storage*** - Indicates the directory where the downloaded files are located.
- ***Type of item to request*** - The type of file to download. You can choose between the following types:
  - ***PTF*** - Patches created between Service Pack releases.
  - ***Maintenance Level*** - The group of patches. The name *Maintenance Level* has been replaced by the name *Technology Level*.
  - ***Technology Level***.
  - ***Service Pack***.
- ***Name of item to request*** - The name of the files to download. In this field, you can specify different values depending on the type you chose earlier:
  - ***PTF*** - PTF number, for example, "U254389."
  - ***Maintenance Level/Technology Level*** - The group of patches, for example, "7100-04."
  - ***Service Pack*** - Service Pack number, for example, "7100-04-03."

- ***Repository to filter against*** - Indicates the directory on which the tool filters the downloaded files. Files that are already in the specified directory are not downloaded.
- ***Maintenance level to filter against*** - Indicates the level of the system, based on which SUMA filters the downloaded files. This means that when the first level of AIX 7.1 (7100-03) is specified, the files in the group of patches from that level are skipped during the download.
- ***Maximum total download size (MB)*** - The maximum size of all the files downloaded by the task. If the downloaded files exceed this value, instead of downloading them, the task will perform a *Preview* action. Further, *-1* means no limit.
- ***EXTEND file systems if space needed?*** - Specifies whether the operating system is to expand the file system if there is no space available when downloading files.
- ***Maximum file system size (MB)*** - Specifies the maximum size to which the file system can be automatically extended (see the above parameter).
- ***Notify email address*** - An email address or a number of addresses separated by semicolons where the SUMA sends notifications.

Download updates

The *Download Updates Now (Easy)* menu (*smit suma_easy*), as shown in Figure 5-4, allows you to easily retrieve specific patches. The number of options to choose from is minimized. The system uses the default parameters defined in the above *Task Defaults* menu. The options allow you to:

- ***Download Maintenance Level or Technology Level*** - Download a whole group of patches;
- ***Download Service Pack***; and
- ***Download All Latest Fixes*** - The SUMA configuration (*Task Defaults*) determines which patches to download.

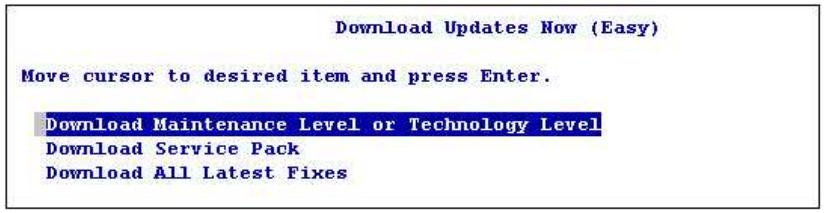

Figure 5-4: smit suma_easy - Download updates.

You can use the *Custom/Automated Downloads (Advanced)* menu (*smit suma_task*) to support less standard tasks. This option allows you to create, delete, view, and change such tasks. The created task can be run immediately, once at a given time, or cyclically. The task creation format is shown in Figure 5-5.

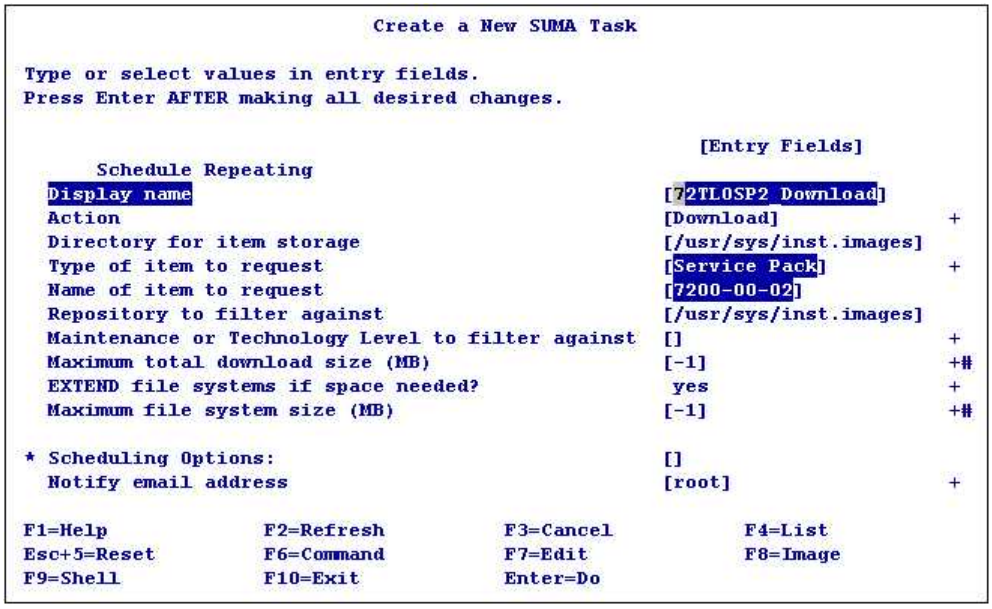

Figure 5-5: smit suma_task_new - Create a new SUMA Task.

The parameters listed here are the same as in the *View/Change SUMA Task Defaults* menu. In the previous SMIT menu, you defined the default values, but in this one, you can change them for a single task. The meanings of these parameters were provided earlier in this chapter.

When defining a task to be run at a particular time or cyclically, you need to configure the *Scheduling Options*. In this parameter, you should specify the execution date in the same format as that used in the Cron tool (i.e., "minute hour day_of_the_month month day_of_the_week"). Sample parameter values:

- 45 2 * * 0 - The task starts every Sunday at 2:45. 0 means Sunday, since the days of the week are defined as numbers from 0 to 6 (Sunday to Saturday).
- 45 2 1 12 * - The task is launched on December 1st at 2:45.

Compare report

You can use the IBM Fix Central pages to search for and download patches. There is also an option to download information about the components (filesets) of a given package (TL, SP). This option is especially useful if a few filesets are lacking from the operating system. You can download the filesets' information about the patch package by clicking the *Compare report* link shown in Figure 5-6.

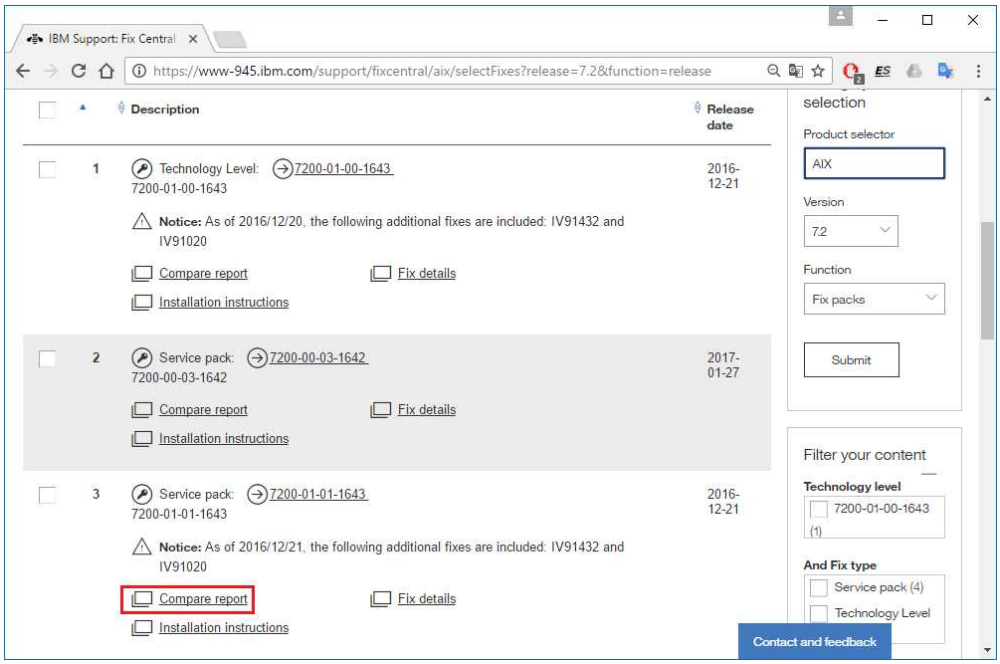

Figure 5-6: Fix Central - Compare report.

The first step toward getting the patches this way is to download the file that contains the list of patches that are in the specific package. You should copy the downloaded file to the AIX operating system, which will then be updated, for example, to the */tmp* directory.

Then, based on both this file and the information that is in the system, you should generate a report listing the missing filesets needed to update the system. Use the *compare_report* command to generate the report. Usage example:

```
# compare_report -s -r /tmp/7200-01-01-1643.compare -l
#(lowerthanlatest1.rpt)
#Installed Software that is at a LOWER level
#PTF:Fileset_Name:Installed_Level:Available_Level
U874558:ICU4C.rte:7.2.0.0:7.2.1.0
U871559:Java7_64.jre:7.0.0.320:7.0.0.370
U871560:Java7_64.sdk:7.0.0.320:7.0.0.370
U872502:X11.adt.lib:7.2.0.1:7.2.1.0
U863743:X11.apps.rte:7.2.0.1:7.2.1.0
#…unnecessary information has been cut.
#(lowerthanmaint.rpt)
#Installed Software that is at a LOWER level than
#   the latest maintenance level
#PTF:Fileset_Name:Installed_Level:Available_Level
U874558:ICU4C.rte:7.2.0.0:7.2.1.0
U871559:Java7_64.jre:7.0.0.320:7.0.0.370
U871560:Java7_64.sdk:7.0.0.320:7.0.0.370
U872502:X11.adt.lib:7.2.0.1:7.2.1.0
# All unnecessary information has been omitted.
```

Parameters used:

- *-s* - Instructs the command to retrieve the current list of filesets installed in order to compare it

to the list downloaded from the IBM Fix Central.

- ***-r /tmp/LatestFixData53*** - Indicates a file downloaded from the IBM Fix Central.
- ***-l*** - Instructs the command to generate the following reports:
  - ***/tmp/lowerthanlatest1.rpt*** - A list of filesets that are at a lower level than the latest available filesets.
  - ***/tmp/lowerthanmaint.rpt*** - A list of filesets that are at a lower level than the filesets in the latest maintenance and technology levels.

The *compare_report* command has far more capabilities than just the one illustrated above. It allows you to compare the components of the systems installed on different servers, or to compare the filesets installed on the operating system with the filesets in the repository. In this case, the repository is the directory where the filesets are located.

Thanks to the generated reports, you can download the necessary patch without having to download the entire package. This way, you can upgrade the system to the desired level (*lowerthanmaint.rpt* report). It should be noted here that, based on best practice for patch installations in the AIX operating system, you should use full service packs rather than individual patches.

Patch installation

You can install patches directly from the command line or from the SMIT menu. There is a link to the installation instructions when downloading the patch group from IBM Fix Central. Therefore, there is no point in describing this process in detail here.

However, you should know that after you copy the downloaded files to the operating system (or make them available on a network resource such as NFS), you should run the *inutoc directory_name* command. *Directory_name* is the name of the directory where the downloaded files are located. This command creates a *.toc* file in that directory. This file contains information about the files found here, and it is used by the installation command.

You can skip the creation of a *.toc* file, but you must then make sure that there is no such file in the directory that contains **.bff* files (there may be a previously created file containing incorrect information). If there is no *.toc* file, the installer will automatically create it.

Another useful thing to do is to name the patches you have downloaded. After being downloaded, they have meaningless names, as in the following example:

```
# ls | head -5   # List the first five fileset in the directory.
U856876.bff
U856877.bff
U856878.bff
U856879.bff
U856880.bff
```

After executing the *bffcreate* command in the directory where they are located, the patches receive names that match their functionality:

```
# bffcreate -c -d .   # Naming of filesets.
# ls | head -5
```

```
bos.adt.syscalls.7.2.1.0.U
bos.diag.util.7.2.1.0.U
bos.rte.shell.7.2.1.0.U
devices.artic960.rte.7.2.1.0.U
security.acf.7.2.1.0.U
```

At this point, you can start the installation of the patches. However, if you want to use the preview option, which tells you about the expected installation status, you should first individually install the installation fileset. Without this step, the preview option will show you that the installation fileset will be installed correctly, and that the remainder of the fileset will be in the *Deferred* status. For example:

```
FILESET STATISTICS
------------------
  286  Selected to be installed, of which:
         1  Passed pre-installation verification
       285  Deferred (see *NOTE below)
*NOTE  The deferred filesets mentioned above will be processed after the
       installp update and its requisites are successfully installed.
```

If you use the *preview* option after installing the installation fileset, you will obtain full information about the expected installation process.

```
# installp -acgXd /tmp/aix72_01_01 bos.rte.install
```

You can install patches from the SMIT menu. Indeed, this is the easiest way. You can enter the appropriate menu by using the *smitty update_all* command, and then select the patches' source. After doing so, you will see the menu shown in Figure 5-7.

```
                Update Installed Software to Latest Level (Update All)

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

                                                        [Entry Fields]
* INPUT device / directory for software                 .
* SOFTWARE to update                                    _update_all
  PREVIEW only? (update operation will NOT occur)       no                      +
  COMMIT software updates?                              yes                     +
  SAVE replaced files?                                  no                      +
  AUTOMATICALLY install requisite software?             yes                     +
  EXTEND file systems if space needed?                  yes                     +
  VERIFY install and check file sizes?                  no                      +
  DETAILED output?                                      no                      +
  Process multiple volumes?                             yes                     +
  ACCEPT new license agreements?                        yes                     +
  Preview new LICENSE agreements?                       no                      +

  WPAR Management
      Perform Operation in Global Environment           yes                     +
      Perform Operation on Detached WPARs               no                      +
          Detached WPAR Names                           [_all_wpars]            +
      Remount Installation Device in WPARs              yes                     +
      Alternate WPAR Installation Device                []


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

Figure 5-7: smit update_all - Patch installation.

Most of the options available here will be explained in the following sections. Now, let me just say that when you install a new "Technology Level," you need to change the "*ACCEPT new license agreements option?*" to "*yes*." This option automatically accepts the license terms of the updated fileset. Otherwise, the installation may fail.

You can install a patch group with the ability to return to the old version (set the "*COMMIT software updates?*" to "*no*" and the "*SAVE replaced files*" to "*yes*"). An update made in this way has the advantages that have already been discussed. However, it requires almost twice as much disk space and lasts almost twice as long. The reason for the long duration of this operation is the need to copy the replaced files in order to preserve the ability to restore them.

Please note that during the installation, AIX will update only the older versions of its filesets. If any of the files selected for installation are older than the current ones, they will be omitted. After installing the patch group, it is usually necessary to restart the system.

You can install individual filesets in a similar way to the installation of a patch group, although you should use the *smitty install_latest* menu.

Updating your system usually means installing newer versions of its components. Therefore, the installation of additional software is similar to the patching of the operating system. You will learn how it works in the subsection concerning the installation of additional system components.

Issues with patches

When you install the patch group, you should verify that the update process has been completed successfully. Hence, you need to check the system level. You can use the *instfix* command with *-i* to display all the installed patches:

```
#  instfix -i | grep ML   # Check Technology Level (Maintenance level - ML).
    All filesets for 7.2.0.0_AIX_ML were found.
    All filesets for 7200-00_AIX_ML were found.
    Not all filesets for 7200-01_AIX_ML were found.   # At least one fileset from the Technology
Level 1 is missing.

#  instfix -i | grep SP   # All files from Service Packs are installed.
    All filesets for 72-00-011543_SP were found.
    All filesets for 72-00-021614_SP were found.
    All filesets for 72-00-031642_SP were found.
    All filesets for 72-01-011642_SP were found.
```

Alternatively, you can use the *oslevel* command to determine the highest service pack reached for the current technology level on the system:

```
# oslevel -s
7200-00-03-1642   # TL0 SP3.
```

You can obtain information on what level of fixes the operating system has reached by filtering the output of the *instfix -i* command by "*ML*." This way, you can also check whether all the required filesets on a given level are in the system in the respective versions.

The last line of the above output shows that the AIX 7.2 operating system we tested is missing some TL1 filesets. This can happen in two situations:

- After installing the patch group - This means that the system update was not completely successful. In this case, analyze the cause and install the missing files.
- After installing an additional system component - This means that the system component that you have installed is older than it should be for your system level. This can occur when you install a new system component from the installation DVDs after patching the system to a higher level. The installed component will be found in the pre-patched version of the system. In this case, you can download the missing fileset from IBM Fix Central or re-install the patch group, which will install the missing fileset.

The lack of a current fileset on a given level is usually not dangerous for the operating system. However, you should always assume that the updates for individual filesets were not created for simply no reason. To ensure that your system is functioning properly, you should maintain order and always have all the file groups updated to the appropriate level.

The solution for individual missing filesets is to download their respective versions and install them. You can check which filesets should be downloaded using the *instfix* command:

```
# instfix -icqk 7200-01_AIX_ML |grep ":-:"
7200-01_AIX_ML:Java7_64.jre:7.0.0.370:7.0.0.320:-:AIX 7200-01 Update
```

The above command looks for all the missing AIX 7.2 TL1 files. The 7200 string represents the system version (AIX 7.2) in which we perform this operation.

The missing filesets can be downloaded from the previously mentioned IBM pages by selecting *Fix search* and then searching by name.

Installation of additional system components

The most convenient way to install additional system components is to install them using SMIT. This way, we can perform several types of installation, as shown in Figure 5-8.

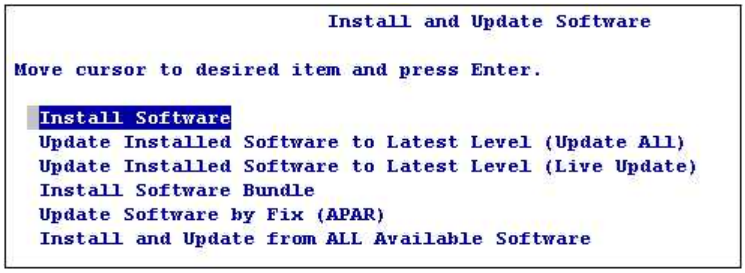

Figure 5-8: SMIT - Installation types.

Install Software - Allows you to install or reinstall any fileset. This option is mainly used for installing single system components.

Update Installed Software to Latest Level (Update All) - An option to update your operating

system by installing a patch group that increases its level.

Update Installed Software to the Latest Level (Live Update) - An option used to call the so-called Live Update. This is a way to install patches without triggering operating system inaccessibility. This option is available from AIX 7.2, and its use requires prior preparation.

Install Software Bundle - Allows you to install multiple filesets assembled into one bundle based on their role. For example, you can choose to install a bundle defined in the system such as:

- PerfTools - Includes all the filesets needed to manage system performance.
- App-Dev - Includes the filesets needed by developers working on AIX.

There are many more bundles of this type, and you can also define your own bundle.

Update Software by Fix (APAR) - Allows you to install filesets identified by the APAR number. Numbers of this type identify the fileset versions that solve a given problem. The solution to the problem is usually to install one or more filesets.

Install and Update from All Available Software - A rarely used option. With this menu, you can re-install the fileset to any version you have, even an older one.

Install software

Selecting any of the options listed above provides you with access to a similar menu. Below, we will focus on the most commonly used option, namely *Install Software*. If you select this option and then specify the path to the filesets you are installing, you will see the menu shown in Figure 5-9.

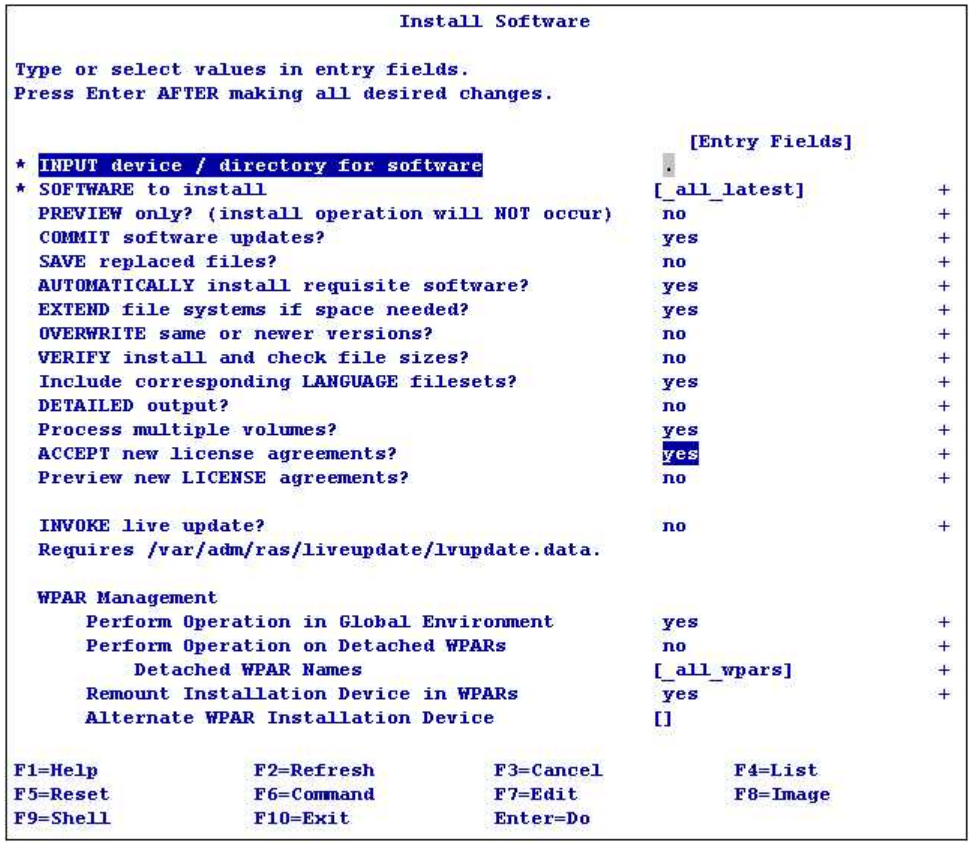

Figure 5-9: smit install_latest.

Key options:

SOFTWARE to install - Allows you to select the filesets for installation. It refers to the filesets that are listed in the directory shown above.

PREVIEW only? (install operation will NOT occur) - If you set this option to "*yes,*" then a simulation will be performed instead of an installation. As a result, no filesets will be installed. The operating system will check the amount of free space in the file systems as well as the dependency of the additional filesets that should be installed in order to install the product. As a result of this simulation, the system tells you what filesets are installed and what filesets are required for the product but are currently missing from the system.

COMMIT software updates? – Setting this option to "*yes*" will commit the changes you made without saving the old filesets. You can use this setting if you are sure that the components you are installing will work correctly. If you set this option to "*no*" and set the "*SAVE replaced files?*" to "*yes,*" you will have the option to roll back the installed components at any time and restore the old system-preserved files located in the */usr/lpp* directory.

AUTOMATICALLY install requisite software? - If you set this option to "*yes*," the system will automatically install the filesets required for the correct installation of the components. Of course, this can only occur if it has access to them, for example, if they are in the same directory as the filesets you

are installing. If you set the option to "*no*" and the operating system has none of the components required for the correct operation of the fileset you are installing, the fileset will not be installed. It will only display a report listing the missing filesets.

EXTEND file systems if space needed? - Setting this option to "*yes*" will automatically expand the file systems if necessary.

OVERWRITE same or newer versions? - You should set this option if you want to reinstall the given fileset or to install an older version. If you set this option to "*yes*," you need to set the "*AUTOMATICALLY install requisite software?*" option to "*no*."

VERIFY install and check file sizes? - This checks the checksum of the files. This way, you can ensure that the installation is running correctly. Setting this option slightly increases the duration of the installation.

Include corresponding LANGUAGE filesets? - This installs message files in the preset language of the operating system. The messages are related to the installed fileset. You should set this option to "*yes*" to ensure the display of messages.

Process multiple volumes? - If the installed files are on numerous media (for example, DVDs), it is recommended to leave the default setting as "*yes*."

ACCEPT new license agreements? - If you do not accept the license associated with the product (filesets), you will not be able to install it. So, set this option to "*yes*" before installing new filesets.

WPAR Management - This section contains a number of OS-level virtualization options. WPARs are rarely used, as opposed to LPARs.

Remove installed software

You can remove any software that is in the *COMMITTED* or *BROKEN* state, which indicates its corruption. Filesets that are in the *APPLIED* state can only be committed (accepted) or rejected. Rejection deletes the specific fileset and restores previous versions.

If you delete a product, all its updates and configuration information will be removed. You can restore a deleted component by reinstalling it. A smitty remove example is given in Figure 5-10.

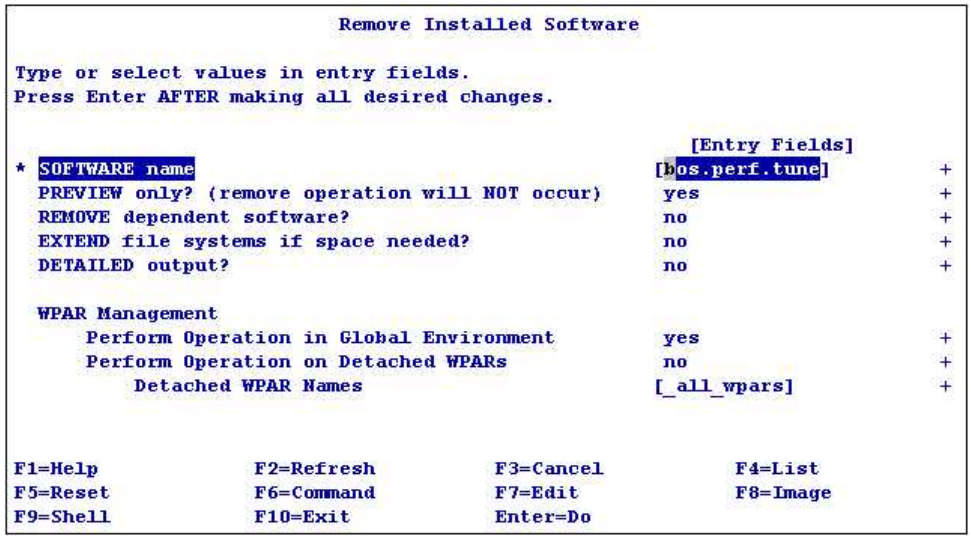

Figure 5-10: smit remove - Removing installed system components.

The only unclear option in this SMIT menu is "*REMOVE dependent software?*." A system component cannot be removed if other components require it. Such an attempt will fail and cause a display of the components that need to be removed beforehand. In that case, you can set this option to "*yes*," and it will remove all the components that depend on what you want to remove.

You should be very careful when using this option due to the possibility of deleting more filesets than you intended. Further, if it does prove necessary to use it, you should first check its effect by running the task in *PREVIEW only* mode.

If you want to remove a product that is in the *APPLIED* state, simply reject it, which will remove and undo the changes made during the installation. Use the *smit reject* menu to reject the changes.

To approve a component that is in the APPLIED state, you can use a similar *smitty commit* menu. It is important to note that the approval of installed filesets will remove the information needed to reject them from the */usr/lpp* directory. The positive aspect of this is the increase in free space in the */usr* file system.

Cloning the operating system (alt_disk)

The AIX operating system is dynamically developed. Several times a year, a new Technology Level and Service Pack are created. The administrator's job is to install them in a safe and user-friendly way, with minimal downtime. A very good mechanism for this task is the cloning of the system.

A clone is a copy of a system that is created on a separate disk. You can boot the system from a clone. Changes made to the original system have no effect on the clone, and vice versa. The clone can be treated as a second operating system installation on the same server from which it can be started.

Cloning is performed on a running system. It is based on three commands:

- ***alt_disk_copy*** - Allows you to create a system clone on an empty disk.
- ***alt_disk_mksysb*** - Allows you to install the system on a separate disk from the backup previously prepared using the *mksysb* tool.
- ***alt_rootvg_op*** - Due to the previous two commands, a new volume group named *altinst_rootvg* is created. With this command, you can perform various operations.

alt_disk_copy command

The *alt_disk_copy* command allows you to create a *rootvg* volume group clone on a separate disk. The cloned system can be upgraded and patched. You can install new filesets and modify your system in order to meet new needs. The clone can also exist in the system as a backup of the rootvg group. The big advantage of cloning is that it can be performed during normal system operation.

Cloning makes it safer to update your system, since the update can take place on a copy of the system. If the system will not run properly, you can just restart and boot from the disk containing the original version (which was cloned before the update).

To create a clone, the server should have a disk that is not assigned to any volume group. In addition, the disk must be of a sufficient size to accommodate the contents of the cloned *rootvg* volume group.

In the following example, the free disk is *hdisk1*, while the currently running system is installed on *hdisk0*. Cloning involves many activities:

```
# alt_disk_copy -d hdisk1   # Cloning the operating system (rootvg group) to 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_hd1
Creating logical volume alt_hd10opt
Creating logical volume alt_hd11admin
Creating logical volume alt_lg_dumplv
Creating logical volume alt_livedump
Creating /alt_inst/ file system.
Creating /alt_inst/admin 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.
Creating /alt_inst/var/adm/ras/livedump 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.
forced unmount of /alt_inst/var/adm/ras/livedump
```

```
forced unmount of /alt_inst/var/adm/ras/livedump
forced unmount of /alt_inst/var
forced unmount of /alt_inst/var
forced unmount of /alt_inst/usr
forced unmount of /alt_inst/usr
forced unmount of /alt_inst/tmp
forced unmount of /alt_inst/tmp
forced unmount of /alt_inst/opt
forced unmount of /alt_inst/opt
forced unmount of /alt_inst/home
forced unmount of /alt_inst/home
forced unmount of /alt_inst/admin
forced unmount of /alt_inst/admin
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 blv=hd5
```

The first step is to create a */image.data* file that describes the structure of the *rootvg* volume group. It is a source of information when creating structures on a cloned disk. Then, the command checks the size of the disk on which the clone is to be created in order to determine if it is sufficient.

Once this is done, all the necessary structures are created on the disk:

- A volume group named *altinst_rootvg*.
- Logical volumes with names the same as the originals but preceded by the prefix *alt_*.
- File systems on logical volumes.

From the source file systems, the files are copied to the structures created on the clone. As you might expect, this is the longest phase of the cloning. Then, the cloned system disk is prepared in such a way that it can serve as a boot disk:

- The ODM database is updated.
- The BLV (boot logical volume) is created, which is a logical volume containing the kernel version for the boot process.
- All the file systems mounted on the cloned disk are unmounted.
- The VGDA is modified - This is the initial disk space that holds metadata about its contents.
- The logical volume control block and file system control blocks are modified.
- The logical volume and file system control blocks are modified.
- The bootlist is modified so that the operating system is booted from the disk clone.

After these operations, the disk clone represents a new volume group known as the *altinst_rootvg*. It is inactive. The disk clone is the first one on the bootlist, which means that the operating system will boot from it after the reboot. After the reboot, the *altinst_rootvg* group will be renamed as the *rootvg*, while the current *rootvg* group will be renamed as the *old_rootvg*:

```
# lsvg   # Immediately after the clone is created.
rootvg
altinst_rootvg

# lspv   # System booted from hdisk0.
hdisk0          00f62177fc932e26                    rootvg          active
```

```
hdisk1          00f6217735697d2a                    altinst_rootvg

# lsvg   # After system reboot.
old_rootvg
rootvg

# lspv   # System started from a clone (hdisk1).
hdisk0          00f62177fc932e26                    old_rootvg
hdisk1          00f6217735697d2a                    rootvg          active
```

To switch between the first and second instances of the operating system, simply change the boot order and reboot the system. The system will boot from this disk, which is first on the bootlist.

When you boot the operating system from the disk that contains its original version, the volume group names will change back to *rootvg* and *altinst_rootvg*:

```
# lsvg
rootvg
altinst_rootvg
```

Update the clone

The *alt_disk_copy* command is executed in three phases. By default, all the phases are performed, although you can run them individually by using the *-P* parameter and specifying which phases you want to run:

- First phase - The structures are created, and data are copied to them. When this phase is over, we gain access to the cloned system. Its file systems are mounted in the */alt_inst* directory, so you can make any modifications.
- Second phase - This validates the structures created during the first phase.
- Third phase - The disk clone is prepared to boot the operating system. After this phase, you can boot the operating system from the clone, but the volume group created on it is unavailable.

The main purpose of a system clone is to make changes to it rather than to the original system version. After making changes, the system can be started from the clone. If our changes will have a negative effect on system stability, we can quickly return to the unchanged version.

Migrating the system to a newer version as well as the upgrading or patching of files is done after the first phase of the command. The above steps will only be performed if you intend to do so and you have entered the appropriate parameters for the command.

You can clone the system and update the clone using one command *alt_disk_copy*, which starts all three phases of the command (this is the default action). You can perform the three phases in turn as well, starting each phase with the *alt_disk_copy* command with other parameters. When you start the second or third phase, you can add parameters to show what you want to install and where you want to do so. At the end of the first or second phase, you can make any necessary modifications to the cloned files that are mounted in the */alt_inst* directory:

```
# alt_disk_copy -d hdisk1 -P 1   # First phase – cloning.
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_hd11admin
Creating logical volume alt_lg_dumplv
Creating logical volume alt_livedump
Creating /alt_inst/ file system.
Creating /alt_inst/admin 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.
Creating /alt_inst/var/adm/ras/livedump 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.

# df -k   # After the first phase you can operate on cloned file systems ("/alt_ *").
Filesystem       1024-blocks      Free     %Used    Iused   %Iused Mounted on
/dev/hd4            344064       167452     52%     11436     23% /
/dev/hd2            2080768      208264     90%     36711     43% /usr
/dev/hd9var         180224       143268     21%     608        2%  /var
/dev/hd3            98304        97512      1%      37         1%  /tmp
/dev/hd1            16384        16012      3%      7          1%  /home
/dev/hd11admin      131072       130708     1%      5          1%  /admin
/proc                 -            -        -       -          -   /proc
/dev/hd10opt        32768        15964      52%     225        6%  /opt
/dev/livedump       262144       261776     1%      4          1%  /var/adm/ras/livedump
/dev/alt_hd4        344064       168020     52%     11406     23% /alt_inst
/dev/alt_hd11admin  131072       130708     1%      5          1%  /alt_inst/admin
/dev/alt_hd1        16384        16012      3%      7          1%  /alt_inst/home
/dev/alt_hd10opt    32768        15964      52%     225        6%  /alt_inst/opt
/dev/alt_hd3        98304        94708      4%      37         1%  /alt_inst/tmp
/dev/alt_hd2        2080768      208648     90%     36711     43% /alt_inst/usr
/dev/alt_hd9var     180224       142896     21%     601        2%  /alt_inst/var
/dev/alt_livedump   262144       261776     1%      4          1% /alt_inst/var/adm/ras/livedump
```

To finish the cloning process, run the *alt_disk_copy* command indicating phases two and three.

To update the system clone after all three phases, you should wake up the clone using the *alt_rootvg_op* command (this will be discussed later in this chapter). You can also update the system during the cloning process itself. These capabilities are detailed in the *smit alt_clone* menu shown in Figure 5-11.

```
                    Clone the rootvg to an Alternate Disk

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

                                                      [Entry Fields]
* Target Disk(s) to install                          [hdisk1]          +
  Phase to execute                                    all              +
  image.data file                                     []                /
  Exclude list                                        []                /

  Bundle to install                                   []               +
    -OR-
  Fileset(s) to install                               []

  Fix bundle to install                               []
    -OR-
  Fixes to install                                    []

  Directory or Device with images                     []
  (required if filesets, bundles or fixes used)

  installp Flags
  COMMIT software updates?                            yes              +
  SAVE replaced files?                                no               +
  AUTOMATICALLY install requisite software?           yes              +
  EXTEND file systems if space needed?                yes              +
  OVERWRITE same or newer versions?                   no               +
  VERIFY install and check file sizes?                no               +
  ACCEPT new license agreements?                      no               +

  Customization script                                []                /
  Set bootlist to boot from this disk
  on next reboot?                                     yes              +
  Reboot when complete?                               no               +
  Verbose output?                                     no               +
  Debug output?                                       no               +

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

Figure 5-11: smit alt_clone.

The above menu runs the *alt_disk_copy* command with the appropriate parameters. As you can see, you can install filesets, bundles, and individual fixes. The menu gives you options similar to those you have for the regular installation of system components. For example, automatically commit installed components, retention of older versions for restoration, etc.

alt_disk_mksysb command

The *alt_disk_mksysb* command allows you to install an operating system from a backup created with the *mksysb* utility. The installation can be performed from the running operating system to any free disk. The *mksysb* utility allows you to install a system from a backup created on another server (see the backup chapter for more information). The following is an example of a system installation on *hdisk1* from a backup (that is, an operating system, which is the *rootvg* group on *hdisk0*). This example uses a backup file as the source, although you can also successfully use a backup created on a tape or DVD.

```
# alt_disk_install -d /tmp/mksysb_1225.bckp hdisk1
+-------------------------------------------------------------------------------+
ATTENTION: calling new module /usr/sbin/alt_disk_mksysb. Please see the alt_disk_mksysb man page
and documentation for more details.
Executing command: {/usr/sbin/alt_disk_mksysb -m /tmp/mksysb_1225.bckp -d "hdisk1"}
+-------------------------------------------------------------------------------+
Restoring /image.data from mksysb image.
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_hd11admin
Creating logical volume alt_lg_dumplv
Creating logical volume alt_livedump
Creating /alt_inst/ file system.
Creating /alt_inst/admin 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.
Creating /alt_inst/var/adm/ras/livedump file system.
Restoring mksysb image to alternate disk(s).
Linking to 64bit kernel.
Changing logical volume names in volume group descriptor area.
Fixing LV control blocks...
forced unmount of /alt_inst/var/adm/ras/livedump
forced unmount of /alt_inst/var/adm/ras/livedump
forced unmount of /alt_inst/var
forced unmount of /alt_inst/var
forced unmount of /alt_inst/usr
forced unmount of /alt_inst/usr
forced unmount of /alt_inst/tmp
forced unmount of /alt_inst/tmp
forced unmount of /alt_inst/opt
forced unmount of /alt_inst/opt
forced unmount of /alt_inst/home
forced unmount of /alt_inst/home
forced unmount of /alt_inst/admin
forced unmount of /alt_inst/admin
forced unmount of /alt_inst
forced unmount of /alt_inst
Fixing file system superblocks...
Bootlist is set to the boot disk: hdisk1 blv=hd5
```

The whole process is almost identical to the cloning of the system using the *alt_disk_copy* command. There are only minor differences that must exist due to the different data source (a disk for *alt_disk_copy* versus a backup file for *alt_disk_mksysb*).

Once the system is installed on the target disk, an inactive *altinst_rootvg* group containing the installed system is created. The target disk is first on the bootlist. This means that if you reboot the operating system, the system will boot from it. After the system reboot, the current *rootvg* volume group will be renamed as the *old_rootvg*, while *altinst_rootvg* will be renamed as the *rootvg*. To switch between instances of the system, simply change the disk from which the system boots (*bootlist* command).

Update of an alternate operating system

The *alt_disk_mksysb* command, like the *alt_disk_copy* command, is executed in three phases. For both commands, the phases involve the same steps. All the phases are performed by default, although you can perform them in turn by using the *-P* parameter and specifying which phases to perform.

Installing a system from a backup using the *alt_disk_mksysb* command does not provide such extensive update capabilities as the *alt_disk_copy* command. There are no parameters for installing the filesets during the second or third phases of the command execution. These deficiencies can be mitigated using the *alt_rootvg_op* command. An example of such a solution follows:

First phase:
```
# alt_disk_install -d /tmp/mksysb_1225.bckp -P 1 hdisk1
```

Update - install all fileset from / updates:
```
# alt_rootvg_op -d hdisk1 -b update_all -l /updates
```

Second and third phase:
```
# alt_disk_mksysb -d /tmp/mksysb_1225.bckp -P 23 hdisk1
```

It should be noted that after the first and second phases, full access is given to the files of the installed system, since all the newly created file systems are mounted in the */alt_inst* directory.

alt_rootvg_op command

This command allows you to operate on clones (alternate system instances). It allows you to update or install additional filesets, or to make additional configurations.

For updates, the clone must be in the state that it was after the first or second cloning phase. The volume group must be active, and its file systems mounted in the */alt_inst* directory. This process is referred to as waking up the clone *(alt_rootvg_op -W)*:

```
# df -k   # before waking up the clone.
Filesystem    1024-blocks      Free    %Used    Iused   %Iused Mounted on
/dev/hd4           344064     167444    52%     11402    23% /
/dev/hd2           2080768    208244    90%     36713    43% /usr
/dev/hd9var        180224     143108    21%     602      2%  /var
/dev/hd3           2195456    664536    70%     39       1%  /tmp
/dev/hd1           16384      16012     3%      7        1%  /home
/dev/hd11admin     131072     130692    1%      7        1%  /admin
/proc                -          -       -       -        -   /proc
/dev/hd10opt       32768      15964     52%     225      6%  /opt
/dev/livedump      262144     261776    1%      4        1%  /var/adm/ras/livedump

# alt_rootvg_op -Wd hdisk1   # Waking up the clone..
Waking up altinst_rootvg volume group ...

# df -k   # After awaking the clone.
Filesystem     1024-blocks      Free    %Used    Iused   %Iused Mounted on
/dev/hd4           344064     167436    52%     11428    23% /
/dev/hd2           2080768    208244    90%     36713    43% /usr
/dev/hd9var        180224     143100    21%     602      2% /var
/dev/hd3           2195456    664532    70%     40       1% /tmp
```

```
/dev/hd1           16384       16012     3%      7         1% /home
/dev/hd11admin     131072      130692    1%      7         1% /admin
/proc                 -           -         -       -         - /proc
/dev/hd10opt       32768       15964     52%     225       6% /opt
/dev/livedump      262144      261776    1%      4         1% /var/adm/ras/livedump
/dev/alt_hd4       344064      167896    52%     11089     23% /alt_inst
/dev/alt_hd11admin 131072      130708    1%      5         1% /alt_inst/admin
/dev/alt_hd1       16384       16012     3%      7         1% /alt_inst/home
/dev/alt_hd10opt   32768       15964     52%     225       6% /alt_inst/opt
/dev/alt_hd3       2195456     2194168   1%      39        1% /alt_inst/tmp
/dev/alt_hd2       2080768     208252    90%     36711     43% /alt_inst/usr
/dev/alt_hd9var    180224      143052    21%     594       2% /alt_inst/var
/dev/alt_livedump  262144      261776    1%      4         1% /alt_inst/var/adm/ras/livedump
```

You can modify, install, and update the system on the awakened clone. The following example will update all the filesets from the */updates* directory:

```
# alt_rootvg_op -d hdisk1 -b update_all -l /updates
```

The *-I* parameter can prove very useful when updating or installing filesets. It allows you to specify the flags with which the *installp* command will run.

After the update and all the necessary operations on the file systems have been completed, the clone needs to be "put to sleep" (*-S* flag):

```
# alt_rootvg_op -Sd hdisk1
Putting volume group altinst_rootvg to sleep ...
forced unmount of /alt_inst/var/adm/ras/livedump
forced unmount of /alt_inst/var/adm/ras/livedump
forced unmount of /alt_inst/var
forced unmount of /alt_inst/var
forced unmount of /alt_inst/usr
forced unmount of /alt_inst/usr
forced unmount of /alt_inst/tmp
forced unmount of /alt_inst/tmp
forced unmount of /alt_inst/opt
forced unmount of /alt_inst/opt
forced unmount of /alt_inst/home
forced unmount of /alt_inst/home
forced unmount of /alt_inst/admin
forced unmount of /alt_inst/admin
forced unmount of /alt_inst
forced unmount of /alt_inst
Fixing LV control blocks...
Fixing file system superblocks...
```

Chapter 6. User Management

Managing system users is one of the primary tasks of an administrator. Older UNIX applications were very often based on the terminal interface, and they required signing in to the system. Now, almost all applications use graphical interfaces, and the login process is controlled by a server-side application. In this case, there are usually only a few user accounts that the system administrators or applications use.

You can group and assign roles to operating system users. A user group is a standard structure that exists in every UNIX system, and it works in the same way. It allows you to create sets of users with specific permissions and hence simplifies the process of modifying permissions. Operating only on user and group privileges has many disadvantages. The main disadvantage is the inability to accurately allocate permissions to multiple user groups. This leads to very frequent root user operations via the *su* and *sudo* commands, as well as the frequent use of SUID and SGID bits (which will be discussed later in this book). In such a case, there are roles that provide far-reaching enhancements to the traditional UNIX management of privileges. These roles are discussed in detail in the Security chapter.

Users, their permissions, and their passwords are usually stored locally on UNIX systems. This means that each system has its own definition of users and groups. This is often a problem for large organizations with hundreds of such installations. The solution to this problem is centralized user management, for example, using the LDAP or another centralized system. AIX can work with the LDAP in this area, but in this chapter, we will focus on managing users from a single-system perspective.

Creating users

The most convenient way to create a user is to use the SMIT utility. In AIX, there are a wide range of parameters describing a user account. Most such parameters receive default values if they are not explicitly defined. This way, the only parameter required when creating a user account is its name.

User data are not stored, as you might suppose, in an ODM database. They are stored in several system files, some of which, such as */etc/passwd* and */etc/groups*, represent the standard for storing user information in UNIX systems.

In Figure 6-1, which is part of the user creation screen of the SMIT menu (*smitty mkuser*), you can see how the parameters are set to the files in the operating system.

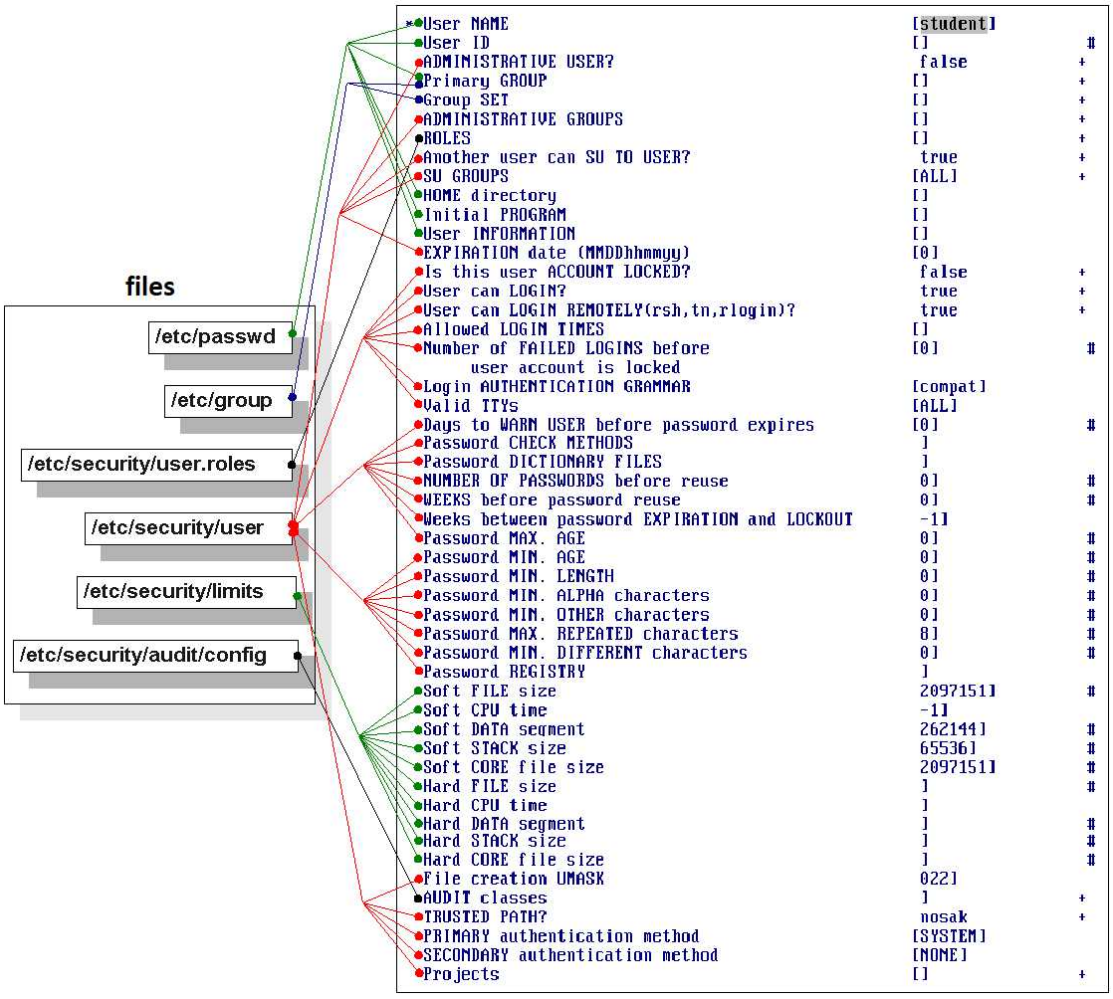

Figure 6-1: smit mkuser - Parameters.

The meaning of the majority of relevant parameters is clear, so there is no need to discuss them. However, those that do require further comment are discussed below.

User ID - Usually, you do not enter a value here, which causes the system to automatically select another free number. Please note that the main parameter identifying the user is simply that number. If you remove the *student* user who has the number *303* but leave his files on the system, they will be visible as *303* user files (with *303* as a user name). Sometimes, it is worthwhile to maintain the consistency of the numbering of users on several systems. This is especially true for clusters or operating systems working on common file systems.

Primary GROUP - The group to which the user belongs. With standard permissions, each file can belong to only one user and one group. The file that the user creates will belong to his primary user group.

Group SET - Additional groups to which the user belongs. The user has access to the files that belong to the groups mentioned here.

Initial PROGRAM - The operating system shell in which the user should work, which by default is */usr/bin/ksh*. Instead of a shell, you can specify the path to the application execution file and thus allow the user to work with the application without having the ability to navigate the operating system. Each program you want to substitute in this option must be listed in the */etc/shells* and */etc/security/login.cfg* files.

EXPIRATION date - Specifies the final time the account can be used. After that time, access will be denied.

Is this user ACCOUNT LOCKED? - Determines whether or not the user account is to be locked. This option is also found in the menu that allows you to change user parameters. Further, this menu allows you to unlock your account, but not when it is locked due to too many unsuccessful login attempts. When this happens, you can unlock the user using the *Reset User's Failed Login Count* menu (*smit failed_logins*).

Allowed LOGIN TIMES - Determines when a user can log on to the system (days of the week, range of hours).

Valid TTYs - The terminals from which the user can log in. This option is important when you log in from a console directly connected to the server. If you log in to the system via Telnet or SSH, it does not matter. In such a case, each user session can be assigned a different virtual terminal number.

Password CHECK METHODS - Allows you to specify any additional requirements that the user password must meet. Each time you change a password, the program indicated by the entry in this field is called to verify that the password meets the specified requirements.

Password DICTIONARY FILES - A comma-separated list of files that contain a list of words that cannot be used as a password.

Password MAX. AGE - The time after which the user cannot log in without changing the password (the user will be prompted to change the password after logging in using the old password).

Password MIN. AGE - The minimum time after which the user can change the password.

Soft FILE size - The maximum file size in 512-byte blocks that a user can create. You can manually increase this parameter for the current session using the *ulimit -f size* command, although you cannot exceed the *Hard FILE size*. If you reach the maximum value, the process will be terminated with the message "*A file cannot be larger than the value set by ulimit*."

Soft CPU time - The maximum time (in seconds) that the CPU can process each user process. This is an effective processing time (for example, the process can run for two days and have an effective processing time of one second). You can manually increase this parameter using the *ulimit -t number_of_seconds* command, although you cannot exceed the *Hard CPU time* value.

Soft DATA segment - The maximum size of a data segment in 512-byte blocks that any user process can use. You can manually increase this parameter using the *ulimit -d size* command, although it cannot exceed the *Hard DATA segment* value.

Soft STACK size - The maximum size of a stack in 512-byte blocks that any user process can use. You can manually increase this parameter using the *ulimit -s size* command, although it cannot exceed the *Hard STACK size* value.

Soft CORE file size - The maximum size of a *core* file in 512-byte blocks that can be created by the user process. The *core* file is created due to an erroneous process termination, and it contains a process memory image that helps diagnose the problem. You can manually increase this parameter using the *ulimit -c size* command, although it cannot exceed the *Hard CORE file size* value.

File creation UMASK - Specifies the default permissions for user-created files. The file mask is given in octal notation, as in the *chmod* command. The difference is that a bit with a value of *1* indicates the lack of a given privilege, while a bit with a value of *0* indicates its presence. Let's look at the example of the mask 022. When you write these numbers in binary form, you get 000 010 010. It indicates full rights for the owner (000), read and execute rights for the group (010), and read and execute rights for the rest of the users (010). So, the rights to the file from the *ls* command perspective will look like this: rwxr-xr-x.

PRIMARY authentication method - Specifies the user authentication method. The default *SYSTEM* setting specifies traditional authentication based on the */etc/passwd* file. You can also authenticate users using the LDAP, which makes it easy to manage users in large organizations.

AUDIT classes - Specifies the auditing class to be assigned to the user. Selecting this option modifies the */etc/security/audit/config* file.

Keystore/File Encryption Algorithm - Algorithms to use for encrypted file systems (EFS). Encrypted file systems are described in more detail in the Security chapter.

The management of user groups is very easy. The main parameter associated with it is the *ADMINISTRATOR list*. This specifies the users who, in addition to the root user, can manage the group (add and remove users from the group).

Change user parameters

You can change the user parameters using the *smit chuser* menu. This menu is identical to that described in relation to creating user accounts, so you can modify almost all the parameters available when creating a user.

Another useful option available on the menu is to lock and unlock users (*smit lockuser*). Typically, UNIX systems use the principle of non-deleting users, but instead locking them. This offers the advantage that the identity of owner of the files that exist within the system is always known. The following example shows the output of the *ls -l* command on a file whose owner has been removed from the system. As you can see, instead of a username, there is a number.

```
# ls -l /home/testuser/smit.log
-rw-r--r--   1 8        system         44898 Oct 19 02:24 /home/testuser/smit.log
```

This number represents the owner of the file. It appears because in the I-node (the record describing the properties of the file), its owner is represented by a number rather than a name. The number is

mapped to the username using the */etc/passwd* file. If you delete a user, its record is deleted from */etc/passwd*, so it becomes impossible to map the identifier to the name and, as a result, you lose the information about who owned the files.

When creating a user account, you specify a parameter that specifies how many unsuccessful login attempts lead to account lockout. An account locked in this way cannot be unlocked through the *Lock/Unlock and User's Account* menu (*smit lockuser*). It can only be unlocked by resetting the failed login counter via the *Reset User's Failed Login Count* menu (*smit failed_login*).

User login process

The user login process for the operating system, whether via a console directly connected to the server or via SSH, looks very similar. After you type in the username and password, the operating system checks to see whether the user exists (*/etc/passwd*), whether you've typed a valid password (*/etc/security/passwd*), and whether the user has login rights.

Upon the successful completion of the above process, the operating system configures the user environment. It reads the parameter files and runs several scripts that set the relevant variables and parameters for the user. These files can be freely modified. You can make changes to customize the user environment to meet the user's changing needs. The files are processed in the following order:

- ***/etc/profile*** - A script processed during every user login. The variables it sets apply to all users. The most important parameters set by default in this file are the terminal type (variable *TERM*), which is responsible for displaying on-screen data and the e-mail message (*MAILMSG* variable).
- ***/etc/environment*** - A file with predefined environment variables that is processed during each user's login. As the file name indicates, only environment variables should be included in it. The most important variables set by default in this file are:
    - ***PATH*** - Specifies the paths that the operating system searches for if you operate on files without providing a direct path to them.
    - ***TZ (Time Zone)*** - Specifies the time zone in which the server is running. Thanks to this variable, the operating system presents the time that is appropriate for the particular zone, and it knows when to change the time from winter to summer (and vice versa). This does not change the time counter on the server, but when read, it is presented according to the settings of the TZ variable. If you change the time zone, then the system commands and functions will automatically display another time, which is appropriate for the zone.
    - ***LANG*** - The language used by the operating system to communicate with users. This variable specifies, for example, the language of the manual (*man*), the language of system utilities (*smit*, *errpt*, etc.), and the language in which the system displays messages.
- ***$HOME/.profile*** - A hidden file located in the home directory of each user. It is processed when using the default system shell (*ksh*). It is used to customize the user environment, which often means overriding some of the variables defined in the */etc/profile* file.
- ***$HOME/.env*** - A hidden file located in the home directory of each user. If the *$HOME/.profile* file contains the string "*export ENV=$HOME/.env*," it has the same meaning as the */etc/environment* file, but it applies to one user rather than to all users.

If you use the *csh* shell, other files are used:

- */etc/csh.cshrc*.
- */etc/csh.login*.
- *$HOME/.cshrc*.
- *$HOME/.login*.

After the system has processed the above files, a typical user is logged on, has a defined environment, and has access to the command line.

You have the ability to define the scope of user access to the operating system using the *.profile* file. For example, you can configure the user profile so that the user does not have access to the command line, but only to the terminal application that launches when the *.profile* file is being processed. Alternatively, the user can have access to a menu that you define in order to give him access to specific functions. You can disable the interrupt signals to prevent the user from exiting to the shells. You can do this by using the *trap* command, which is an internal shell command.

The command is referred to as internal because it does not exist as an executable file, but is instead part of the system shell (*ksh*, *csh*, *bsh*, or other). It allows you to define a response to the signals that a user can send (for example, *Ctrl+C*, *Ctrl+Z*, etc.) while processing a *.profile* or any script. An example of using the *trap* command follows:

```
# cat .profile
trap 'kill -16 $$' 1 2 3 15
PATH=/usr/bin:/etc:/usr/sbin:/usr/ucb:$HOME/bin:/usr/bin/X11:/sbin:.

export PATH

if [ -s "$MAIL" ]           # This is at Shell startup.  In normal
then echo "$MAILMSG"        # operation, the Shell checks
fi                          # periodically.

./program
exit
```

The first line of the above file means that the 1, 2, 3, and 15 signals that will be generated during the processing of the *.profile* file will be intercepted and kill the shell process of our session. Each of these signals could interrupt the script processing, in this case the *.profile* processing, and enable the user to work on the command line. Due to the signals being intercepted, the user will not be able to interrupt the script. Hence, the script will either process completely or the shell process in which it is running will be killed. In the above example, both cases will result in the end of the session (exit at the end of the script or kill -16 if you try to break it).

You can use *kill -l* to check the signals that can be directed toward any process. The means by which the process handles this signal has been decided by the developer during the creation of the corresponding program. If the developer did not consider the support needed for the signal, it will be ignored (this does not apply to signal 9, which causes the process to stop completely).

Tracking the user

If you need to fully track a user, enable auditing on the system and specify the behavior to be tracked for specific users. Auditing issues are described in the Security chapter. The following are the typical log files that work independently of the audit system.

There are several files that collect information about user activity in the operating system:

- ***$HOME/.sh_history*** - A file that contains a list of recent user commands based on the *sh* shell. This file is intended to be used by its owner, although it gives the system administrator the ability to monitor user activity. In the context of monitoring, this file has one disadvantage, namely the fact that the user has full access to it. It is a text file, so the user can freely change its contents.

  To obtain full information about the user's behavior on the system, you need to perform use auditing. If you do not use auditing, you can run a process that constantly monitors the contents of the *.sh_history* file. Such a process should continuously save the lines that appear in the history file to another file that the user does not have write access to. An example of monitoring process:

```
# nohup tail -f /home/testuser/.sh_history > /var/adm/users/testuser.history &
[1]     237752
```

  In the above example, the *nohup* command causes the *tail* command to remain in the system after the session closes (otherwise it would be terminated at the end of the session). The *tail -f* command traces the new lines that appear in the file and writes them to the standard output. The new lines are redirected (>) to the appropriate file.

- ***/var/adm/wtmp*** - A file that contains login information. From this file, you can obtain the following information: when, from what IP address, and for how long the user was logged on. The file is binary. You can use the *last* command to view its contents, for example:

```
# last | grep testuser
testuser     pts/1        10.10.254.241          Oct 19 03:22   still logged in.
testuser     pts/1        10.10.254.241          Oct 19 03:21 - 03:21  (00:00)
```

  As you can see, *testuser* has logged into the operating system twice, and one of his sessions is still active.

  • ***/etc/security/lastlog*** - A file that contains detailed information about recent successful and unsuccessful user attempts to log in to the system. Of particular importance here is the information about the IP address from which the login occurred as well as the number of unsuccessful logins (*unsuccessful_login_count*). Please note that if the number of unsuccessful login attempts exceeds the value specified when creating the user account, the account will be locked until the failed logins counter has been reset (*smit failed_logins*).

```
# cat /etc/security/lastlog
# All unnecessary information has been omitted.
test:
        time_last_unsuccessful_login = 1476865241
        tty_last_unsuccessful_login = /dev/pts/1
        host_last_unsuccessful_login = 10.10.254.241
```

```
        unsuccessful_login_count = 0
        time_last_login = 1476865322
        tty_last_login = /dev/pts/1
        host_last_login = 10.10.254.241
# All unnecessary information has been omitted .
```

- ***/var/adm/sulog*** - A file that contains information about using the *su* command, which allows you to switch to another user, including the *root* account (the *root* user can switch to any user without entering a password).

  For security reasons, the ability to directly login to *root* on UNIX operating systems is usually limited. Therefore, the *su* command, which allows you to switch to this user, is a necessary element. Theoretically, you could eliminate the need to switch to the *root* user by using the *sudo* command. In some cases, you can also stop using your *root* account and manage your system using the *Role Based Access Control (RBAC)*. In practice, however, many organizations still use the switch option, since this usually improves system management.

By reviewing the above file, you can verify successful and unsuccessful switches to individual users:

```
# tail -3 /var/adm/sulog   # „tail -3” displays the last three lines of the file.
SU 10/19 03:15 + pts/0 root-test
SU 10/19 03:39 - pts/2 test-root
SU 10/19 03:39 - pts/2 test-root
```

In the example above, the + character indicates a successful attempt, while the – character indicates a failed attempt to switch the user. For instance, the first line shows that the *root* user successfully switched to the *test* user. The following line shows an unsuccessful attempt to switch from the *test* user to the *root* user.

Files related to user management

There are many files related to user management in UNIX systems. You do not need to know about these files if you manage the system with utilities (SMIT menus, dedicated commands). However, if there is a problem in the operating system, such knowledge may be crucial to identifying its solution.

The two main user management files are */etc/passwd* and */etc/group*:

- ***/etc/passwd*** - Stores information about users on UNIX systems. Every user who exists in the system is described in this file by one line. The individual user information is separated by a "*:*" and the following content is provided:

  username: password (currently in another file): user id: group id:: home directory: user shell

```
# cat /etc/passwd
root:!:0:0::/:/usr/bin/ksh
daemon:!:1:1::/etc:
# All unnecessary information has been omitted.
student:*:202:1::/home/student:/usr/bin/ksh
```

In older UNIX systems, the password was hashed directly into the */etc/passwd* file. Hashing is a one-way function, so even if you have a hashed version of your password, there is no algorithm for recovering a password from it. Passwords protected in this way can only be cracked using *brute force*. You can also use the dictionary method based on the most commonly used passwords.

As each user has the right to read the */etc/passwd* file, one of them could read another user's password and try to crack it. To prevent users from reading the passwords, they have been moved to the */etc/security/passwd* file. Only the *root* user can access this file. In place of the password in the */etc/passwd* file there is always a "*!*" or "***" character.

User files in file system structures are marked with a user ID rather than a username. The file display commands convert the identifier into a name based on the entries in the */etc/passwd* file. Therefore, if you delete a user and leave his files, you can obtain the information that the owner of the file is a number (i.e., 324).

The primary group ID specifies the group number to which the user-created files belong. The name of the group described by this identifier can be found in */etc/group*.

- ***/etc/group*** - A file that stores user group information in UNIX systems. Each user group in the operating system is described by one line. The information is separated by a "*:*" and the following content is provided:

  group name:: group id: members of the group

```
# cat /etc/group
system:!:0:root,srvproxy,esaadmin
staff:!:1:ipsec,srvproxy,esaadmin,test
# All unnecessary information has been omitted.
```

Additional files containing user information and their properties are found in the */etc/security* directory. Only the *root* user has full access to this directory. Those users who are members of the *security* group have read permission. The most important files are (all the files listed below are text files):

- ***/etc/security/passwd*** - Contains hashed user passwords.
- ***/etc/security/group*** - Contains additional user group attributes. These attributes are to enable or disable the ability to administer a group by a non-root user and a list of group administrators.
- ***/etc/security/user*** - Contains additional attributes for all users. Each user has a section in this file that contains those attribute values that differ from the default values in the default section.
- ***/etc/security/login.cfg*** - Contains logon parameters, that is, the applicable shell, the maximum number of sessions, the time (seconds) available to enter the password at logon, and the user authentication method on the system.
- ***/etc/security/limits*** - Contains the system resources limit for users. Each user has a section in this file that describes those parameters that differ from the default parameters in the default section. Examples of the parameters that can be specified here are the maximum file size, maximum amount of processor time that a process can use, etc.
- ***/usr/lib/security/mkuser.default*** – Contains the default parameters that are assigned to

newly created users when no other parameters are given during creation. The default parameters listed here can be freely modified. In this way, you can improve the process of creating user accounts in your system. This is important when creating many user accounts, although that is currently a rather rare situation.

- ***/usr/lib/security/mkuser.sys*** - A script called by the *mkuser* command. It creates the home directory of the user with the appropriate access rights and basic profile. This file can be customized to suit your needs.

Chapter 7. Storage Management

Storage management, as with most aspects of IT, is constantly evolving and improving. During the earlier stages of IT development, servers had their own disks, and they operated on them directly and exclusively. This had a number of advantages, such as the lack of impact of other systems on the server's disk performance. It also had disadvantages, such as the low scalability or the lack of the possibility to use advanced High Availability and Disaster Recovery solutions (data replication). In addition, the unused space could not be used for other servers or other purposes. At a certain stage of development, the concept of a network in disk space management appeared. In this way, technologies such as the SAN (Storage Area Network) and iSCSI, which use the Local Area Network for data transmission, were developed.

The use of SANs involves the use of central and intelligent storage systems. It enables the sharing of disk space across multiple servers. Space that is no longer needed by one system can be allocated to another. This may include advanced data management methods. This appeared to be the only direction of development for a while, although it is now being corrected by the introduction of ever more efficient solid-state drive (SSD) solutions. The minimum latency that SANs introduce into data transmission is irrelevant in spinning disk drives (HDD) that have a high latency by default. However, it is important when using SSDs, where the response time has dropped drastically. Hence, in very specific solutions that do not benefit from SANs, we still use direct attached disks.

Previously, the AIX disk space was fully configured on the operating system side. Today, you mainly operate on LUNs (logical disks) created on the storage system and shared by the SAN. In such a case, the configuration of the space is usually beyond the reach of the server or operating system.

Regardless of where your disk space is set up, it is important to understand its basic structure and characteristics. The primary standard for disk space configuration is RAID (Redundant Array of Independent Disk). Its implementations are found on many storage systems and in many operating systems. For disks that are directly attached to the server, a dedicated hardware controller usually handles the RAID array. With RAID, you can simultaneously write and read data from multiple disks as if you were operating on only one, thereby increasing the I/O performance. Moreover, it allows you to introduce some redundancy that makes your set of disks less susceptible to data loss in the event of a hardware failure.

RAID (Redundant Array of Independent Disk)

There are several different RAID implementations. Each of them has both advantages and disadvantages, so an implementation that works in one situation might not be appropriate in another. Descriptions of the most important implementations are given below.

RAID 0 - Striping

Level 0 does not actually deserve a RAID name because it has no redundancy, which is indicated by the letter "R." Data are written and read at the same time from all the disks in the array. The minimum number of disks in array is two.

When creating this structure within the operating system using a RAID controller, a virtual disk is created. Its size can be represented as the number of disks multiplied by the smallest disk size. Therefore, it is best to use disks of the same size (besides, not every controller can build RAID on disks of different sizes).

This virtual disk is divided into chunks (known as stripes), the size of which you can usually define when creating a structure. The following chunks are created by the round robin algorithm on all the disks involved, as shown in Figure 7-1.

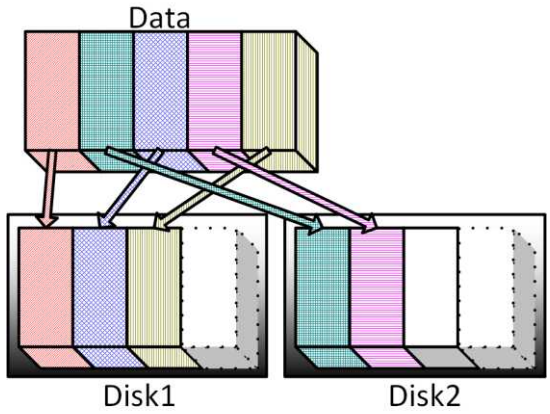

Figure 7-1: RAID 0 - Striping.

With a 64 KB stripe and two disks in RAID 0, a 128 KB file will be located on two disks. There will be 64 KB on the first disk and 64 KB on the second. This will result in significant acceleration, since the read or write operation of this file can be performed by two disks simultaneously.

Advantages:

- Significant write and read performance.
- Simplicity - The algorithm for performing disk operations is trivial, which renders software implementations as efficient as hardware.

Disadvantages:

- No redundancy - The failure of one disk causes the loss of all data from the array. This disadvantage limits the use of this solution in most cases.

Solutions:

Typically, this is not the case for servers running single, directly attached disks. You can use this RAID implementation successfully if you work on LUNs created on a storage system. Such LUNs have disk failure protection on the storage system side. You can also use this solution if failure tolerance is of only minor importance and the only criterion is speed.

RAID 1 - Mirroring

Level 1 is fully redundant, meaning that it represents perfect protection against data loss. This has led to its frequent use as a means of protection for system disks in servers. RAID 1 is made up of two disks. If you want to protect more disks, you should create several independent RAIDs of this type.

When creating this structure in the operating system using a RAID controller, a virtual disk representing the mirror is created. The created disk is the size of the smallest disk used to build this structure. Since we usually use disks of the same size, we can say that after building a mirror we will have half the disk capacity.

Both physical disks are divided into the same number of stripes of the same size. Each stripe on the first disk has a copy on the second disk. This means that the number of write operations is doubled, since the same thing must be written on both discs. This is not the case with read operations that can be shortened by dividing a task between two disks. The diagram of RAID 1 is shown in Figure 7-2.

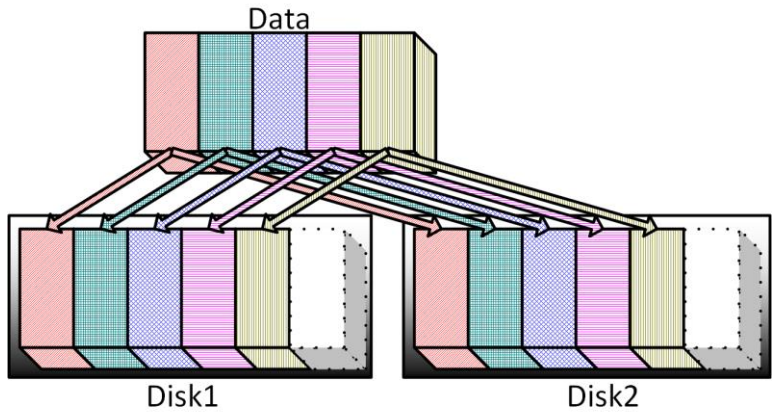

Figure 7-2: RAID 1 - Mirroring.

Let's look at this example in more detail. The stripe is 64 KB in size, and we have a mirror made of two disks. A 128 KB file will be located on both disks. In total, 256 KB will be written. Most writes will be performed in parallel on two disks, which will lengthen the write operation. This will occur due to sending twice as much data through the bus and waiting for the write to end on the slower disk. The read can be performed at high speed, which is comparable to reading in RAID 0, since the array is able to divide the read operations between the two disks. Both disks contain the same data, so there is nothing to prevent you from reading part from the first disk and part from the second disk.

Despite the fast readings, the double write makes RAID 1 slow in terms of writing, although it is fault tolerant.

Advantages:

- Full redundancy - As two copies of the data are stored at the same time, there is little chance of losing that data.
- Simplicity - The algorithm for performing disk operations is trivial, which renders software implementations as efficient as hardware.

Disadvantages:

- Very high cost of redundancy - Only half the disk capacity in the array is used.
- Low write performance - Due to the need to perform double write operations, the performance of such an operation is lower than the performance of a single disk.

Solutions:

When operating on internal disks, system disks are usually protected in this way. They do not need to exhibit high performance, but they do require good protection in case of failure. If the installation is virtualized, then the Virtual I/O Server system disks are protected in this way. Another factor that makes this ideal for system drives is the simplicity of operations on such a structure, as well as the fact that a single disk from such a RAID can be treated as a stand-alone unit rather than as a part of a RAID. This feature simplifies the code that is responsible for booting the system, since it can limit itself to operating on a single disk at boot time until the code responsible for the mirror is loaded. The code does not need to have algorithms to read data from the RAID. It just needs to read the kernel from one of the drives and give it control. The kernel can already operate on the entire RAID.

RAID 10 - Striping and mirroring

RAID 10 is a combination of two RAIDs, that is, RAID 1 and RAID 0. It has full redundancy inherited from level 1 and high speed inherited from level 0. This type of RAID is the fastest of all the levels. It is very well protected against the loss of data due to failure, but at the cost of sacrificing half the total disk capacity.

To build a RAID of this type, you must have at least four disks, usually of the same size. After creating such a RAID, a virtual disk half the size of the capacity of the disks is created.

This virtual disk is divided into chunks (known as stripes), the size of which you can usually define when creating the RAID. The next chunks are created by the round robin algorithm on half the disks. In addition, each stripe has its own copy on a different disk, which protects the system against the loss of data due to a single disk failure. The four-disk RAID 10 configuration is presented in Figure 7-3.

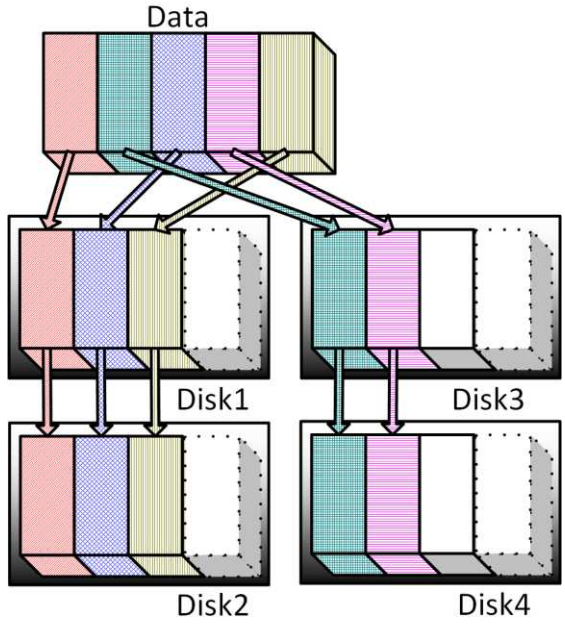

Figure 7-3: RAID 10 - Mirroring and striping.

Let's look at this example in more detail. With a 64 KB stripe and four disks in RAID 10, a 128 KB file will be located on two disks, and each stripe will have a copy on another disk. By writing 128 KB, we need to physically write 256 KB. Despite this, we obtain the acceleration of the operation, since the writes are distributed between the four disks.

Advantages:

- Significant write and read performance - If we consider the read and write speeds, we can say that this is the fastest RAID of all the implementations that provide data protection (only RAID 0 is faster).
- Full redundancy - Due to storing two copies of the data at the same time, there is little chance of losing that data.
- Simplicity - The algorithm for performing disk operations is trivial, which makes software implementations as efficient as hardware.

Disadvantages:

- Very high cost of redundancy - Only half the disk capacity in the array is used.

Solutions:

This is applicable when high read and write performance is required. It is very often used to store database files. Consider the fact that when using SSDs, it offers no such advantage over RAID 5 as with HDDs. In many operating systems, you can create this level of RAID using software and achieve good performance.

RAID 5 - Striping, distributed parity

You can build the smallest RAID 5 on just three disks. Usually no more than 8–16 disks are used. RAID 5 has a redundancy that renders it immune from the failure of one disk from a defined group. The failure of the next disk causes a complete loss of data from the RAID. It can be very efficient at reading, although it can work slowly in the case of writing. Sequential writes of large data volumes are quite efficient; however, random small writes can cause a dramatic drop in performance.

RAID 5 is based on algorithms that calculate parity. The parity is disposed between all the disks in such a way that the loss of one of them does not result in the loss of data. The performance of one of the array disks is correct, albeit less efficient due to the need to calculate the data that should be on the inaccessible disk. The calculation is based on both the data and the parity of the available disks. Due to the calculations made during the operation of this RAID, it is unlikely to be used as software. Such an implementation would be much slower than a hardware one, and it would result in a waste of computing power.

To create a RAID of this type, you must have at least three disks, usually of the same size. When a RAID is created, a virtual disk is created of the size of the total capacity of all the disks minus the capacity of one disk.

The virtual disk is divided into chunks (known as stripes). Its size can usually be defined when creating a RAID. The next chunk is created using the round robin algorithm on subsequent disks. A few stripes form a group (stride), each consisting of one stripe of each disk. One stripe of each group holds parity. The parity stripes are evenly distributed in order to balance the load of all the disks. An example of such a RAID built on three disks is shown in Figure 7-4.

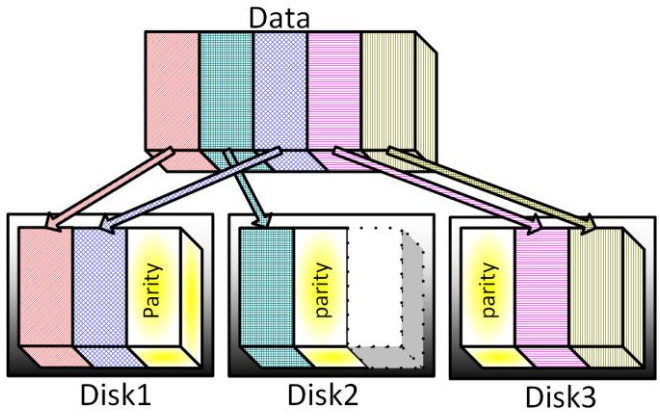

Figure 7-4: RAID 5.

Let's look at this example in further detail. With a 64 KB stripe and three disks in RAID 5, a 128 KB file is located on three disks. Two have the file, while the third has parity. In total, 192 KB is saved. Each change of one portion of the data stored on the disk causes two changes to be made, since new parity information must be calculated and written. Typical readings can be made at very high speeds, as with RAID 0, because it does not force parity updates.

Advantages:

- Significant read performance.
- Low cost of redundancy - You lose the capacity of only one disk from all the disks used to build the RAID.

Disadvantages:

- Poor write performance - This is especially the case with many small operations.
- Processor load determined by a parity calculation algorithm - The parity calculation algorithm causes an additional processor load. This behavior causes a significant loss of performance when writing, especially for many small write operations.

Solutions:

It is used wherever the economic aspect is important, since it uses disk space more efficiently (we lose the capacity of only one disk from all those used in RAID 5). However, it is not typically used in systems that require very high write performance. RAID 5 is often used in application servers and web servers because, in this case, the main operations on the disks are read operations. Yet, it is rarely used for database servers. The use of SSDs makes the disadvantages of RAID 5 less noticeable. This means that it is applicable for all types of use cases.

Additional information

Historically, there are also RAID levels between 1 and 5, namely RAID 2, RAID 3, and RAID 4, although they are inefficient and practically unused. RAID 2 was created to use drives that were unable to verify the readability of the data itself (currently all drives can do this), while RAID 3 and RAID 4 are RAID 5 predecessors. They work less efficiently than their younger brother, and their main issue is the fact that parity is written to a dedicated disk, which causes excessive disk loading and results in performance loss.

There is also a RAID 6 that is designed to provide more data protection than RAID 5, while still providing good disk space utilization. It works very similarly to RAID 5, except that in one group (stride) there are two stripes for control data. This means that the RAID is immune to the simultaneous loss of two disks. The reason for using this type of RAID is to use large and slow disks. In case of a disk failure, it is necessary to replace and rebuild the RAID.

In the case of large and slow disks, rebuilding can take many hours and heavily load the remaining disks, increasing the risk of further failures and the inaccessibility of data. RAID 6 counters this kind of risk.

Disk arrays and internal RAID controllers allow you to define one or more disks as a "Hot Spare." If one of the disks in the array fails, it is replaced by the "Hot Spare" disk. This process can take a long time, since the replacement drive must be written with the data that was on the damaged disk. In the case of RAID 5, the data are calculated from the information stored on the remaining disks, while for RAID 1 and 10, the data are copied from the second working copy.

Comparison of the RAIDs

When comparing the different RAIDs in terms of their performance, we can conclude that the fastest are RAID 0 and RAID 10. If you use RAID 5 to read only, it can be as fast as RAID 10. The performance comparison is presented in Table 7-1.

Table 7-1: RAID performance comparison.

	Read	Write	Overall Performance
RAID 0	Very Good	Very Good	Very Good
RAID 1	Very Good/Good[1]	Poor	Poor/Good
RAID 10	Very Good	Very Good	Very Good
RAID 5	Very Good	Poor[2]	Good

[1] Depends on the implementation, since the operating system can split read jobs between disks in RAID 1 or always read from one disk.
[2] With large blocks of writes, performance may significantly increase, although performance decreases dramatically with small blocks of writes.

The reasons for this difference in performance between RAID 10 and RAID 5 are illustrated in the following scenario. A 64 KB data block is written in RAID 10 and RAID 5. Assuming both are built on four disks, the stripe size is 64 KB.

RAID 10
User writes a block that resides on one pair of disks, so two operations are performed:

1. Write the data block on the first disk = 1 I/O.
2. Write the data block on the second disk = 1 I/O.

So, a single write from the operating system perspective requires two write operations.

RAID 5
User writes a block of stripe size. So, in a RAID 5 built on four disks, the following operations are performed:

1. Write the data block on the appropriate disk = 1 I/O.
2. Read the corresponding data blocks from the other two disks = 2 I/O.
3. Calculation of parity.
4. Write the data block with parity to the third disk = 1 I/O.

A single write from the operating system perspective requires two write operations, two read operations, and a calculation. This scenario also shows why a sequential write of large amounts of data in RAID 5 can prove efficient. In such a case, the read operations are eliminated, since all the stripes in each group (stride) are modified and only the parity needs to be calculated.

From the fault tolerance perspective, RAID 1 and RAID 10 have the lowest risk of data loss. The fault tolerance of each RAID is shown in Table 7-2.

Table 7-2: Failure resistance of the RAIDs.

	Minimum number of disks	Failure resistance	Failure of one disk	Failure of 50% of disks
RAID 0	2	Lack	Loss of data	Loss of data
RAID 1	2	High	Data available	Data available[3]
RAID 10	4	High	Data available	Data available or not[4]
RAID 5	3	Average	Data available[5]	Loss of data[6]

[3] Since this RAID is usually built from two disks, a 50% failure implies the failure of one disk.
[4] Depends on which disks fail.
[5] The array is in degraded mode. The missing data are calculated, which can cause a significant loss of performance.
[6] The failure of more than one disk causes complete data loss.

Disk management in AIX

We often use a SAN and external storage to manage disk space. External storage devices are smart devices, and they provide a variety of configuration capabilities in terms of both performance and availability. They share logical drives for the operating system. The operating system manages them using its internal mechanisms.

AIX has its own internal virtualization layer to manage the available disk space from its perspective. This layer is quite complex and flexible. By using it, we can examine every component of the disk subsystem, as well as verify and modify data placement without an interruption in system operations. The disk management mechanism is known as the **Logical Volume Manager (LVM)**. It has many advantages:

- It allows you to change the file system size. In traditional file systems, resizing a file system is a complicated task. It requires deleting the partition on which the file system is located, creating a new one with its new size, and restoring all the files on it. In case of AIX, the file system size can be changed with one command during normal operation, and the task takes only a few seconds.
- Logical volumes (equivalent to the partitions in traditional file systems) do not have to be placed on continuous space on the disk. They do not have to be on one disk; they can be scattered on many of them.
- It allows you to easily perform various disk operations, such as relocating structures, changing their parameters, creating RAID 0, 1, and 10 configurations, and so on.
- By operating on multiple disk arrays, it is possible to relocate data between the arrays without generating system inaccessibility.

Disk management is concerned with managing the various components that make up the disk system. There are a number of basic structures within disk management:

- **Volume Group (VG)** - The parent structure over all others. Group physical volumes (disks), thereby defining some of their characteristics.
- **Physical Volume (PV)** - Simply put, this is the entire physical or logical disk (LUN) provided

to the operating system by a disk array or internal RAID controller. The physical volume may belong to a volume group. In such a case, it is partitioned into fixed-size physical partitions, the size of which is determined by the parameters of the volume group. It is visible in the operating system as */dev/hdiskX*. It is important to keep in mind that in this book, in most cases, the term "disk" is synonymous with the term "physical volume."

- **Physical Partition (PP)** - A fixed portion of a physical volume. Due to being divided into physical partitions, the logical volume and thus the file system do not need to be located on a physical volume in the form of a continuous area. Resizing a logical volume and file system involves adding or removing a PP.
- **Logical Partition (LP)** - The logical equivalent of a PP, and logical partitions are mapped to physical partitions. One LP always points to one, two (mirroring), or three (double mirroring) physical partitions on one or more physical volumes.
- **Logical Volume (LV)** - A structure consisting of one or more logical partitions. The logical volume can be compared to a regular partition in any operating system. The difference is that the LV does not have to be a continuous area. It consists of a set of "blocks" (LP/PP) that can be placed on different physical volumes. File systems are created on logical volumes.
- **File System (FS)** - A structure that directly stores our data is created on the LV. You can only create one file system per logical volume.

The Logical Volume Manager structures, as well as the relationships between them, are displayed in Figure 7-5.

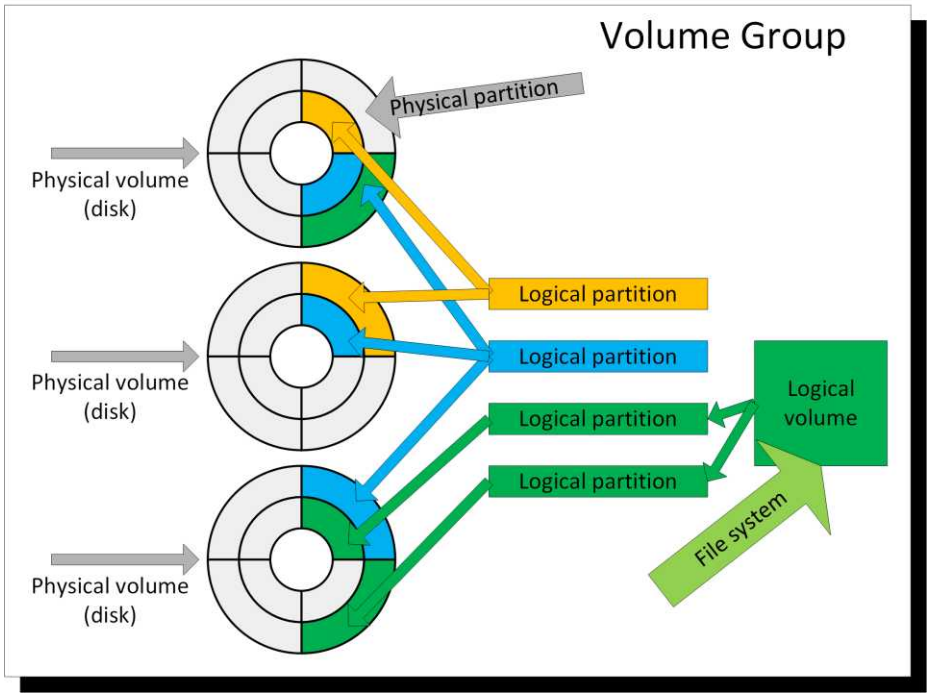

Figure 7-5: Disk structures of AIX.

Information about volume groups and logical volumes is stored in two places:

- **Object Data Manager database (ODM)** - The ODM database was described in previous chapters.

- **Volume Group Descriptor Area (VGDA)** - Each physical volume belonging to this group. It is a separate area that exists on each physical volume that holds configuration information for the entire volume group. This area lets you, for example, move a whole volume group to another server. You can do so by moving the disks and importing volume group data from the VGDA area of one of the disks (imported data is stored in the ODM).

Volume Groups (VG)

Volume groups are a basic logical structure that has control over stored data. This is a superior structure to the other structures, defining some parameters of its subordinate structures. Its directly subordinate structures are the physical volumes (PV), and all the structures which was indirectly created on physical volumes. One physical volume can belong to only one group.

A volume group must have at least one physical volume. When the last physical volume is removed from the group, the group is deleted. The group must be active (vary on) in order to allow access to its resources. Activating a group can be compared to mounting a file system.

Once the operating system is installed, there is a rootvg volume group that contains the disks selected for the system installation. This group should only be used to store the operating system. It is good practice to store data in separate volume groups. This ensures that the system is well organized and separated from the data. Such separation is of great importance in the case of the backup and recovery of data. Indeed, the backup chapter of this book explains why separation is so important for backups.

There are several types of volume groups in AIX. These types were created in AIX to meet the ever-increasing demand to increase the amount of data stored in a single group. In addition, the ways in which the different volume groups behave do not differ significantly. The differences between the various types of volume groups are presented in Table 7-3.

Table 7-3: Differences between the various types of volume groups.

VG type	Max. PV per VG	Max. LV in VG	Max. PP in VG	Max. PP size
Original VG	32	256	1016 (PP in PV) · 32 (PV) = 32512	1 GB
Big VG	128	512	1016 (PP in PV) · 128 (PV) = 130048	1 GB
Scalable VG	1024	4096	2097152	128 GB

The Big Volume Group was introduced in AIX 4.3.2, while it was Scalable Volume Group in AIX 5.3. Previously, in AIX 4.3.1, a change was made to allow more than 1016 PPs per PV in Original Volume Group. This is achieved at the cost of the maximum PV per VG. If you set 2032 PP in PV in the Original VG (that is, two times 1016), the maximum number of PVs per VG will decrease by half (to 16).

All new volume groups should be created as Scalable VG. Others exist for reasons of compatibility with earlier versions of the operating system. The properties of the volume groups are described by the parameters that you can assign to them. Use the *lsvg* command to obtain information about the volume groups and their parameters. You can view the characteristics of any group using this command.

Basic ways to use the *lsvg* command:

- ***lsvg*** - Displays information about all the volume groups.
- ***lsvg VGname*** - Displays the parameters of a volume group.
- ***lsvg -l VGname*** - Displays information about the logical volumes that belong to the volume group "VGname."
- ***lsvg -p VGname*** - Displays information about the physical volumes that belong to the volume group "VGname."
- ***lsvg -M VGname*** - Displays information about the distribution of the logical volumes of a given volume group. It also specifies on which physical partitions the logical volumes are located.

Parameters that describe the volume group

```
# lsvg
rootvg
datavg

# lsvg rootvg
VOLUME GROUP:       rootvg                   VG IDENTIFIER:  00f6217700004c0000000157d28ec368
VG STATE:           active                   PP SIZE:        16 megabyte(s)
VG PERMISSION:      read/write               TOTAL PPs:      511 (8176 megabytes)
MAX LVs:            256                      FREE PPs:       156 (2496 megabytes)
LVs:                12                       USED PPs:       355 (5680 megabytes)
OPEN LVs:           11                       QUORUM:         2 (Enabled)
TOTAL PVs:          1                        VG DESCRIPTORS: 2
STALE PVs:          0                        STALE PPs:      0
ACTIVE PVs:         1                        AUTO ON:        yes
MAX PPs per VG:     32512
MAX PPs per PV:     1016                     MAX PVs:        32
LTG size (Dynamic): 256 kilobyte(s)          AUTO SYNC:      no
HOT SPARE:          no                       BB POLICY:      relocatable
PV RESTRICTION:     none                     INFINITE RETRY: no
DISK BLOCK SIZE:    512                      CRITICAL VG:    no
FS SYNC OPTION:     no
```

Key parameters:

- **VOLUME GROUP** - The name of the volume group.
- **VG STATE** - The volume group state. There are three states:
  - **Active/Complete** - The group is fully active (the *varyonvg* command was run on it).
  - **Active/Partial** - The group is active, but some PVs are not active (they are probably damaged).
  - **Inactive** - The group is inactive, since the *varyonvg* command was not executed.
- **VG PERMISSION** - Read-only or read-write access privileges.
- **MAX LVs** - The maximum number of logical volumes that can exist in a volume group.
- **TOTAL PVs** - The current number of physical volumes in the volume group.
- **STALE PVs** - The number of damaged/not configured physical volumes in a group.
- **HOT SPARE** - Determines whether the group has a "hot spare" disk, that is, one that will

replace another disk if it is damaged.

- **PP SIZE** - The size of the physical partition. Each physical volume is divided into physical partitions of PP SIZE. The space for the logical volumes and file systems is allocated by chunks of that size. Therefore, this size specifies the minimum space that can be subtracted from or added to the logical volume. The size of the physical partition remains constant throughout the lifetime of the volume group, which means that it cannot be changed.
- **FREE PP** - The number of free physical partitions in the entire volume group (i.e., the amount of free disk space in the group).
- **USED PP** - The number of physical partitions that are used in the entire volume group, that is, the amount of disk space used in the group.
- **QUORUM** - Determines whether the quorum mechanism is on (2) or off (1). More information about this parameter is provided later in this chapter.
- **STALE PPs** - The number of damaged physical partitions.
- **AUTO ON** - This parameter determines whether the volume group is to be automatically activated upon system startup.

Logical volumes in a volume group (lsvg - l)

```
# lsvg -l rootvg
rootvg:
LV NAME             TYPE       LPs     PPs     PVs  LV STATE      MOUNT POINT
hd5                 boot       2       2       1    closed/syncd  N/A
hd6                 paging     32      32      1    open/syncd    N/A
hd8                 jfs2log    1       1       1    open/syncd    N/A
hd4                 jfs2       21      21      1    open/syncd    /
hd2                 jfs2       127     127     1    open/syncd    /usr
hd9var              jfs2       11      11      1    open/syncd    /var
hd3                 jfs2       70      70      1    open/syncd    /tmp
hd1                 jfs2       1       1       1    open/syncd    /home
hd10opt             jfs2       2       2       1    open/syncd    /opt
hd11admin           jfs2       8       8       1    open/syncd    /admin
lg_dumplv           sysdump    64      64      1    open/syncd    N/A
livedump            jfs2       16      16      1    open/syncd    /var/adm/ras/livedump
```

Key parameters:

- **LV NAME** - The logical volume name. You can name any new LV. After the installation of the operating system, the rootvg group always contains the logical volumes specified in the above output.
- **TYPE** - The type of logical volume. The five basic types are:
  - **JFS/JFS2** - Types that indicate that the logical volume is intended to hold a file system (JFS or JFS2).
  - **JFSLOG/JFS2LOG** - Types that indicate that the logical volume is intended to hold a file system log (JFSLOG or JFS2LOG).
  - **paging** - A type that indicates that the logical volume is intended to hold a paging space (also known as a swap space).
  - **boot** - A type that indicates that the logical volume is the Boot Logical Volume, which has the kernel loaded at boot time.
  - **sysdump** - A type that indicates that the logical volume is intended to hold a dump. A dump is an image of the state in which the system was at the time of the problem that

caused the failure. In the event of a failure, the system writes its status to this location.

The types are not strictly defined. When creating a logical volume, you can type any string into the type field. Therefore, the type should not be taken literally. The logical volume has the same structure regardless of the type it is assigned to.

- **LP** - The number of logical partitions allocated to a given logical volume. Use this value to calculate the size of the logical volume. By multiplying it by the size of the physical partition (PP SIZE), you get the size of the logical volume. The PP SIZE can be checked using the *lsvg VGname* command.
- **PP** - The number of physical partitions that make up a logical volume. By comparing the value of a PP with that of a LP, you can determine how many copies are stored. In the example above PP = LP = 1, so each logical partition has one physical partition. This configuration means that the data are kept in a simple form, without a mirror. It is possible to define a single or double mirror. You can verify that the data are mirrored using the *lslv -l LV name* command.
- **PV** - The number of physical volumes on which the physical partitions of a given logical volume are located. If a single mirror is used, the PP number will be twice as large as the LP number.
- **LV STATE** - The logical volume state. There are three possible states:
  - **opened/syncd** - The logical volume is open. This means that the file system that is created on it is mounted, or else some program uses this LV to bypass the file system. If the logical volume is configured as a mirror, both copies of the data are synchronized (*syncd*).
  - **opened/stale** - The logical volume is open. If the logical volume is configured as a mirror, the copies of the data are not synchronized (stale). This may indicate that the disk on which one copy of the data is located is corrupted. If that is not the case, you should synchronize both copies (for example, by using the *syncvg* command).
  - **closed** - The LV is closed. This means that no one is using this LV.
- **MOUNT POINT** - Shows the directory where the file system located on this LV is mounted by default.

Physical volumes in a volume group (lsvg -p)

```
# lsvg -p rootvg
rootvg:
PV_NAME     PV STATE    TOTAL PPs   FREE PPs     FREE DISTRIBUTION
hdisk0       active        542         433     108..87..21..108..109
hdisk1       active        542         427     108..81..21..108..109
```

Key parameters:

- **PV NAME** - The name of the physical volume.
- **PV STATE** - The physical volume state. It shows the state of a given physical volume. Access to the physical volume is only possible when it is *active*. It can also be in a *stale* state, indicating its inaccessibility or damage.
- **TOTAL PPs** - The total number of physical partitions in the physical volume.
- **FREE PPs** - The number of free physical partitions.
- **FREE DISTRIBUTION** - The number of free PPs in each region of the disk, starting from

the left:
- **outer edge**;
- **outer middle**;
- **center**;
- **the inner middle**; and
- **inner edge**.

In the case of physical disks, each of these parts differs according to the average data access time. During the creation of file systems, the administrator affects their performance by manipulating the distribution of the logical volumes on which they are located. In the case of disk arrays, the manipulation of these parameters does not make sense, since we have no knowledge of the physical characteristics of a LUN. The same holds true for SSDs, where data distribution is irrelevant. In such a case, overwritten data may appear in other memory cells due to the mechanisms that protect against the too frequent write to the same cell.

Quorum

AIX allows you to use a mechanism that does not allow the activation of a volume group where 50% or more of the physical volumes are unavailable. The quorum parameter controls this mechanism. This parameter is set automatically, and it determines the number of descriptors (VGDA - Volume Group Descriptor Area) that must be available so that the group can be activated. Each disk in the group has a VGDA. It is important to note that the first disk in the volume group has two VGDAs, while the rest has only one.

If this parameter is set to 1 (*lsvg VGname*), then the quorum is disabled. Otherwise, the number assigned to it is an integer corresponding to the number of VGDAs that must be available in order to achieve quorum. The values of the enabled quorum for different numbers of disks are shown in Table 7-4:

Table 7-4: The value of the quorum parameter for different disk counts.

Number of disks in VG	Number of VGDAs in VG	Quorum parameter value in VG
1	2	2
2	3	2
3	4	2
4	5	3
…	…	…

If half or more of the disks in the VG are damaged, then the data stored there may be inconsistent or inaccessible. Unknowingly attempting to use the disks may end up corrupting the data that have been retained. Therefore, the failure of the volume group is a signal to the administrator that there is a large problem with the disk, and that this problem should be resolved prior to resuming operation. The use of this mechanism is recommended for groups that consist of many disks.

You should not use this mechanism against the *rootvg*, which has only two disks. The quorum in the two-disk volume group is 2. This means that if one of the disks fails, the group may not be activated. This will occur in the event of the failure of the first disk that has two VGDAs. If the *rootvg* group cannot be activated, the operating system will not be able to boot even if the loss of one disk is not a problem for the mirror. If a disk from that group was damaged during system operation, it would not

have any negative effects on the system, although after the restart, the system would not start. AIX disables quorum by default in volume groups that contain two disks.

Adding a volume group

When you add a volume group, you should have approximately 2 MB of free space in the root directory (/). This is necessary because when adding a group, the operating system also creates structures that represent it in the file system. You can use the SMIT menu (*smit mkvg*) to create a group. Once selected, you can choose the type of volume group to create. Three types of groups are available:

- *Original*;
- *Big*; and
- *Scalable*.

These types were described at the beginning of this chapter. At this point, it must be recalled that groups other than the Scalable type exist for reasons of compatibility with earlier versions of the operating system. When you select a group type, the menu shown in Figure 7-6 will be displayed. It is similar for each type. The menu below is for creating a Scalable volume group.

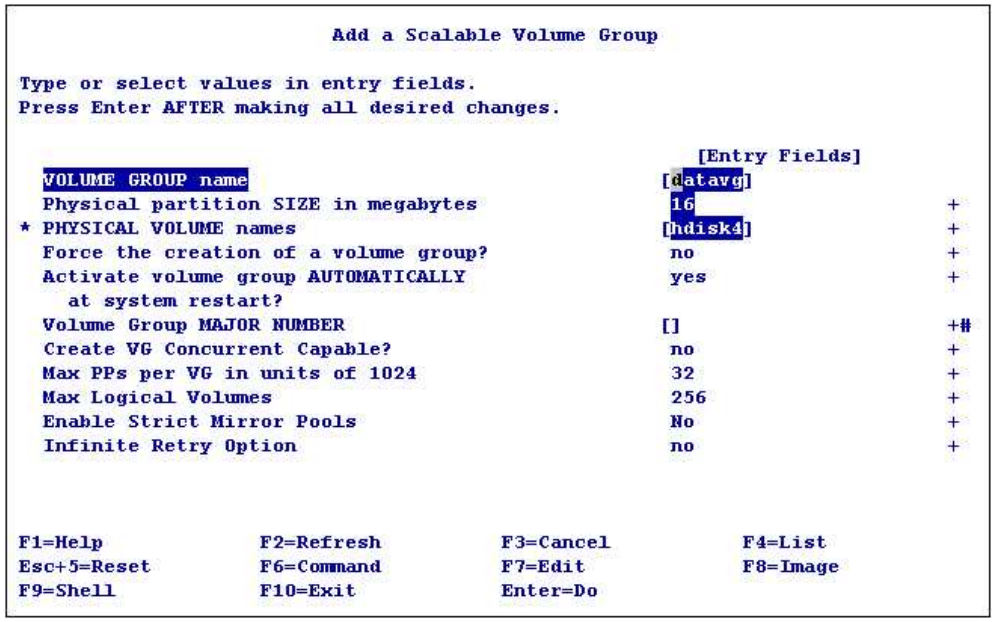

Figure 7-6: smit mkvg - Create a volume group (mkvg).

Key parameters:

• **Physical partition SIZE in megabytes** - Determines the size of the physical partition for the entire volume group. The value you specify here will determine the size of the disk space that will be the unit of allocation for both the logical volume and file system. In other words,

every logical volume and file system in this volume group will be made up of pieces of this size.

When defining a Scalable group, there are no real limitations to worry about. This is the case for the older types of volume groups. When defining this parameter for an original or big volume group, remember that the maximum number of physical partitions per disk is 1016. Due to this limitation, setting a physical partition size to too low value will result in the impossibility of adding a disk greater than 1016* (the size of the physical partition).

Let's look at this example in more detail. We have created an original volume group with a physical partition size of 128 MB. The previously used disks have a size of 72 GB. If you try to add a 146 GB physical volume to this volume group, the action will fail. For this task to succeed, the disk would have to be divided into more than 1016 physical partitions, which is impossible due to the above-mentioned limitation.

This limitation (1016 physical partitions per physical volume) can be partially overcome by creating a volume group from the command line (*mkvg*). In this case, you can use the *-t* parameter to specify a higher number of PPs per disk. When you increase the number of physical partitions on a physical volume, you limit the number of physical volumes in the volume group. The relevant relationship is shown in Table 7-5.

Table 7-5: Number of PPs per PV vs number of PVs per VG.

Number of PPs per PV	Original VG Number of PVs per VG	Big VG Number of PVs per VG	-t parameter when creating VG
1016	32	128	
2032	16	64	-t 2
3048	10	42	-t 3
4064	8	32	-t 4
5080	6	25	-t 5
…	…	…	…

The *-t* parameter also exists in the *chvg* command, which changes the properties of the volume group. This command can be used during normal system operation.

- **PHYSICAL VOLUME names** - Specifies the disks (PVs) that will be part of the volume group. You can add a disk to the volume group using the command *extendvg VGname hdiskX*. A disk can only belong to one volume group.
- **Activate volume group AUTOMATICALLY at system restart** - Determines whether or not the volume group is to be activated during the operating system startup (*varyonvg* command).
- **Volume group MAJOR NUMBER** - Specifies the volume group identifier. When the operating system refers to any device, it uses two numbers, namely the *major number* and the *minor number*. As AIX treats volume groups as devices, it only refers to them by these numbers. Users refer to a VG by its name, so this parameter does not matter to them. It is best to leave it unchanged. In clustered solutions where a volume group can be mounted on one server and then on the other, it is important to maintain the consistency of the numbering.

- **Create VG concurrent capable** - This option is mainly used in clustered solutions. Setting the volume group in *enhanced concurrent* mode allows multiple servers to use this volume group simultaneously. With this setting, the volume group works in the *active* mode on one of the servers and allows you to perform all operations, while on the others it works in *passive* mode.

Another way to create a volume group is to use the *mkvg* command. The majority of parameters described here can be changed using the *chvg* command after creating a volume group, although you cannot change the size of the physical partition.

Changing parameters

Changes to individual parameters can be made using the SMIT menu (*smit chvg*). This menu makes use of the *chvg* command.

You can convert a volume group into its newer form (options: *Change to big VG format?* and *Change to scalable VG format?*). The conversion of the original group to the big group is a simple operation, and it can be carried out on the mounted (*varyonvg* command) volume group without any negative influence on the contents (logical volumes, and file systems). Conversion to a scalable group is more complex. To do so, you must first disable (*varyoffvg*) the volume group, which involves unmounting the file systems that belong to it.

The conversion process in both cases does not represent a big change for the system. Therefore, the whole operation takes only a few seconds and carries little risk. It is important to note that it is not possible to convert the *rootvg* group.

The changes that you can perform using the SMIT menu are shown in Figure 7-7.

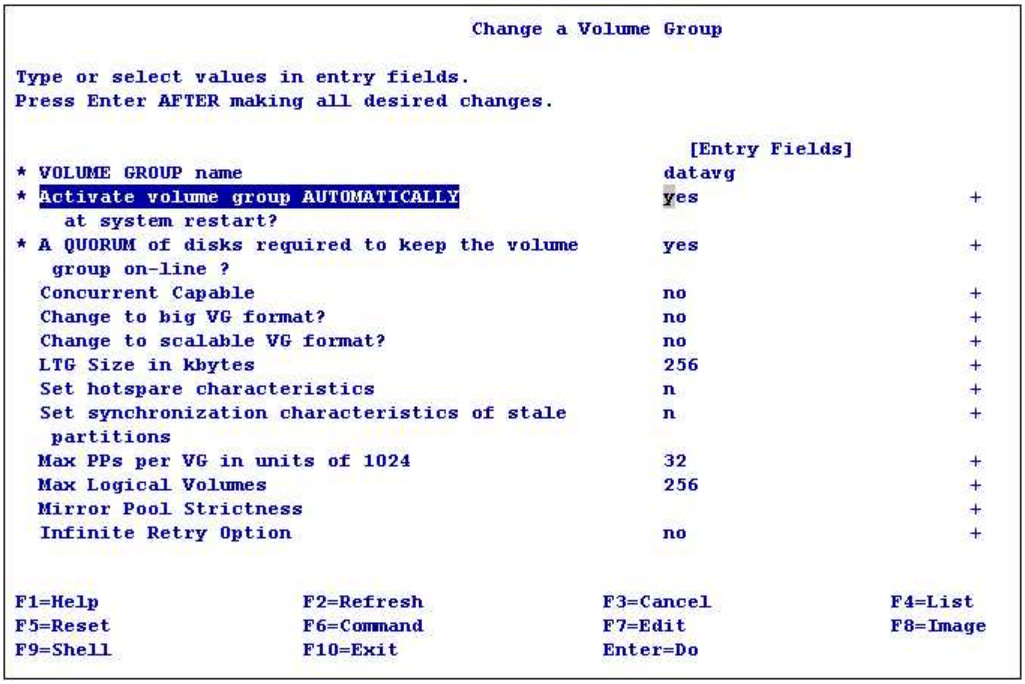

Figure 7-7: Change the parameters of the volume group.

Basic operations

You can perform basic operations on a volume group using the SMIT menu or via the command line. The latter method is described below. Good practice requires you to know how to modify the system using the command line, without the need for more complex tools. This knowledge allows you to automate the system management with scripts. Properly written scripts in turn allow you to perform a series of actions simultaneously, thereby saving your time.

The commands shown below allow you to apply many parameters in order to adjust their behavior to your specific needs. Their basic application is hence detailed below.

Activation of a volume group

```
# varyonvg VGname    # Activation of the group „VGname”.
```

If you want to perform any operations on file systems or logical volumes that belong to a given volume group, it must be active. The group activation will fail if the quorum rule is not met or if there is a discrepancy between the VG characteristics stored in the *Object Data Manager* (ODM) files and the characteristics stored in the *Volume Group Descriptor Area* (VGDA).

The *rootvg* is automatically activated during startup. Other groups can be activated automatically or not. You can verify this by checking the *AUTO ON* setting in the output of the *lsvg VGname* command.

Deactivation of a volume group

```
# varyoffvg VGname    # Deactivation of the group „VGname”.
```

The deactivation of each group takes place automatically when the system is shutting down. The file systems and logical volumes that belong to a volume group are not available when that group is inactive. Deactivation can be performed “manually” at any time, although it will fail if any file system that belongs to the group is mounted or any logical volume is open.

Adding a physical volume

```
# extendvg VGname hdiskX    # Add the physical volume hdiskX to "VGname" group.
```

If there is no space on the file system and there is no free space in the VG, then it is necessary to add new physical volumes to the group. This can be done using the above command without affecting the current operation of the system. The command will fail if the PV that is added already belongs to a volume group or the group to which the disk is added is inactive.

Removing a physical volume

```
# reducevg VGname hdiskX    # Remove the physical volume hdiskX from „VGname” group.
```

The volume group must be active at the time you attempt to remove the disk from it. When you remove all the disks from the group, it no longer exists. You can use the *-d* parameter, which will automatically remove all the LVs that are on the disk. However, it is better not to use this parameter, but to instead remove the logical volumes separately. This renders it less likely that an error will occur.

Other parameters of the volume group

By using the *chvg* (Change Volume Group) command, you can change several other parameters. One such parameter is the automatic activation of the VG during the operating system startup:

```
# chvg -a y VGname
```

Moving a volume group between systems

There is a simple way to move a VG with all its logical content , that is, with the LV and file systems, from one AIX to another. The transfer can be done with disks that belong to a volume group, and it can be done in the following way.

Source system

Before moving a volume group, you must deactivate it:

```
# varyoffvg VGname
```

Then, remove the volume group's information from the source system:

```
# exportvg VGname
```

More precisely, this removes the volume group's information from the ODM database and configuration files, which renders the group invisible in the system. Any data housed on disks are not lost during this operation. The data that describe the volume group are not lost, since such data are still stored in the VGDA located on each of the disks that belong to the group.

The next step is to remove the disks that belong to a group from the system:

```
# rmdev -dl -X
```

-l - Changes the status of the disk from *available* to *defined*.
-d - Removes the device (disk) from the ODM database (the disk is no longer visible in the system).

After performing this operation, the disks that belong to the moved volume group can be moved to the target system.

Target system

In order to make the freshly added disks visible on the system, run the *cfgmgr* configuration program. The next step is to import the volume group to the target system:

```
# importVG -y groupVG hdiskX # Save VG information in the ODM database.
```

The volume group description is read from *hdiskX* and saved in the system configuration. The name of the volume group must be unique within the system, so you can name it after the *-y* parameter. You can skip this parameter and allow the system to assign its own unique name (which will of course be less user friendly).

If there are no conflicts with existing file systems, the import process creates mount points and entries in the */etc/filesystems* file that describe the imported file systems. Where the logical volume names in an imported group overlap with the existing names, they are changed to unique names. Following the import, the group is in an inactive (*varyoff*) state. You can activate it manually using the *varyonvg VGname* command. You can also set it to automatic activation during system startup using the *chvg -a y VGname* command.

Logical Volumes (LV)

A logical volume is a set of equally sized logical partitions (LPs). A logical partition is similar to a physical partition. You could say that this is a pointer to one, two (mirror), or three (double mirror) physical partitions. Thus, writing data to a logical partition that points to two physical partitions results in the same data being written in two places. As a result, RAID 0, 1, and 10 can be created at the logical volume level rather than at the physical volume level (disks). An example of logical volumes is given in Figure 7-8.

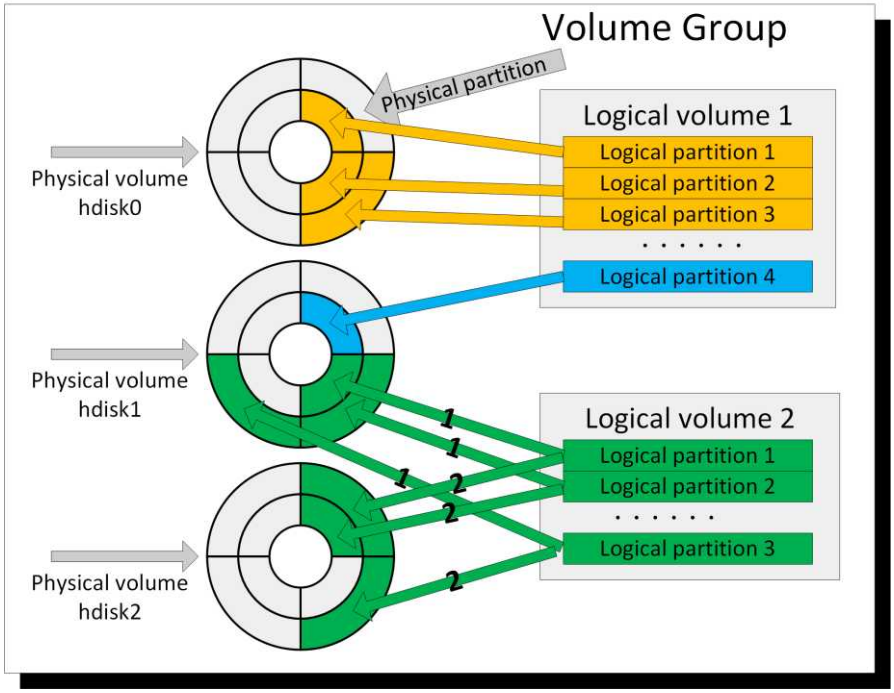

Figure 7-8: Logical volumes' structure.

As you can see in the above figure, a logical volume can be built from any physical partition that resides on any physical volume that belongs to the same volume group. For logical volume operations, it does not matter whether these physical partitions are next to each other or if they are scattered all over the disk. From the perspective of the operations performed on it, they represent a continuous area. The *logical volume 2* shown in the figure above demonstrates how the mirror works. As you can see, one logical partition points to two physical ones, which results in two data writings to two physical partitions simultaneously.

When resizing a logical volume, logical partitions are added or removed. This means that the smallest size by which you can resize a logical volume is the size of the physical partition.

A logical volume can be compared to a traditional partition. As with a partition, you can create file systems on a logical volume or work directly with it.

The logical volumes are represented within the system by two files in the */dev* directory. The first has the same name as the logical volume, while the second is preceded by the letter *r*. According to this rule, the logical volume containing the root file system (/), that is, *hd4*, is represented by:

- */dev/hd4* - A block device, cached by the operating system.
- */dev/rhd4* - A character device, not cached by the operating system.

Block and character devices

In all UNIX systems, there is a division between the block devices and the character devices. The main difference between the two is the way the system operates on them. Some devices may exist as both a block and a character. In that case, the way in which the system will operate on the device is determined by the device type it uses (block device or character device). Typical devices that exist in both types are logical volumes and physical volumes.

Block devices

Block devices, including logical volumes, can be identified by the letter *b* that defines the file type.

```
# ls -l /dev/hd4
brw-rw----   1 root     system       10,  4 Aug 23 07:35 /dev/hd4
```

Such devices are buffered by the operating system, which often renders operations on them faster than in the case of character devices. Buffering means that when the data are written, they first go to the memory area responsible for buffering the writings and readings from the device. Writing from memory to disk is only performed periodically or after a certain amount of data have been collected.

In the case of logical and physical volumes, the delayed write operation allows the operating system to write records in a different order than that in which they appeared in the buffer. This allows the system to optimize the writes, for example, by combining them into larger packages. The main benefit of this behavior is the improved performance of the operations performed on these devices, but this does not occur in all cases.

Character devices

Character devices are identified by the letter *c* that defines the file type, as well as by having a name starting with the letter *r* (*raw device*).

```
# ls -l /dev/rhd4
crw-rw----   1 root     system       10,  4 Aug 23 07:35 /dev/rhd4
```

Character devices are not buffered by the operating system, which means that any write to the device is done immediately. The data that are written to the character devices always go to them in the same order in which they were issued.

Character devices are often used by databases, which improves the performance of I/O operations. Why should this improve performance if buffered access to the device has been determined to be faster? The answer lies in the fact that databases have their own caching system. It is tailored to the way they work, so it is more efficient than buffering by the operating system. Additionally, when the database system is running on a buffered device, double buffering takes place. The same data cached by the database go to the operating system buffer prior to the write to disk. This generates additional memory operations and hence the inefficient use of memory to store the same data in two places.

The parameters that describe the logical volume (lslv)

You can use the *lslv* command to display the basic parameters describing a logical volume:

```
# lslv hd3   # Basic characteristics of LV.
LOGICAL VOLUME:     hd3                    VOLUME GROUP:   rootvg
LV IDENTIFIER:      00f6217700004c00000001574d9300a7.7 PERMISSION:     read/write
VG STATE:           active/complete        LV STATE:       opened/syncd
TYPE:               jfs2                   WRITE VERIFY:   off
MAX LPs:            512                    PP SIZE:        16 megabyte(s)
COPIES:             1                      SCHED POLICY:   parallel
LPs:                38                     PPs:            38
STALE PPs:          0                      BB POLICY:      relocatable
INTER-POLICY:       minimum                RELOCATABLE:    yes
INTRA-POLICY:       center                 UPPER BOUND:    32
MOUNT POINT:        /tmp                   LABEL:          /tmp
MIRROR WRITE CONSISTENCY: on/ACTIVE
EACH LP COPY ON A SEPARATE PV ?: yes
Serialize IO ?:     NO
INFINITE RETRY:     no                     PREFERRED READ: 0
```

The parameters *TYPE*, *MOUNT POINT*, *LV STATE*, and *PP SIZE* were described when discussing volume groups, and they have the same meaning here. There are a number of other interesting parameters:

- ***MAX LPs*** - The maximum number of logical partitions that a logical volume can have. This is the value assigned to the logical volume at creation. You can change this value to be within the limits of the volume group type applied.
- ***COPIES*** - The number of physical partitions per logical partition. Value 2 means mirror, while value 3 means double mirror.
- ***LPs*** - The number of logical partitions that make up a logical volume. With this parameter, you can check the size of a logical volume by multiplying the *LPs* by the *PP SIZE*.
- ***STALE PPs*** - The number of damaged physical partitions in a logical volume.
- ***INTER-POLICY*** - The policy of allocating physical partitions on disks. As you know, the physical partitions of one logical volume can be found on multiple disks. This parameter determines whether the *Logical Volume Manager* will allocate all the PPs to the minimum number of disks, or else spread them to the maximum number of disks when creating or expanding a logical volume. The *INTER-POLICY* can be set in two ways:
  - ***minimum*** - The system tries to put all the PPs on one physical volume. If they do not fit, then it puts them on the minimum number feasible.
  - ***maximum*** - The system tries to deploy all the PPs on as many disks as it can within the volume group. This arrangement increases the performance of disk operations, since the operations performed on one logical volume are spread across multiple disks.
- ***INTRA-POLICY*** - Specifies the disk regions in which to allocate physical partitions for a given logical volume. Nowadays, this is becoming less and less important, since most disk resources are made available to the system from external disk storage. This parameter is only relevant when you work with spinning disks that are connected directly to the server. In that case, this setting may have a significant effect: the closer to the center of the disk, the shorter the data access time, and the closer the external edge, the faster the sequential read and sequential write. Therefore, the most used LVs should be placed in the center of the disk.

However, the LVs on which the writes and reads are performed using large blocks should be placed on the outer edge. The internal edge is the least efficient, so it is best to store data that is rarely used there.

Physical volumes are divided into five areas:

- ***Edge***.
- ***Middle***.
- ***Center***.
- ***Inner Middle***.
- ***Inner Edge***.

The above physical volume division is depicted in Figure 7-9.

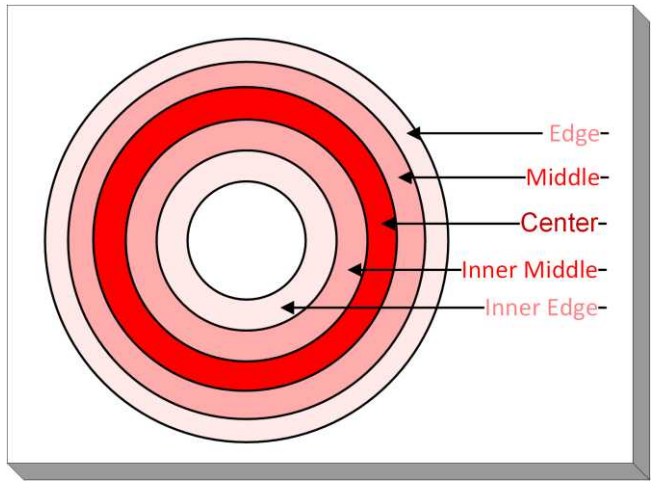

Figure 7-9: Disk areas - INTRA-POLICY

- ***MIRROR WRITE CONSISTENCY*** - A mechanism for speeding up system recovery.

  Consider the case where this mechanism is disabled, and a system failure has occurred, which results in the mirrored LV being not properly closed. In such a situation, before it can be used, the operating system will have to synchronize its copy (*syncvg -f -l LV name*). This process is performed during the system startup and it may take a few seconds or minutes. With the mechanism turned on, the system makes additional writes. It writes information about the logical volume synchronization. With this additional information, synchronization is faster, and it can be performed in the background in a transparent manner. The main disadvantage of this mechanism are the extra disk writes. They can reduce the performance of I/O operations.
- ***EACH LP COPY ON A SEPARATE PV?*** - Causes each copy of the logical partition to be created on a separate physical volume. During the normal operation of the system, the value of this parameter should always be set to *YES*; otherwise, when creating a mirror, you can get both copies on one disk. Such a mirror would slow down the writing, and it would not give you the benefit of greater availability.
- ***SERIALIZE IO*** - Merges several records into one. If the system receives several write orders for subsequent disk blocks, it will merge them into one larger write task, thereby improving the

performance of I/O operations. Despite the apparent benefit of using this mechanism, it is generally advisable to disable it (this is the default setting) because the file systems and databases have their own mechanisms and it would be unfavorable to duplicate them.

- ***WRITE WERIFY*** - Lets you quickly detect abnormalities in the stored data and maintain their consistency. With the mechanism turned on, after each write to the disk the same data are read to verify the correctness of the record. This mechanism is extremely rarely used, since the only reason to use it is to test the correctness of internal disks.
- ***SHED POLICY*** - Specifies how mirrored logical volumes work. It allows you to manipulate how they are read and written, making them more efficient, or increasing their reliability. There are four modes:
  - ***Sequential*** - All reads are performed from the primary disk (from the first copy). Writes are made on the first copy, then in the next step on the second copy, and finally on the third. After the last copy has been written, the operation is considered complete. This is a mode that increases the reliability of the disk subsystem, although it greatly limits its performance.
  - ***Parallel*** - The reads are distributed between all the disks in the mirror (all the copies of the logical volumes). The read operation is performed by a less loaded disk, which is the one that has fewer tasks waiting in the queue. At the same load, the primary disk (with the first copy of the data) is selected. Writing is performed simultaneously to both disks, and it is considered complete when both tasks are completed. This is the default setting that gives you the highest possible mirrored LV performance. This solution can make the differential performance of each disk in the mirror less onerous by directing more read operations to the less loaded disk.
  - ***parallel write/sequential read*** - A mixed mode. The read is always performed from the primary disk in the same way as in *sequential* mode, while the write is performed simultaneously in the same way as in *parallel* mode.
  - ***parallel write/round robin read*** - This is very similar to *parallel* mode, although the readings are evenly distributed. Therefore, it is a good configuration for logical volumes created on disks with identical performance.
- ***BB POLICY*** - Determines whether the damaged blocks are to be relocated or not.
- ***RELOCATABLE*** - Determines whether the given LV is subject to a reorganization process. The reorganization will be discussed later in this chapter.
- ***UPPER BOUND*** - Specifies the maximum number of physical volumes that can be used to create a mirror.

Logical volumes placement (lslv -l)

You can obtain information about the logical volume placement on disks using the command *lslv -l LV name*. This is illustrated by the following example:

```
# lslv -l hd3
hd3:/tmp
PV               COPIES        IN BAND       DISTRIBUTION
hdisk1           006:000:000   100%          000:000:006:000:000
hdisk0           006:000:000   100%          000:000:006:000:000
```

Key parameters:

- ***PV*** - The physical volumes on which the logical volume is located. In the above example, the LV is on two physical volumes.
- ***COPIES*** - Three fields separated by colons tell us about (starting from left):
  - The number of logical partitions that have one physical partition on a given physical volume. This is the most common possibility.
  - The number of logical partitions that have two physical partitions on a given physical volume. This is usually the case with an incorrectly built mirror, since having two copies on one disk does not prevent any failure, but rather doubles the amount of time it takes to write data.
  - The number of logical partitions that have three physical partitions on a given physical volume. The same comments that were made in relation to the previous case apply to this setting, except that the amount of time required to write data is tripled.
- ***DISTRIBUTION*** - Shows where the physical partitions of a given LV are located. Starting from the left, you can see the number of PPs on the edge, middle, center, inner middle, and inner edge. In the above case, all the PPs (six) are on the center.
- ***IN BAND*** - Determines what percentage of deployment recommendations have been reached. The recommendations are assigned to a logical volume when it is created. They can be changed throughout its existence. The recommendation for the logical volume *hd3* was to place all the physical partitions in the center of the physical volume. This was 100% completed. This means that all the PPs of this LV have been placed where they should, that is, in the center area of the disk.

Detailed logical volumes placement (lslv -l)

You can obtain detailed information about the logical volume placement on disks using the command *lslv -m LV name*. This is illustrated in the following example:

```
# lslv -m hd3
hd3:/tmp
LP    PP1  PV1               PP2  PV2               PP3  PV3
0001  0219 hdisk1            0296 hdisk0
0002  0220 hdisk1            0297 hdisk0
0003  0300 hdisk1            0300 hdisk0
0004  0301 hdisk1            0301 hdisk0
0005  0302 hdisk1            0302 hdisk0
0006  0303 hdisk1            0303 hdisk0
```

In this way, you can obtain detailed information about the physical partitions of a logical volume. In the above example, you can see that logical partition number 1 of the logical volume *hd3* has two copies. The primary on *hdisk1* is on physical partition 0219, while the secondary on *hdisk0* is on physical partition 0296.

Adding a logical volume

You can add a logical volume using the SMIT utility or the *mklv* command. When creating an LV from the SMIT, you assign values to individual parameters. The names of these parameters can be misleading, since they can be different to those obtained using the *lslv* command. The parameters in the

SMIT menu and their relation to the parameters obtained using the *lslv* command are presented in Figure 7-10.

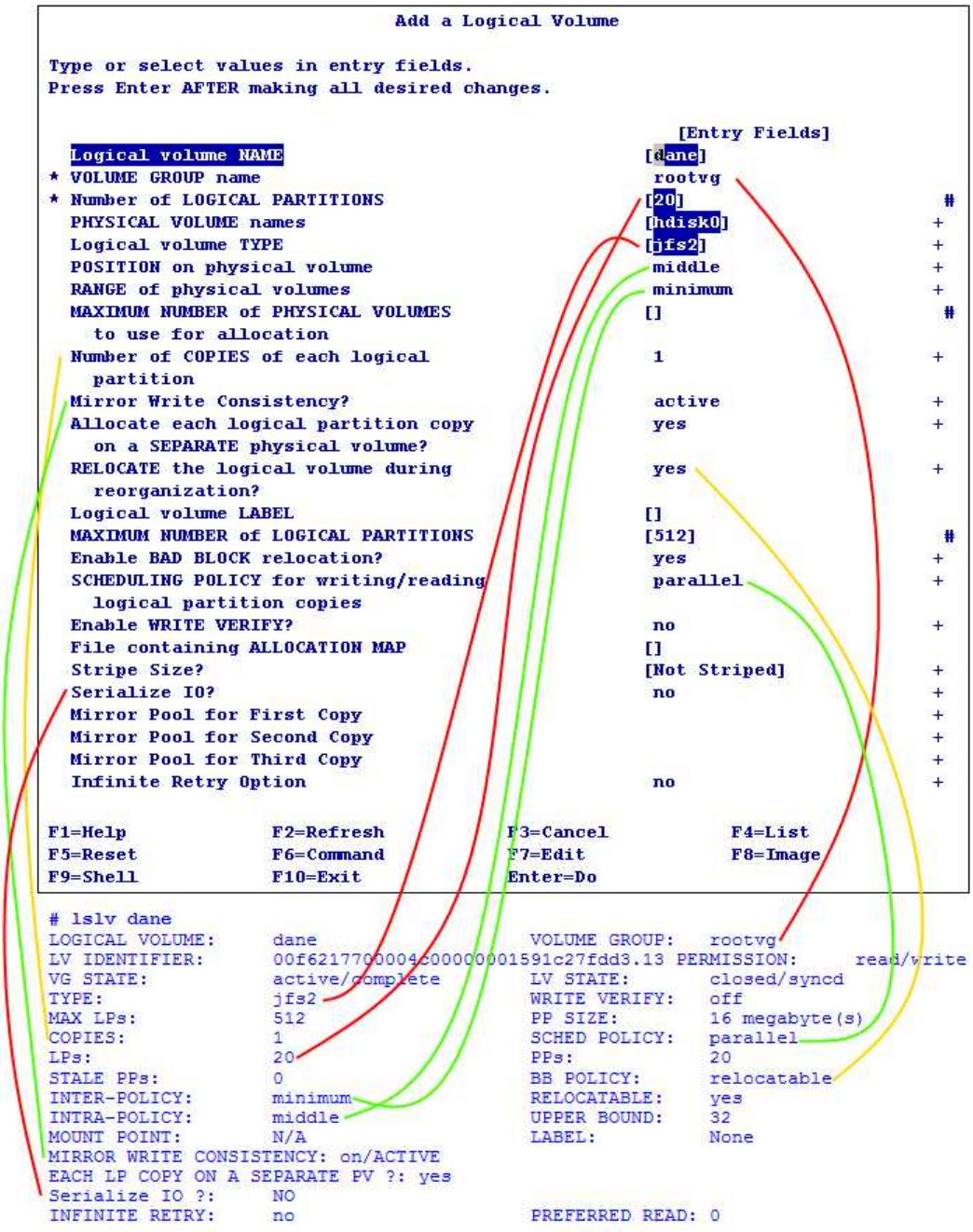

Figure 7-10: smit mklv - Mapping to the lslv output.

The majority of parameters that can be set when adding a logical volume are described at the beginning of this chapter. Below is a description of those parameters that have not been previously discussed or an additional comment about parameters that are already known:

- ***Number of LOGICAL PARTITIONS*** (*LPs* in the *lslv* command) - The size of a logical volume. You can convert it to megabytes by multiplying the number of logical partitions by the size of the physical partition (*PP SIZE*).
- ***PHYSICAL VOLUME names*** - Specifies the physical volumes on which to create a logical volume. If several physical volumes are specified, the best way to allocate logical partitions to them will depend on the setting of the *RANGE of physical volumes*.
- ***POSITION on physical volume*** (*INTRA-POLICY* in the *lslv* command) - Specifies where the physical partitions of the logical volume should be placed. This parameter is only relevant when using physical spinning disk drives connected directly to the server. In such a case, it directly affects the performance of I/O operations.
- ***RANGE of physical volumes*** (*INTER-POLICY* in the *lslv* command) - This can take two values:
    - ***minimum*** - The operating system will try to allocate physical partitions on the minimum number of disks. This means that they will be placed:
        - On one disk in the case of creating an LV in one copy (*Number of COPIES of each logical partition = 1*).
        - On two disks in the case of creating a mirrored LV (*Number of COPIES of each logical partition = 2*)
        - On three disks in the case of creating an LV in three copies (*Number of COPIES of each logical partition = 3*)
    - ***maximum*** - The system will try to distribute physical partitions on all the PVs on which the logical volume is created (on all the *PHYSICAL VOLUME names* listed). This setting allows you to achieve higher I/O performance.
- ***Number of COPIES of each logical partition*** (*COPIES* in the *lslv* command) - Determines whether to create a normal logical volume (value 1), a mirrored logical volume (value 2), or a double mirrored logical volume (value 3).
- ***Allocate each logical volume on a SEPARATE physical volume?*** - This parameter can take three values:
    - ***Yes*** - Specifies that it will not be possible to have two logical copies of the partition on one physical volume. This is set by default. Typically, this setting is correct, since it prevents the mirror from being created on a single disk, which usually make no sense.
    - ***No*** - The opposite of the above setting.
    - ***Superstrict*** - This means that the logical partitions of a single mirror cannot be located on a physical volume that already has the partitions of another mirror.
- ***RELOCATE the logical volume during reorganization?*** (*RELOCATABLE* in the *lslv* command) - Determines whether the operating system should relocate the logical partitions of the logical volume according to the allocation policy (*INTRA-POLICY*) during the relocation process (*reorgvg* command).
- ***Enable BAD BLOCK relocation?*** (*BB POLICY* in the *lslv* command) - This parameter determines whether the system is to replace the damaged disk block with another one. Such operations are invisible to the user, since the only trace is in the error log.
- ***File containing ALLOCATION MAP*** - Here, you can specify the path to the file that specifies on which physical volumes and on which physical partitions to create a logical volume. The file should have entries for each logical partition of the logical volume. The entries should look like this: *PVname: PPnum1 [-PPnum2]*. An example of a file that defines a logical volume consisting of five physical partitions follows:

```
hdisk1:166
hdisk1:168
hdisk1:170-172  # This line means the same as: hdisk1:170 hdisk1:171 hdisk1:172.
```

Stripe Size? - This parameter is used to create striping (i.e., RAID 0 or RAID 10). To create such a configuration for RAID 0, create a logical volume on at least two physical volumes, while for RAID 10 you should create it on four. You can set values from 4 KB to 128 MB. The speed of the striped logical volume depends on what value is selected. Large values are good for LVs, where the reads and writes are performed with large blocks of data. The smaller the blocks that will be used with disk operations, the smaller the value that should be set. Usually, a better way is to use striping without the *Stripe Size* option. This method involves creating a volume group with a small physical partition as well as creating logical volumes with the option *RANGE of physical volumes = maximum*. This will create a stripe with a stripe size equal to the size of the physical partition. Since we normally operate on LUNs from external storage, such a stripe can be large without compromising performance. Using this approach, we can get rid of all the restrictions that the *Stripe Size* can impose on us.

Another way to add a logical volume is to use the command line and the *mklv* command:

```
# mklv -y'dane' -t'jfs2' -c'2' rootvg 40 hdisk0 hdisk1
```

Parameters used:

- ***-y*** - The name of the logical volume to be created.
- ***-t*** - The logical volume type.
- ***-c*** - The number of logical volume copies.
- ***rootvg*** - The volume group in which we create a logical volume.
- ***40*** - The number of logical partitions (size of the created LV).
- ***hdisk0 hdisk1*** - The physical volumes on which the logical volume is to be created.

Most parameters have default values, so you only need to specify a few of them when executing a command. The required parameters are:

- The name of the logical volume;
- The volume group in which to create it; and
- The number of logical partitions.

Adding a mirrored logical volume

A logical volume can be created as a mirror. If you add a logical volume using the SMIT menu, you need to set three parameters. These parameters are shown in Figure 7-11.

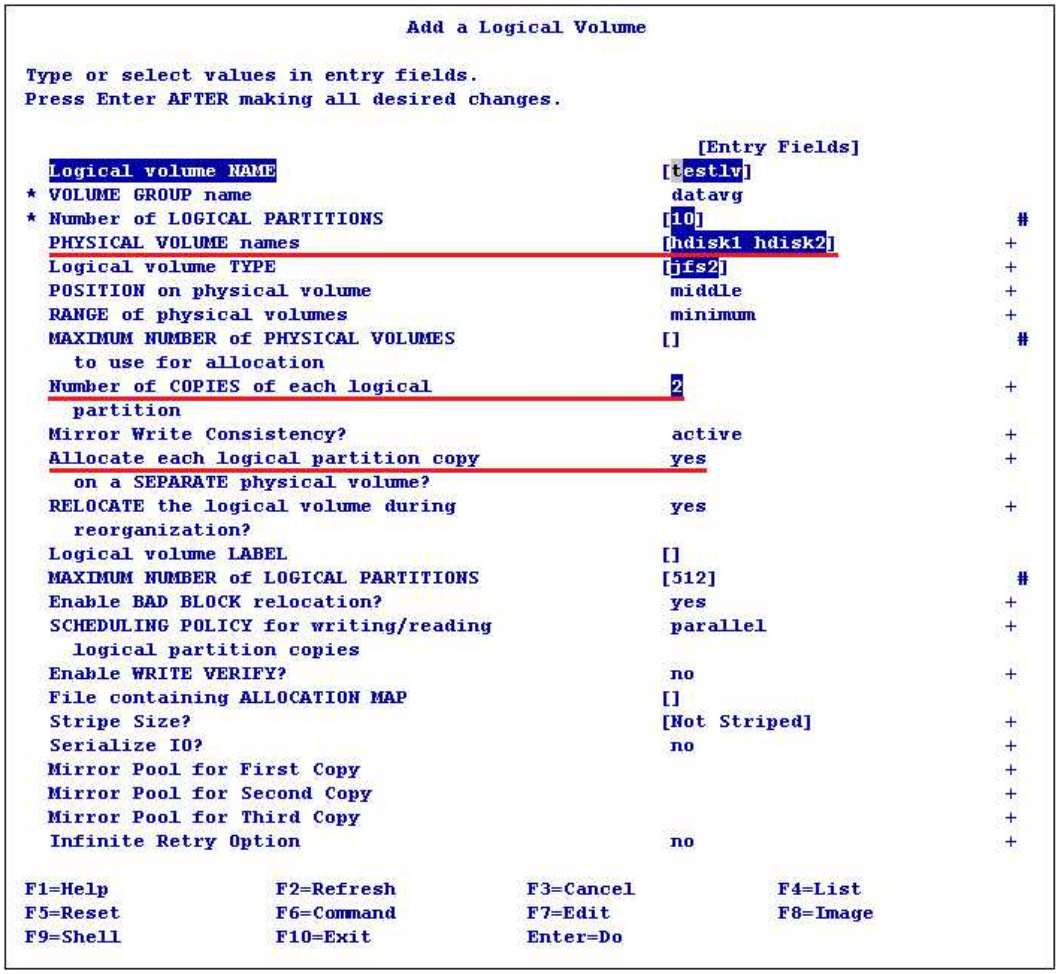

Figure 7-11: Adding a mirrored logical volume.

- ***PHYSICAL VOLUME names*** - This should point to two disks.
- ***Number of COPIES of each logical partition*** - This should be 2.
- ***Allocate each logical partition copy on a SEPARATE physical volume?*** - This should be set to yes.

Once you have created a logical volume, you can verify that you created it in the correct way.

```
# lslv -m testlv
testlv:N/A
LP    PP1  PV1               PP2  PV2               PP3  PV3
0001  0204 hdisk1            0204 hdisk2
0002  0205 hdisk1            0205 hdisk2
0003  0206 hdisk1            0206 hdisk2
0004  0207 hdisk1            0207 hdisk2
0005  0208 hdisk1            0208 hdisk2
0006  0209 hdisk1            0209 hdisk2
0007  0210 hdisk1            0210 hdisk2
0008  0211 hdisk1            0211 hdisk2
0009  0212 hdisk1            0212 hdisk2
0010  0213 hdisk1            0213 hdisk2
```

As you can see, each logical partition has two physical partitions, so everything went according to plan.

Splitting the mirror into independent logical volumes

In AIX, it is possible to split mirrors into independent logical volumes. You can do this using the *splitlvcopy* command. The logical volumes created in this way are independent of each other, and they should be treated as separate entities. You can perform a splitting operation on an active logical volume, but this is not recommended. Potentially, it can lead to data inconsistencies, so it is best to unmount the file system created on the logical volume. This will close the logical volume. If you used direct access to the logical volume, you should close the application that is using it.

The following example demonstrates how to split copies of the *testlv* logical volume that was created in the previous section:

```
# splitlvcopy -y testlv_new testlv 1 hdisk2
testlv_new
```

The above command causes a copy of the *testlv* logical volume that resides on the physical volume *hdisk3* to be split, and it creates a separate LV called the *testlv_new*. The number 1 used in the command specifies that *testlv* should remain in one copy. If you extract one copy from a logical volume that is made up of three copies, then the number would be 2.

The situation after the logical volume separation is presented in the following output and in Table 7-6.

```
# lsvg -l datavg
datavg:
LV NAME             TYPE       LPs     PPs     PVs  LV STATE      MOUNT POINT
testlv              jfs2       10      10      1    closed/syncd  N/A
testlv_new          jfs2       10      10      1    closed/syncd  N/A
```

Table 7-6: A logical volume separated by the splitlvcopy command

Original logical volume	Extracted logical volume

`# lslv -m testlv`
`testlv:N/A`
`LP  PP1  PV1  PP2  PV2  PP3  PV3`
`0001  0204 hdisk1`
`0002  0205 hdisk1`
`0003  0206 hdisk1`
`0004  0207 hdisk1`
`0005  0208 hdisk1`
`0006  0209 hdisk1`
`0007  0210 hdisk1`
`0008  0211 hdisk1`
`0009  0212 hdisk1`
`0010  0213 hdisk1`
	`# lslv -m testlv_new`
	`testlv_new:N/A`
	`LP  PP1  PV1  PP2  PV2  PP3  PV3`
	`0001  0204 hdisk2`
	`0002  0205 hdisk2`
	`0003  0206 hdisk2`
	`0004  0207 hdisk2`
	`0005  0208 hdisk2`
	`0006  0209 hdisk2`
	`0007  0210 hdisk2`
	`0008  0211 hdisk2`
	`0009  0212 hdisk2`
	`0010  0213 hdisk2`

After the operation, there are two logical volumes in place of one *testlv*:

- ***testlv*** - The original, which is located entirely on the physical volume *hdisk1*.

- ***testlv_new*** - The new logical volume, which is separated from the *testlv* copy that was on the *hdisk2* physical volume.

Changing the parameters of a logical volume

The changes that you can make to logical volumes are presented in Figure 7-12.

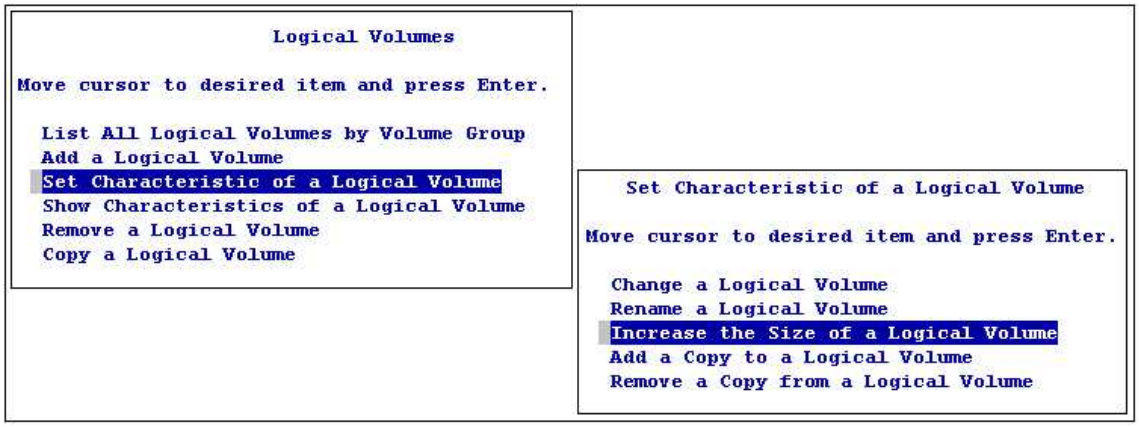

Figure 7-12: Changing the parameters of a logical volume.

You can make the following changes:

1. **Change a Logical Volume** (*chlv* command) - Some of the parameters specified when creating a logical volume can be changed "on the fly," for example, *INTRA-POLICY*, *SHED POLICY*, *BB POLICY*, and so on.
2. **Rename a Logical Volume** (*chlv -n 'new_name' old_name* command).
3. **Increase the Size of a Logical Volume** (*extendlv* command) - The size is increased by the specified number of physical partitions. This is done in a way that is transparent to users. An example of the command:

   ```
   # extendlv examplelv 10  # Increases LV size by 10 PP.
   ```

   The physical partitions added to the logical volume will be in the disk space that complies with the allocation policy (*INTRA-POLICY*). If there is no place available in these areas, the partitions will be in the areas closest to the allocation policy.
4. **Add a Copy to a Logical Volume** (*mklvcopy* command) - In other words, create a mirror (or double mirror) of the LV. As mentioned earlier, mirrors in AIX are created at the logical volume level rather than at the entire disk level (PV).
5. **Remove a Copy from a Logical Volume** (*rmlvcopy* command) - In other words, remove a mirror (or double mirror) from a given LV, for example:

   ```
   # rmlvcopy hd3 1 hdisk0 # Remove all LV copies from hdisk0 so that LV has only 1 copy.
   ```

Moving logical volumes within a volume group

A very useful LVM feature is the ability to move logical volumes and thus file systems between physical volumes within a volume group. This allows you to move data you want to delete from the disk, or to move data to a faster disk in order to improve performance.

Consider a situation where you add new disks to the system to improve the I/O performance. Simply adding the disks will not improve the performance of your system, you will need to allocate the data accordingly. In a traditional disk system, there would be a serious problem in relation to moving data between the disks, which would require stopping applications that use the relevant files. On AIX, this can be done without any interruption to the application, with the use of the *migratepv* command.

Moving data between disks is done at the physical partition level. The following example shows the transfer of all the physical partitions from *hdisk1* to *hdisk2* (Figure 7-13).

```
# migratepv -l lv00 hdisk1 hdisk2
```

-l - Indicates which LV is moved. All the LVs would be moved from *hdisk1* to *hdisk2* without this parameter.

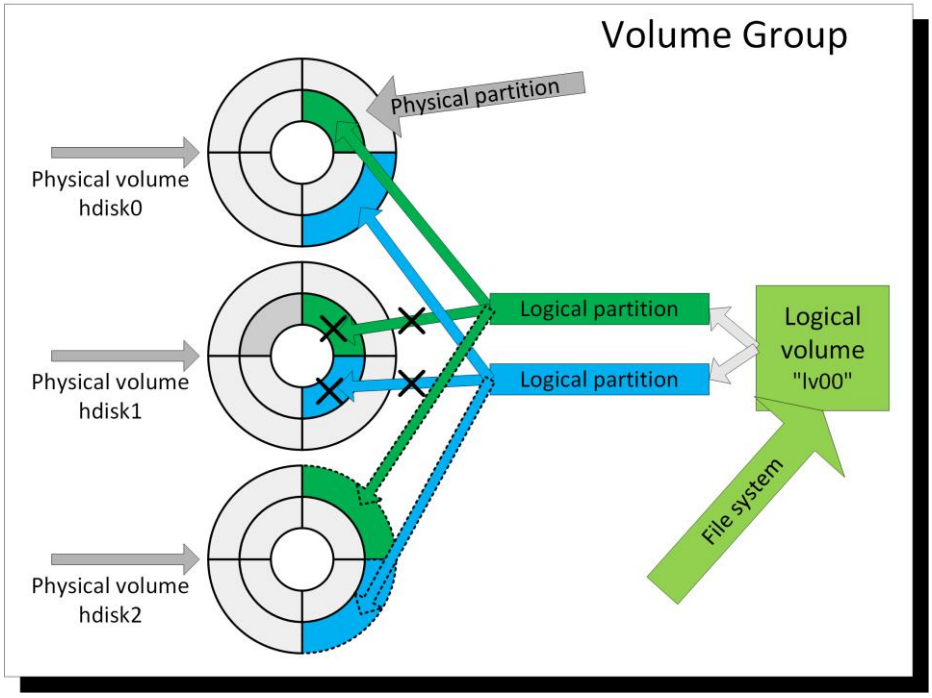

Figure 7-13: Logical Volumes migration between disks.

In the above example, the operations are performed on a logical volume that is mirrored. You may consider that a mirror exists if one logical partition points to two physical partitions (see above figure). After issuing the *migratepv -l lv00 hdisk1 hdisk2* command, all the physical partitions of the logical volume *lv00* are transferred from the *hdisk1* to the *hdisk2*. The physical partitions of this logical volume that are on the *hdisk0* remain unchanged.

It is important to note that moving the logical volumes as described above can only be done within the same volume group. Although it is possible to move a logical volume to another volume group, it cannot be done "on the fly." Such an operation would require copying the logical volume (*cplv*) and then replacing the original with the copy.

Copying a logical volume

Unlike migrating a volume (*migratepv*), copying can take place between any logical volume groups. Volume groups can vary in terms of the size of the physical partition. When copying logical volumes, use the *cplv* command or the *smit cplv* menu. The disadvantage of this tool when compared with *migratepv* is that the logical volumes must be closed during the copying. This means that no processes can work on them, and if there are file systems, they cannot be mounted.

Copying a logical volume is a simple operation. Much more interesting is the process for moving a file system, including a logical volume, to another volume group.

An example task is to move the */new* file system from the *newVG* volume group to the *new2VG*. The volume groups look like this:

```
# lsvg -l newVG
newVG:
LV NAME             TYPE       LPs   PPs   PVs  LV STATE      MOUNT POINT
loglv01             jfslog     1     1     1    open/syncd    N/A
lv00                jfs        38    38    1    open/syncd    /new

# lsvg -l new2VG
new2VG:
loglv02             jfslog     1     1     1    closed/syncd  N/A
```

To achieve this goal, you need to follow several steps:

1. Unmount the */new* file system. The *open* state indicates that it is mounted.

   ```
   # umount /new
   ```

2. Copy the logical volume *lv00* containing the */new* file system to the *new2VG* volume group. Give the name of the copy (in this case *lv00_copy*), otherwise the system will give the default name.

   ```
   # cplv -v new2 -y lv00_copy lv00
   cplv: Logical volume lv00 successfully copied to lv00_copy .
   ```

   After copying the volume group, it should look like this:

   ```
   # lsvg -l newVG
   newVG:
   LV NAME             TYPE       LPs   PPs   PVs  LV STATE      MOUNT POINT
   loglv01             jfslog     1     1     1    closed/syncd  N/A
   lv00                jfs        38    38    1    closed/syncd  /new

   # lsvg -l new2VG
   new2VG:
   ```

```
LV NAME             TYPE       LPs   PPs   PVs  LV STATE      MOUNT POINT
lv00_copy           jfs        304   304   1    closed/syncd  N/A
loglv02             jfslog     1     1     1    closed/syncd  N/A
```

You may notice that the copied logical volume consists of more logical and physical partitions than the original volume. The reason for this difference is the smaller size of the physical partition in the *new2VG* volume group than in the *newVG*.

3. Modify the entries in the */etc/filesystems* file as shown in Table 7-7.

Table 7-7: Moving a file system between volume groups.

Original entry	Modified entry
/new: dev = /dev/lv00 vfs = jfs log = /dev/loglv01 mount = true options = rw account = false	/new: dev = /dev/lv00_copy vfs = jfs log = /dev/loglv02 mount = true options = rw account = false

The original logical volume, which contains the file system, has been replaced with a copy. It was also necessary to change the file system log, since each file system must use a log that is in the same volume group.

Of course, rather than changing an existing entry in */etc/filesystems*, you can add a new one and then define a new mount point. You can operate on both file systems independently.

After performing the above steps, the file system is in the new volume group. It can be mounted *(mount /new)*. Once you mount and validate the file system, you can delete the original logical volume.

Reorganize the logical volumes in a volume group

When creating a logical volume, an allocation policy (*INTRA-POLICY*) is specified that indicates which part of the disk to allocate. As mentioned earlier, this parameter only matters if the operating system directly uses spinning disks. Nowadays, as most systems use disk arrays and SSD disks, this parameter has lost its importance. Nevertheless, it is worth knowing about the possibilities it offers.

It is often the case that after a lengthy observation of the system, one can conclude that the selected organization is not optimal. In such a situation, it is possible to change the allocation policy and then reorganize the distribution of the logical volumes. You can perform a reorganization on open logical volumes, such as those that contain the currently mounted file system. The only negative effect of reorganization is the temporary decrease in I/O operations.

It is probably best to consider an example of a reorganization:

There is a *testlv* logical volume that contains a */testlv* file system. This is one of the most commonly used file systems, and the operations on it are taking too long. The logical volume placement is as follows.

```
# lslv -l testlv
testlv:/testlv
PV                COPIES        IN BAND       DISTRIBUTION
hdisk0            010:000:000   100%          010:000:000:000:000
```

As you can see, the *testlv* logical volume is on the edge of the disk. Such a position would only guarantee performance for rare but large reads and writes, which is not the case here. The most effective solution to this problem is to move the logical volume to another part of the disk, which guarantees better performance. The most appropriate part is its center, which guarantees the shortest data access.

The first step in such cases is to change the policy (*INTRA-POLICY*). You can do this with the */usr/sbin/chlv -a 'c' testlv* command or the *smit chlv1* menu, as shown in Figure 7-14:

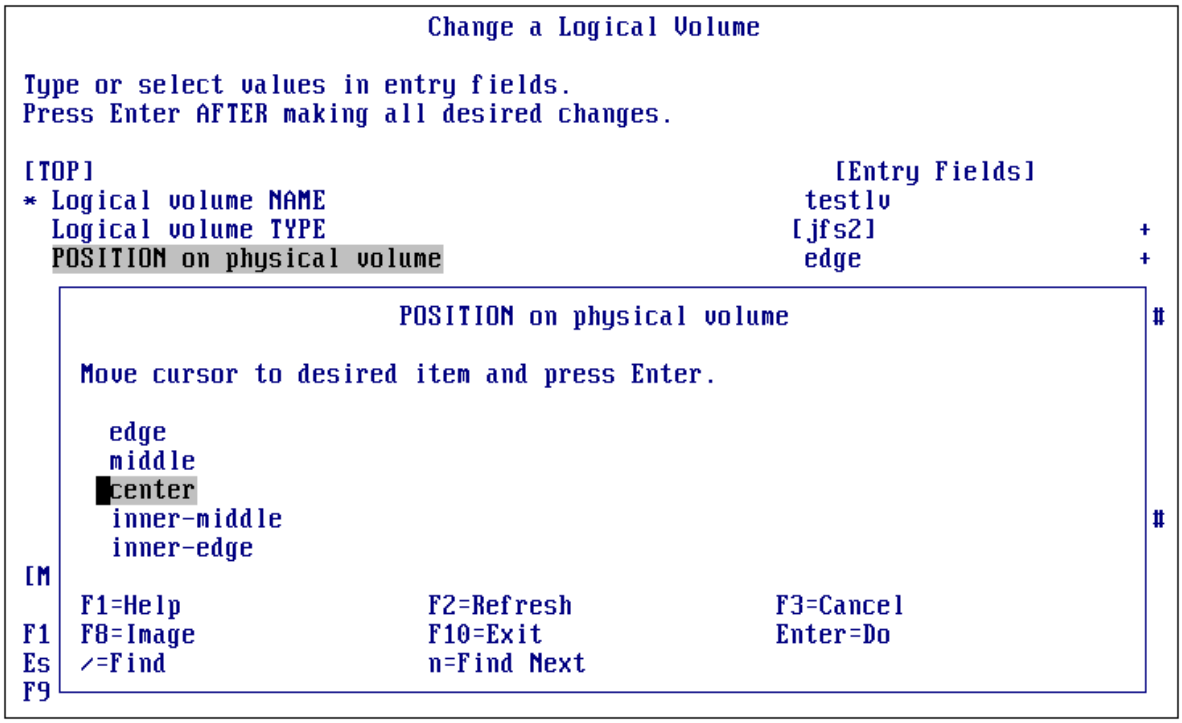

Figure 7-14: smit chlv - Change the policy for PP placement on disk.

The above figure shows a policy change. This change orders the creation of new physical partitions of the *testlv* logical volume in the center of the disk. However, this operation does not yet move the current physical partitions into place according to the policy. When you change the policy, you notice that the *lslv -l* command shows the changed logical volume information:

```
# lslv -l testlv
testlv:/testlv
PV                COPIES        IN BAND       DISTRIBUTION
hdisk0            010:000:000   0%            010:000:000:000:000
```

The *IN BAND* changed from *100%* to *0%*. This means that the distribution of the physical partitions is *0%* consistent with the policy. The policy places all the PPs in the middle of the disk, but currently they are on the edge. If one of these ten physical partitions were in the middle of the disk, the *IN BAND* would be *10%*.

The next step is to begin the process of reorganizing the logical volume that moves the physical partitions so that they are aligned with the new policy. If this is not possible due to a lack of space in a specific part of the disk, then the physical partitions will be placed as close as possible to that part (for example, they will be moved to the *middle* or *inner middle* instead the *center*). You can start the reorganization using the *reorgvg* command:

```
# reorgvg rootvg testlv
0516-962 reorgvg: Logical volume testlv migrated.
```

For a reorganization to be performed, the logical volume must have the *RELOCATABLE* parameter set to *yes* (you can check this using the *lslv LVname* command). After the command is run, you may notice that the physical partition (*DISTRIBUTION*) is changed so as to be policy compliant (*IN BAND = 100%*).

```
# lslv -l testlv
testlv:/testlv
PV                COPIES        IN BAND       DISTRIBUTION
hdisk0            010:000:000   100%          000:000:010:000:000
```

From now on, the I/O operations on the file system */testlv* should be performed faster. This rule applies to physical volumes that are physical spinning disks. The placement is not relevant for physical volumes that are LUNs from the disk array. Due to how disk arrays work, in most cases we do not know whether the *edge* is actually the *edge* and the *center* is actually the *center*.

The reorganization can affect the whole volume group. To perform this operation, you must first change the allocation policy for individual logical volumes in the group, and then run the *reorgvg VGname* command.

Metadata synchronization

Information about volume groups and logical volumes is always stored in two places:

1. In the Database Object Data Manager (ODM) database.
2. In the Volume Group Descriptor Area (VGDA), that is, at the beginning of each physical volume in the volume group.

It may happen that the data in both places are different. This may result in the logical volume information being incorrectly displayed. In such a case, you should synchronize the information for the entire volume group or selected logical volumes. You can do so using the *synclvodm* command, for example:

```
# synclvodm -v rootvg hd5 hd6   # Synchronization of information about 2 LVs in the rootvg group.
synclvodm: Physical volume data updated.
synclvodm: Logical volume hd5 updated.
synclvodm: Logical volume hd6 updated.
```

-v - This parameter shows the updated status of individual logical volumes on the screen.

Synchronization is not a dangerous operation. However, just in case, you can perform an ODM database backup by copying the files from the following directories to a safe place:

```
/usr/lib/objrepos
/usr/share/lib/objrepos
/etc/objrepos
```

Physical volumes (PVs)

You can identify a physical volume as a physical disk. In some cases, this identification will be fully correct. Yet, it is not always true. A PV is a logical structure, and what it represents is not always a physical disk. The physical volume can also be a logical structure that is presented to the operating system by an external or internal disk array. Such a structure can be a fragment of several disks working as a RAID, or any other thing that allows you to read and write the data.

The physical volume is usually seen by the system as a */dev/hdiskX* device. For LUNs presented by some disk arrays, the device name may be different. A physical volume is divided into fixed-size physical partitions (PP). Their size is determined by the volume group to which the PV belongs. This means that all the physical volumes in a volume group are divided into physical partitions of the same size. Their size is a power of two, and it ranges from 1 MB to 128 GB (for a *scalable* volume group).

Parameters of the physical volume

You can obtain basic information about physical volumes using the *lspv* command. It displays the PV name, the PVID (physical volume identifier), the volume group to which it belongs, and its status.

```
# lspv
hdisk0      0056072ad41d1b79                rootvg          active
hdisk1      0056072ad5b7570f                rootvg          active
hdisk2      0056072ad5ee0985                datavg          active
```

- ***STATUS PV*** - Shows whether the physical volume works properly. If a volume group is on, it is in one of several states:
  - *active* - The physical volume works properly.
  - *missing* - The system does not see the physical volume.
  - *removed* - The physical volume has been removed.
  - *concurrent* - This status is only for clusters. It means that a given physical volume can be used simultaneously by several systems.
- ***PVID*** - The identifier of the physical volume represented by a hexadecimal number, for example, 0056072ad41d1b79. For a disk or logical structure to become a physical volume and be used by a Logical Volume Manager, it must have a physical volume identifier. Sometimes, after allocating the disk to the server and running the configurator (*cfgmgr*), it is assigned an identifier. If this does not happen, you need to allocate this identifier manually by calling:

  ```
  # chdev -l hdiskX -a pv=yes
  ```

  You can obtain detailed information about a specific physical volume using the *lspv PVname*

command.

```
# lspv hdisk0

PHYSICAL VOLUME:    hdisk0                  VOLUME GROUP:     rootvg
PV IDENTIFIER:      00f62177ff249ff1 VG IDENTIFIER   00f6217700004c0000000157d28ec368
PV STATE:           active
STALE PARTITIONS:   0                       ALLOCATABLE:      yes
PP SIZE:            16 megabyte(s)          LOGICAL VOLUMES:  12
TOTAL PPs:          511 (8176 megabytes)    VG DESCRIPTORS:   2
FREE PPs:           156 (2496 megabytes)    HOT SPARE:        no
USED PPs:           355 (5680 megabytes)    MAX REQUEST:      256 kilobytes
FREE DISTRIBUTION:  21..00..00..33..102
USED DISTRIBUTION:  82..102..102..69..00
MIRROR POOL:        None
```

Most of the parameters are intuitive or have already been discussed. However, there are a number of parameters still worth mentioning:

- ***STALE PARTITIONS*** - The number of damaged physical partitions.
- ***FREE DISTRIBUTION*** - The amount of free space in the following areas of the disk. Starting from the left:
  - *Edge*.
  - *Middle*.
  - *Center*.
  - *Inner Middle*.
  - *Inner Edge*.
- ***USED DISTRIBUTION*** - This has the opposite meaning to ***FREE DISTRIBUTION***.
- ***ALLOCATABLE*** - Determines whether the system can allocate further physical partitions on a given physical volume. The ability to allocate can be changed using the command *chpv -a [n | y] PV name*.
- ***LOGICAL VOLUMES*** - The number of LVs that are at least partially on the PV.
- ***VG DESCRIPTORS*** - The number of physical partitions occupied by the VGDA.

Distribution of logical volumes (lspv -l)

The *lspv -l PV* command shows the logical volumes that are at least partially located on the listed physical volume. An example of this command follows:

```
# lspv -l hdisk0
hdisk0:
LV NAME               LPs     PPs     DISTRIBUTION          MOUNT POINT
hd6                  32      32      00..32..00..00..00    N/A
hd5                  2       2       02..00..00..00..00    N/A
hd4                  21      21      00..00..21..00..00    /
hd8                  1       1       00..00..01..00..00    N/A
lg_dumplv            64      64      64..00..00..00..00    N/A
hd11admin            8       8       00..00..08..00..00    /admin
livedump             16      16      16..00..00..00..00    /var/adm/ras/livedump
hd10opt              2       2       00..00..01..01..00    /opt
hd9var               11      11      00..06..01..04..00    /var
hd2                  127     127     00..64..63..00..00    /usr
hd1                  1       1       00..00..01..00..00    /home
hd3                  70      70      00..00..06..64..00    /tmp
```

Key parameters:

- ***LV NAME*** - The name of the logical volume that is on the disk. The fact that a given volume is listed here does not imply that it is entirely located on the examined physical volume. The logical volume can be placed on multiple physical volumes.
- ***LPs*** - The number of logical partitions of the logical volume that are on the disk. You cannot use this information to specify the size of a logical volume, since it can be spread across multiple disks.
- ***PPs*** - The number of physical partitions that are located on the disk. The above example shows that for each LV, the number of physical partitions is equal to the number of logical partitions. In this case, this does not mean that each LV has only one copy, since all the data presented after this command apply only to the disk being examined.
- ***DISTRIBUTION*** - The distribution of the logical volume on the disk. The numbers represent the number of physical partitions occupied by the logical volume in the subsequent disk areas.

Physical volume map (lspv -p)

The *lspv -p -PVname* command provides a detailed disk space map containing information about the logical volume types and mount points. An example of this command follows:

```
# lspv -p hdisk0
hdisk0:
PP RANGE  STATE   REGION        LV NAME             TYPE       MOUNT POINT
  1-2     used    outer edge    hd5                 boot       N/A
  3-23    free    outer edge
 24-39    used    outer edge    livedump            jfs2       /var/adm/ras/livedump
 40-103   used    outer edge    lg_dumplv           sysdump    N/A
104-135   used    outer middle  hd6                 paging     N/A
136-141   used    outer middle  hd9var              jfs2       /var
142-205   used    outer middle  hd2                 jfs2       /usr
206-206   used    center        hd8                 jfs2log    N/A
207-208   used    center        hd4                 jfs2       /
209-217   used    center        hd2                 jfs2       /usr
218-218   used    center        hd9var              jfs2       /var
219-224   used    center        hd3                 jfs2       /tmp
225-225   used    center        hd1                 jfs2       /home
226-226   used    center        hd10opt             jfs2       /opt
227-234   used    center        hd11admin           jfs2       /admin
235-235   used    center        hd4                 jfs2       /
236-237   used    center        hd2                 jfs2       /usr
238-255   used    center        hd4                 jfs2       /
256-307   used    center        hd2                 jfs2       /usr
308-311   used    inner middle  hd9var              jfs2       /var
312-312   used    inner middle  hd10opt             jfs2       /opt
313-376   used    inner middle  hd3                 jfs2       /tmp
377-409   free    inner middle
410-511   free    inner edge
```

Key parameters:

- ***PP RANGE*** - The range of physical partitions that a given line describes. For example, the last line describes the physical partitions from 410 to 511.

- ***STATE*** - The status of the PPs:
  - ***free***.
  - ***used***.
  - ***stale***.
  - ***VGDA*** - Reserved for the VGDA (Volume Group Descriptor Area). This type of physical partition only occurs when the area at the beginning of the PV is not sufficient to store all the volume group configuration data. Thus, it can usually be found in large volume groups that contain many physical volumes.
- ***REGION*** - This indicates the disk area in which the specified physical partitions are located.

Adding a physical volume

The process for adding a physical volume usually begins with presenting a disk to the operating system. This action appears different for internal drives, external storage, and virtualization, so we will not focus on it.

After doing this, run the *cfgmgr* configuration program. The program will detect the new disk in the system and add it to the configuration. The detected disk may not have a physical volume identifier (PVID), which will prevent it from being used by the *Logical Volume Manager*. This can be checked by reading the list of physical volumes:

```
# lspv
hdisk0      0056072ad41d1b79          rootvg          active
hdisk1      0056072ad5b7570f          rootvg          active
hdisk2      none
```

In the above example, *hdisk2* does not have a PVID. To correct this, you must run the *chdev hdisk2 -l -a pv = yes* command. If you do not specify a disk ID, it will be specified automatically when you add a disk to the volume group.

When added to the rootvg, it will look like this:

```
# lspv
hdisk0      0056072ad41d1b79          rootvg          active
hdisk1      0056072ad5b7570f          rootvg          active
hdisk2      0056072ad5ee0985          rootvg          active
```

Removing a physical volume

If you need to remove a disk from the system, you need to do a few things first.

1. Remove all the logical volumes (*rmlv* command) from the disk and then remove the disk from the volume group (*reducevg* command).
2. Change the state of the disk from *available* to *defined*, and delete it from the ODM database (*rmdev -dl PV name*).

After performing the above steps, the disk configuration is removed from the system, and the disk can be unmapped or pulled out if it is a physical disk.

If the disk is unmapped without the above steps being completed, the system will retain information about it and its status will change to *missing*. The system can be cleaned later (if the disk did not contain the only copy of the data). However, leaving the information about the removed disk may prove a nuisance during further activities.

Changing a physical volume

Use the *chpv* command to change the parameters of the physical volume. You can make a number of changes:

- Enabling or disabling the physical partition allocation on a physical volume. This option is described as *ALLOCATABLE* in the *lspv PVname* command, and it is enabled by default. This means that on a given PV, you can create logical volumes and extend existing ones. You can disable this option using the *chpv -a n PVname* command and enable it using the *chpv -a y PVname* command.
- Enabling and disabling a physical volume. With this option, you can render a PV unavailable. This blocks the execution of an I/O operation on it, although it does not block the ability to manage it. To make a PV unavailable, run *chpv -v r PVname*. To make it available, run *chpv -v PVname*.
- Specify a disk as a "hot spare" for the volume group to which it belongs. Simultaneous to making the disk a "hot spare," its ALLOCATABLE parameter is turned off, which prevents the allocation of the physical partitions of the PV to a logical volume. If one copy of the logical volume is corrupted (along with a disk failure), it can be automatically rebuilt to the "hot spare" disk that is available for such events. Only an empty physical volume can be used as this type of disk. A "Hot spare" is created using the *chpv -h y PVname* command. You can change it back to a normal disk using the *chpv -h n PVname* command.

It is important to note that using the *chpv* command requires some space in the */tmp* file system. If the command fails, first check to see if there is sufficient free space.

AIX, SAN, and MPIO

By comparing the existing and prior servers, you can observe a large variation in the approach to disk space management. Once, a server was considered to be in unity with its disk space. It had its own directly attached disk resources to which it had exclusive access. This kind of storage access is known as DAS (Direct Attached Storage). In some cases, this is still a solution. It is increasingly common when using SSDs to shorten the disk access time.

During the next stage of IT development, the number of systems within organizations grew. At the same time, the systems became increasingly variable. Their development accelerated, and their life cycle was shortened. The disadvantages of DAS solutions became increasingly burdensome. Disk space was hard to share. If the server did not have enough disk space, it had to be expanded. If it had too much

disk space, the space was usually not used. These factors have led to the development of storage networking. This has in turn led to the elimination of known disadvantages in the form of disk islands as well as the spread of external storage that isolates part of the disk space management functionality from servers.

When talking about network access to storage, we can think of many different technologies. Some operate on files. Some work on data in a traditional block form, which is consistent with operating on a physical disk from an operating system perspective. A typical technology that allows access to file data is NAS:

NAS (Network Attached Storage) - All technologies that allow access to files over a network (usually a LAN). In this case, file systems are defined on external devices in relation to our system, for example, on external disk arrays or NAS servers. The system operates on files shared using protocols such as NFS (*Network File System*) associated with UNIX or CIFS (*Common Internet File System*) associated with Windows.

The next group are technologies used to access data in block form. This type of access has one common feature. It still uses SCSI (*Small Computer Systems Interface*) to work with data. Hence, the definition of a LUN (Logical Unit). As in the case of classical SCSI, a LUN represents the disk space of a certain size presented in this case by the external storage. Historically, there has been a change in the physical interface for SCSI, although there has been no change in the means of communication with the disk space. A number of technologies can be distinguished in this group:

- **SAN (Storage Area Network)** - A universal solution for block access to data. Most large organizations rely on SANs for block access. The main feature of a SAN is the use of the network to communicate with the disk space while generating only small delays in data access. The SAN is based on the fibre channel protocol, which in turn serves as the "pack" for the SCSI protocol, which is used to communicate with external storage.
- **iSCSI (Internet SCSI)** - A common solution, although it does not provide such low latency or high data security as a SAN. The reason for this is the transmission medium, which is a TCP/IP network that is more complex and has more delays in transmissions. In this case, the SCSI protocol is transferred over the TCP/IP network.
- **FCoE (Fiber Channel over Ethernet)** - A less common solution than the first two. In this case, the SCSI protocol is transmitted in the fibre channel frames. In turn, the fibre channel is transmitted in Ethernet frames, in a lower layer than in the iSCSI. This provides lower latency than the iSCSI. Special switches and adapters for Converged Ethernet are needed for this solution.
- **FCIP (Fiber Channel over IP)** - A solution that transfers SCSI in fiber channel packaging over a TCP/IP network. This solution is typically used to stretch SANs over long distances.

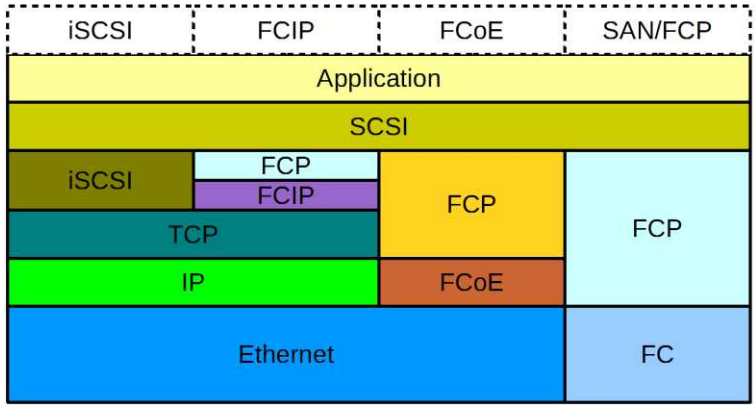

Figure 7-15: Summary of the block access protocols.

The most commonly used data block technology is the *Storage Area Network*. Further information in this chapter hence applies to that technology.

AIX and Storage Area Network

To understand how AIX works in the context of storage, it is worth mentioning a few facts about SANs. A Storage Area Network is a universal solution used by large organizations to manage disk space. It works based on the fibre channel protocol, which is the carrier of the SCSI protocol. This solution is resistant to the failure of individual components, and it is also very scalable. It can operate at high speeds achieved by multiplexing access lines, while generating only small delays.

A SAN can operate on multiple switches, with many different topologies and configurations.

"Dual fabric" configurations are most commonly used for reliability. Such configurations physically separate two independent and separately configured SANs, which are known as "fabrics." A dual fabric configuration example that uses two SAN switches is given in Figure 7-16.

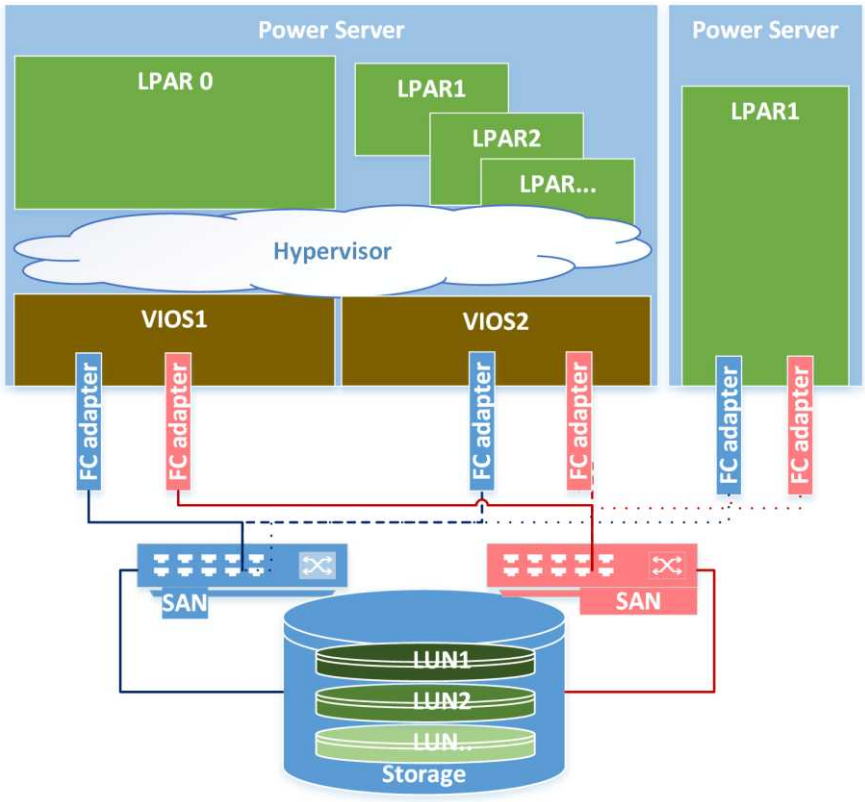

Figure 7-16: Dual fabric configuration.

The above figure shows an example of a simple configuration. The virtualized server shown is equipped with four Host Bus Adapters (HBA). Two adapters are connected to one fabric and two to the other. This case shows the high availability of the solution, that is, the failure of a single switch, entire fabric, or several HBAs does not result in the disruption of disk access. It also shows the scalability of performance, which is achieved through the multiplication of access interfaces on the server and external storage.

From a configuration perspective, a few more steps are needed to render the LUN available to the server via the disk array:

1. **Zoning**

Zoning creation involves entering the proper configuration of each fabric. This configuration allows devices to communicate with others in the same fabric. Communication can be established by specifying the physical ports to which the server and storage are connected, or by specifying addresses:

```
# lscfg -vl fcs0 | grep Network
        Network Address.............10000000C99ACBF4
```

Please note that this address is only used for configuration purposes. The shorter, 24-bit FCID (*Fiber Channel ID*) addresses are used for addressing the frames that are sent on the SAN. During the HBA port logon process to SANs, the WWNs are mapped to the FCIDs in order to use as little management

information as possible during the frame transmission (24 instead of 64 bits). Address mapping is completely transparent to the system in which we use the WWN address.

2. LUN Masking

Zoning enabled transmission between devices in the SAN. The next step in our example is to configure the external storage. At the storage level, you also need to specify which devices/servers should see the LUN. In this case, it also takes place at the WWN address level. LUN masking specifies which LUN should be presented to which WWNs by which storage ports.

From the operating system perspective, it is important that this configuration is presented as multiple access paths to a given LUN. In the following case, you can see that access is via four paths, two for each HBA. This means that a single port on the server side communicates with two ports on the storage side.

```
# lspath | grep hdisk0
Enabled hdisk0 fscsi0
Enabled hdisk0 fscsi0
Enabled hdisk0 fscsi1
Enabled hdisk0 fscsi1
```

When using multiple paths, disk access is efficient and high availability is maintained. From the operating system perspective, the key devices that represent the HBA and the SAN are the *fcsX* and the *fscsiX*, for example:

```
# lsattr -El fcs0
DIF_enabled   no          DIF (T10 protection) enabled                      True
bus_intr_lvl              Bus interrupt level                               False
bus_mem_addr  0xffe76000 Bus memory address                                 False
bus_mem_addr2 0xffe78000 Bus memory address                                 False
init_link     auto        INIT Link flags                                   False
intr_msi_1    254501      Bus interrupt level                               False
intr_priority 3           Interrupt priority                                False
lg_term_dma   0x800000    Long term DMA                                     True
max_xfer_size 0x100000    Maximum Transfer Size                             True
num_cmd_elems 500         Maximum number of COMMANDS to queue to the adapter True
pref_alpa     0x1         Preferred AL_PA                                   True
sw_fc_class   2           FC Class for Fabric                               True
tme           no          Target Mode Enabled                               True

# lsattr -El fscsi0
attach       switch    How this adapter is CONNECTED         False
dyntrk       yes       Dynamic Tracking of FC Devices        True+
fc_err_recov fast_fail FC Fabric Event Error RECOVERY Policy True+
scsi_id      0x1134e   Adapter SCSI ID                       False
sw_fc_class  3         FC Class for Fabric                   True
```

There are a number of important parameters to consider:

- ***num_cmd_elems*** - The number of messages (reads/writes, control messages) that can be sent simultaneously through the interface. All subsequent messages will wait until the previous ones are confirmed. Each acknowledgment indicates that the next message may be sent. This parameter can be very important from a performance perspective, although there is no single correct setting. It depends on the performance of the storage with which the system communicates. Normally, you retain the defaults, or else you will work out your own settings

after tests.

- ***max_xfer_size*** - The maximum size of a single read/write operation. High values are usually important for backup purposes.
- ***dyntrk*** - The *yes* setting makes the device driver aware of and resistant to potential changes in the SAN architecture. This setting is in line with best practices.
- ***fc_err_recov*** - Setting this parameter to *fast_fail* allows the device driver to react quickly to problems on one of the access paths. The driver disables that path, allowing the system to work on the remaining paths. This setting is also in line with best practices.

A useful command that displays a lot of configuration data and statistics is *fcstat*. You can see, for example, the adapter speed (supported and negotiated), WWN addresses, FCID address, and adapter statistics. An example of the command follows:

```
# fcstat fcs0

FIBRE CHANNEL STATISTICS REPORT: fcs0

Device Type: 8Gb PCI Express Dual Port FC Adapter (df1000f114108a03) (adapter/pciex/df1000f114108a0)
Serial Number: 1C019087F8
Option ROM Version: 02781174
ZA: U2D1.11X4
World Wide Node Name: 0x20000000C9B01108
World Wide Port Name: 0x10000000C9B01108

FC-4 TYPES:
  Supported: 0x0000012000000000000000000000000000000000000000000000000000000000
  Active:    0x0000010000000000000000000000000000000000000000000000000000000000
Class of Service: 3
Port Speed (supported): 8 GBIT
Port Speed (running):   4 GBIT
Port FC ID: 0x01134e
Port Type: Fabric
Attention Type:   Link Up
Topology:  Point to Point or Fabric

Seconds Since Last Reset: 14333334

        Transmit Statistics     Receive Statistics
        -------------------     ------------------
Frames: 372083470               819730134
Words:  150201496064            348493861376
# All unnecessary information has been omitted.
```

MPIO (Multipath I/O)

MPIO means that access to the disk is possible with multiple paths. A classic example is the SAN discussed above, but that is not the only example. Another popular case in which MPIO is used is virtualization (dual VIOS configuration). In this configuration, access to a particular disk resource is based on MPIO and two paths, one through each Virtual I/O Server.

The paths can be managed by the software built into the operating system or the drivers provided by the storage vendor. The drivers provided by the storage vendor provide more disk management options, including the dynamic balancing of disk operations between all paths, presenting more information about LUNs, or even the management of advanced features of the storage, such as data replication or snapshots. The LUN provided by the storage array is a disk within the operating system, typically the *hdiskX*.

AIX uses a kernel extension known as PCM (Path Control Module) to manage multiple paths (MPIO). AIX provides the default PCM, and each storage vendor can write its own PCM for its device. The supported devices and their default drivers can be displayed using the following command:

```
# manage_disk_drivers -l
Device              Present Driver        Driver Options
2810XIV             AIX_AAPCM             AIX_AAPCM,AIX_non_MPIO
DS4100              AIX_APPCM             AIX_APPCM,AIX_fcparray
DS4200              AIX_APPCM             AIX_APPCM,AIX_fcparray
DS4300              AIX_APPCM             AIX_APPCM,AIX_fcparray
DS4500              AIX_APPCM             AIX_APPCM,AIX_fcparray
DS4700              AIX_APPCM             AIX_APPCM,AIX_fcparray
DS4800              AIX_APPCM             AIX_APPCM,AIX_fcparray
DS3950              AIX_APPCM             AIX_APPCM
DS5020              AIX_APPCM             AIX_APPCM
DCS3700             AIX_APPCM             AIX_APPCM
DS5100/DS5300       AIX_APPCM             AIX_APPCM
DS3500              AIX_APPCM             AIX_APPCM
# All unnecessary information has been omitted.
```

The *AIX_AAPCM* driver differs from the *AIX_APPCM* driver in terms of the way it uses paths (*AA = Active/Active*, *AP = Active/Passive*).

When using built-in drivers, depending on the types of devices available to choose from, there may be several algorithms for using the paths. The algorithms that can be used in the context of a disk can be read from the disk properties, for example:

```
# lscfg -vl hdisk1 | head -1
  hdisk1     U78A0.001.DNWHWGD-P1-C2-T2-W50050768013047ED-L1000000000000  MPIO IBM 2145 FC Disk

# lsattr -R -l hdisk1 -a algorithm
fail_over
round_robin
shortest_queue
```

As you can see in the above outputs, *hdisk1* is a LUN presented from the *IBM 2145* storage. You can verify that this device is a *San Volume Controller* (SVC). The drive can operate on paths with three different algorithms:

- ***fail_over*** - The algorithm for high availability only. I/O operations are performed with only one path. If it fails, then another available path with the highest priority is used.
- ***round_robin*** - I/0 operations are distributed to all the available paths according to their priority. The paths that have the same priority perform the same amount of I/O operations, while the paths with higher priority perform more I/O operations in proportion to their priority.
- ***shortest_queue*** - The algorithm distributes traffic on all tracks, preferring those that have the least elements in the queue. In this case, priority is ignored.

You can use the *lspath*, *lsmpio, rmpath*, *chpath*, and *mkpath* commands to manage paths. An example of *lspath* follows:

```
# lspath | grep hdisk1
Enabled hdisk0 fscsi0
Enabled hdisk0 fscsi0
Enabled hdisk0 fscsi1
Enabled hdisk0 fscsi1
```

You can manage the paths using the *chpath* command. In the following output, two paths that go through *fscsi0* have priority 1. So, in the case of the *fail_over* algorithm, the path with the lower number will be used. In the case of the *round_robin* algorithm, the I/O operations will be distributed equally to all the paths.

```
# lspath -AHE -l hdisk1 -p fscsi0  -i 0
attribute value              description  user_settable

scsi_id   0x10100            SCSI ID      False
node_name 0x50050768010047ed FC Node Name False
priority  1                  Priority     True

# lspath -AHE -l hdisk1 -p fscsi0  -i 1
attribute value              description  user_settable

scsi_id   0x10200            SCSI ID      False
node_name 0x500507680100484d FC Node Name False
priority  1                  Priority     True
```

By changing the priority of path number 1, we will change the situation so that another path becomes active.

```
# lsmpio | grep hdisk2   # Path status prior to modification ("Sel" = active path).
hdisk1  0        Enabled  Sel,Opt      fscsi0  50050768013047ed,2000000000000
hdisk1  1        Enabled  Opt          fscsi0  50050768012047ed,2000000000000
hdisk1  2        Enabled  Opt          fscsi1  50050768013047ed,2000000000000
hdisk1  3        Enabled  Opt          fscsi1  50050768012047ed,2000000000000

# chpath -l hdisk1 -i 0 -a priority=2   # Modification of priority.
path Changed



# lspath -AHE -l hdisk1 -p fscsi0  -i 0   # The priority is changed to 2.
attribute value              description  user_settable

scsi_id   0x10100            SCSI ID      False
node_name 0x50050768010047ed FC Node Name False
priority  2                  Priority     True
```

```
# lsmpio | grep hdisk2   # State of the paths after modifying the priority.
hdisk1  0        Enabled  Opt          fscsi0  50050768013047ed,2000000000000
hdisk1  1        Enabled  Sel,Opt      fscsi0  50050768012047ed,2000000000000
hdisk1  2        Enabled  Opt          fscsi1  50050768013047ed,2000000000000
hdisk1  3        Enabled  Opt          fscsi1  50050768012047ed,2000000000000
```

As you can see, path number 1 (*Sel*) has become an active path. The change will be visible after any I/O operation on the disk. The remaining paths are passive paths that wait to take control in the event of an active path failure.

The paths displayed by *lspath* can have multiple states. On the outputs of the *lspath* command given in this chapter, you can see four paths in the *enabled* state, which means that they are functional and ready for operation. There are four other possible states:

- ***disabled*** - The path is manually disabled and not used for I/O operations.
- ***failed*** - The path is corrupted and not used for I/O operations. One way to restore it is to fix the technical problem and then remove the path and rediscover it.
- ***defined*** - The path is not configured and not used for I/O operations. This state can occur after explicitly turning off the path using the *rmpath -l* command.
- ***missing*** - A path that was not detected after the system restart and existed earlier in the system.

It is important to pay special attention to the next two parameters that are set at the disk level, namely *hcheck_interval* and *hcheck_mode*. They allow for both periodic path validation and automatic restoration following path failure.

```
# lsattr -El hdisk2 | grep hcheck
hcheck_cmd      test_unit_rdy           Health Check Command            True+
hcheck_interval 60                      Health Check Interval           True+
hcheck_mode     nonactive               Health Check Mode               True+
```

- ***hcheck_mode*** - Specifies which paths should be verified. There are three settings:
  - ***nonactive*** - The paths not used for I/O operations are checked. This also applies to the paths marked as *failed*. Thus, this setting allows you to restore a previously failed path. This is the default and, in most cases, desired setting.
  - ***enabled*** - All the paths specified as *enabled* are checked, regardless of whether they are used for I/O operations or not. This mode will not automatically restore the failed paths.
  - ***failed*** - Only those paths marked as *failed* are verified. In this mode, the corruption of the backup path that is enabled but not used for I/O operations will not be detected.
- ***hcheck_interval*** - Specifies the number of seconds between subsequent path checks. A value of 0 means that the check is disabled.

If the external storage device driver is used, the path handling may be different. Advanced tools and additional commands can be available for operating on a storage array.

Chapter 8. File systems

So far, we have discussed such structures as volume groups, logical volumes, and physical volumes. However, despite their importance, they could not function without a single structure - the file system (FS).

The AIX file system is always created on a logical volume (LV). Only one file system can be created on one LV. An FS cannot be scattered across many LVs. If you do not use raw devices (i.e., you do not use disks or logical volumes directly to store data), then the number of file systems corresponds to the number of logical volumes (not including the LV for the paging space and log file systems).

The file system allows you to store data in the standard way that you are used to, that is, in the form of files and directories. There are files of different types and access to them depends on the assigned access rights. In other words, from the perspective of the user, AIX file systems behave like typical UNIX file systems, but from the perspective of the operating system side, it is more complex.

In AIX, you can create four different file systems:

- **JFS (Journaled File System)** - A file system with logging. This means that it keeps track of changes not yet committed to the file system. Today, it is used rather sporadically due to its many limitations.
- **JFS2 (Enhanced Journaled File System)** - An enhanced version of a file system. It appeared for the first time in AIX 5.1 and is still in use today.
- **CDRFS (CD ROM File System)** - A file system that represents a CD/DVD.
- **NFS (Network File System)**.

Journaled File System

This file system has existed in AIX for a long time. Due to its age and limitations, it has been replaced with an improved version, JFS2. The behavior of the JFS file system is very similar to that of JFS2, so the general file system operating principles will be shown below in the JFS example.

JFS, as its name suggests (Journaled File System), is a file system with transaction logging. This makes it less likely to lose its consistency than classic file systems. Yet, the consistency mechanism requires that the system perform additional disk operations. These operations could potentially slow down the file system, although they do not result in significant slowdowns, and usually the protection of the file system is worth it.

Each JFS can occupy only one logical volume. Its purpose is to store files in the LV in a consistent and easily accessible fashion. The minimum unit that this file system can read or write is a block of 4096 bytes (4 KB). The main file system structures are files and directories. However, there are other structures that are invisible to the user, although they are equally important for the proper performance of the file system. These structures are:

- Superblock;
- I-nodes;
- Data blocks;
- Allocation groups; and
- File system log.

Superblock

This is a necessary structure for storing basic information about the file system. It has a size of 4096 bytes, and it is stored in the first logical volume block. In the case of a superblock failure, you can lose all the data in the file system. Fortunately, its copy is located in block 31. The information stored in a superblock comprises:

- The file system size;
- The total number of data blocks in the file system;
- The file system status;
- The Allocation Group Size; and
- The file system log address.

In the case of the failure of the first block, you may receive one or more of the following messages when you try to mount the file system:

```
# mount /testjfs
mount: 0506-324 Cannot mount /dev/lv00 on /testjfs: A system call received a parameter that is not
valid.
```

The message does not tell us why we cannot mount the file system, but in such cases, we can infer that there is a problem with the superblock. You can restore it using the *fsck* (*File System Consistency Check*) command, for example:

```
# fsck -p /dev/LVname   # LVname - logical volume on which FS was placed.
log redo processing for /dev/rlv00
/dev/rlv00 (/testjfs): Superblock is marked dirty (FIXED)
/dev/rlv00 (/testjfs): 8 files 7272 blocks 222104 free
```

The *-p* parameter automatically repairs certain errors. When *fsck* encounters a damaged superblock, it restores its contents from the copy located in block 31. If you run *fsck* without the *-p* parameter, the file system will not be repaired, for example:

```
# fsck /testjfs
Not a recognized filesystem type. (TERMINATED)
```

Another way to solve the problem is to directly copy block 31 to block 1 using the *dd* command.

```
# dd count=1 bs=4k skip=31 seek=1 if=/dev/LVname of=/dev/LVname
```

The above structure also exists in the JFS2 file system, and the above operations are performed in the same way for both file systems.

I-nodes

The i-node represents each file and directory in the system. All the file parameters and pointers to the data are stored within its structure. JFS maintains the number of i-nodes specified when creating the file system. This means that there is a limit placed on the number of files in each JFS file system.

A significant amount of important information is stored in the i-node:

- File type.
  - ***-*** - Usual file.
  - ***d*** - Directory.
  - ***b*** - Block device, file buffered by the operating system.
  - ***c*** - Character device, file not buffered by the operating system.
  - ***l*** - Symbolic link.
  - ***p*** - FIFO file (first in, first out).
  - ***s*** - Socket file.
- Number of symbolic links.
- The last file access and modification times.
- File permissions.
- Addresses of logical disk blocks in which data are stored.

The larger the file, the larger the space is needed for its i-node. For i-nodes to occupy as little disk space as possible, three types of addresses are used. The system automatically selects the type of addressing based on the file size. If the file exceeds the size allocated for one type of addressing, then the following blocks are allocated with another type of addressing. There are three types of addressing within the JFS file system:

- **Direct** - Used for files that are no larger than 32 KB (8 * 4096 bytes). The i-node in this case occupies one allocation unit (i.e., 4 KB). Most of the allocation unit is occupied by the file parameters. The last 32 bytes hold eight direct addresses of the disk unit in which the data are stored. If you looked at the contents of the file in Figure 8-1, you would see the contents of the following blocks: 33, 34, 35, 38, 41, 42, and 43. Their addresses would come from the allocation unit that stores the i-node.

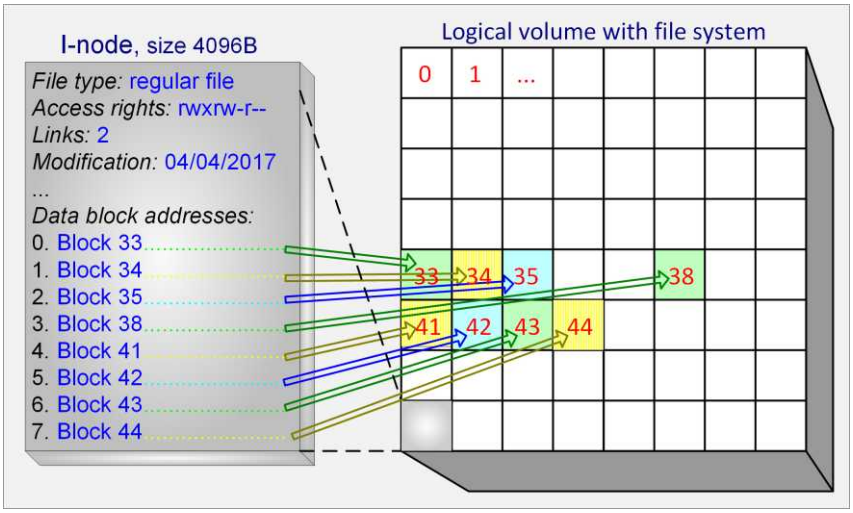

Figure 8-1: I-node, direct addressing.

This type of addressing is limited to 32 KB because in the allocation unit that stores the i-node, there is room for the eight addresses of the blocks with data. Addressing eight blocks of 4 KB gives 32 KB. The rest of the i-node is occupied by other parameters that define the file.

- **Single indirect** - Used for files larger than 32 KB, but not larger than 4 MB (1024 · 4 KB). The i-node occupies two allocation units (i.e., 8 KB). Seven addresses from the first allocation unit are not used. The eighth address points to the allocation unit, which is filled up with the addresses of the allocation units that contain data. This is shown in Figure 8-2.

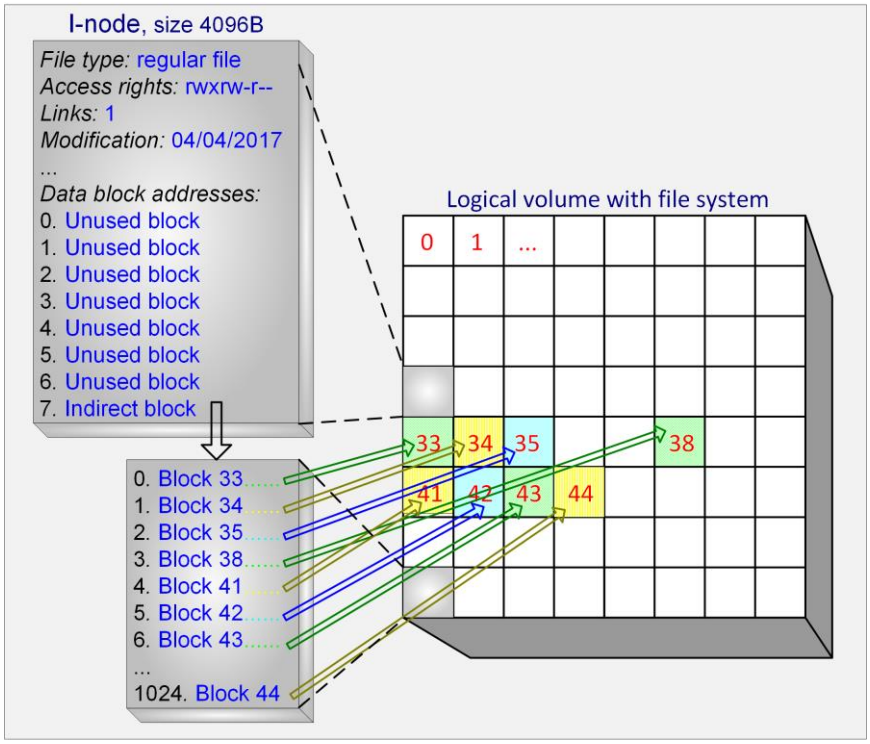

Figure 8-2: I-node, single indirect addressing.

The size is limited to 4 MB because one block of 4 KB is used to store the addresses of the allocation units that contain data. Each address consists of 4 bytes, that is, the allocation unit holds 1024 addresses. 1024 * 4 KB (allocation unit) = 4 MB.

- **Double indirect** - Used for files larger than 4 MB. Each i-node occupies at least three allocation units, and at a maximum file size it can occupy 1026 allocation units, which is more than 4 MB. A file with so large an i-node would have to be 2 GB. Seven addresses from the first allocation unit are not used. The eighth address points to the allocation unit, which is filled with addresses (1024 addresses) that indicate the next allocation units containing 512 addresses. This addressing type can handle files up to 2 GB (1024 * 512 * 4 KB). With the introduction of AIX 4.3, the size of the blocks pointed to by one address was changed from 4 KB to 128 KB. Due to this change, the maximum file size in the JFS file system has grown to 64 GB. This addressing type is shown in Figure 8-3.

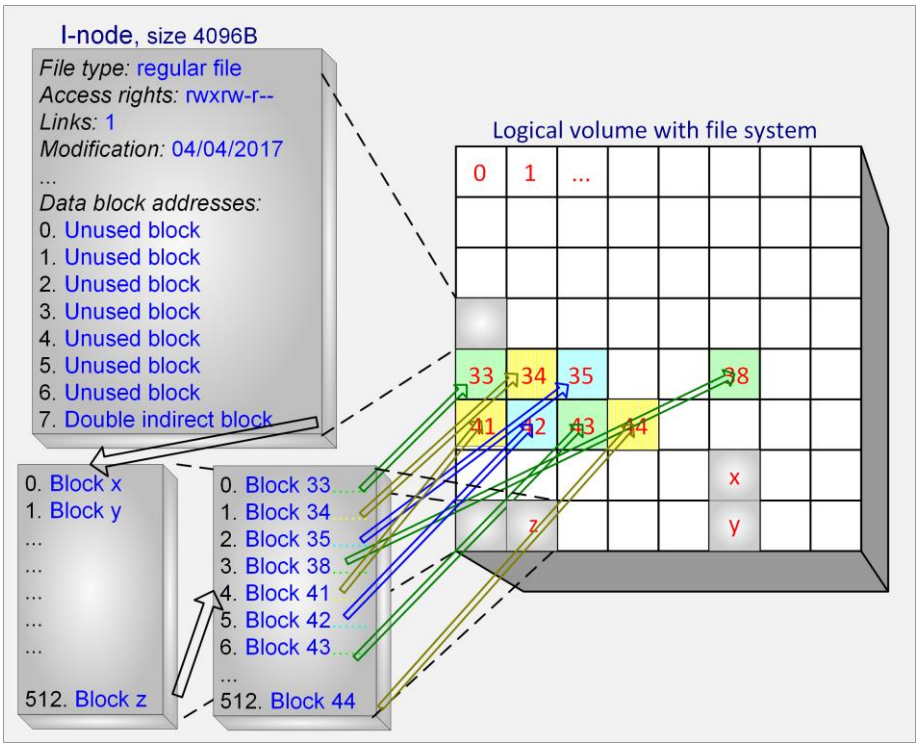

Figure 8-3: I-node, double indirect addressing.

Data blocks

The file system is divided into blocks known as allocation units. A block represents the minimum unit a system can write or read. It is a multiple of a disk block that is typically 512 bytes in size (although disk drives that use 4 KB blocks are increasingly common). In the JFS, the data block has a size of 4096 B, and it is equivalent to the memory page of the operating system.

By default, only one fragment of a file can be located in a data block. So, if the file does not have a size that is a multiple of 4096, and in most cases it does not, part of the disk space remains unused.

The loss of capacity is shown in Figure 8-4. When you save two files, one of size 6250 B and the other of size 9100 B, there are two partially used data blocks, which cannot be used to write another file. We lose capacity in this way: 1942 B + 3188 B = 5130 B.

Figure 8-4: Data blocks - Wastage of disk space.

There is a way to limit this type of loss. When you create a file system, you can define the size of the fragments into which the data block is to be split. The fragments can have a size of 512, 1024, 2048, or 4096 bytes (4096 is equal to the size of the data block, so it implies zero fragmentation). If you use fragments smaller than the size of the data block, the system will be able to save a few file ends in one data block. This will result in less disk space loss.

When writing two files of the same size, as in the previous example, for a 1024 B fragment instead of a 5130 B fragment, you lose 1034 B. The loss is reduced by placing the ends of the two files in one block, as shown in Figure 8-5.

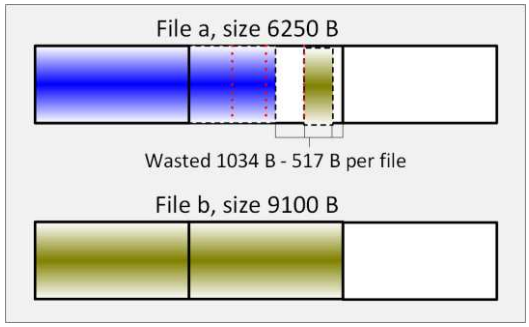

Figure 8-5: Data blocks - Using smaller fragments than the data block.

This leads to an additional advantage. In the case of file read operations, the system must read only four blocks instead of five from the disk, which reduces the number of I/O operations. However, managing such a file system requires more operations from the operating system and is therefore more labor intensive.

Allocation groups

The file system is divided into allocation groups. Each allocation group contains data blocks and the i-nodes that refer to those blocks. It can have a size of 8, 16, 32, or 64 MB.

When you create a 256 MB file system and have a 64 MB group size, you will get a file system consisting of four allocation groups (64 MB each), each of which will have its own i-nodes table. The

size of this table is determined by the NBPI (Number of Bytes Per I-node) parameter, which indirectly indicates how many files you can create on a given file system.

Due to the division into allocation groups, you can extend the file system "on the fly." When you expand the above-mentioned file system by 128 MB, the system will add two groups of allocations of 64 MB each. Each of these groups will contain a space with data blocks as well as a space with an i-node table that describes those blocks.

File system log

The JFS is a file system with logging. This means that before the file modification operation is performed, the change information first goes to the file system log. This protects the file system from losing consistency, which could occur in the event of a system crash while writing. If the record is not complete, the changes made in the i-nodes table are rolled back using log records, and the file system recovers consistency. The recovery operation is very fast and does not require a file system scan.

The logging mechanism in file systems works in a very similar fashion to the transaction log mechanism in databases. The difference is that the database log writes complete information about both metadata changes and changes to individual values in the tables, while the file system writes only the metadata to its log. There is no information in the file system log about changed data. The metadata of file systems are structures such as i-nodes, maps of free and occupied blocks, etc.

Logging provides a number of benefits:

- **Consistency of data** - When a crash occurs, data integrity is restored using the log.
- **Fast restore of consistency** - The restoring of consistency takes place at boot time, and it usually lasts from only a few seconds to a few minutes. In the case of a loss of consistency in a file system without logging, restoring consistency could take from several minutes to several hours depending on the size of the file system. Such an operation would involve scanning the whole file system.

AIX typically holds one JFS log file per volume group. It is created automatically when the first file system is created in a volume group. If both the JFS and JFS2 are used, two logs will be created separately for each file system type. This can be seen in the following output:

```
# lsvg -l rootvg
rootvg:
LV NAME             TYPE       LPs   PPs   PVs  LV STATE      MOUNT POINT
hd5                 boot       1     2     2    closed/syncd  N/A
hd6                 paging     30    60    2    open/syncd    N/A
hd8                 jfs2log    1     2     2    open/syncd    N/A
hd4                 jfs2       1     2     2    open/syncd    /
hd2                 jfs2       154   308   2    open/syncd    /usr
hd9var              jfs2       4     8     2    open/syncd    /var
hd3                 jfs2       7     14    2    open/syncd    /tmp
hd1                 jfs2       1     2     2    open/syncd    /home
hd10opt             jfs2       2     4     2    open/syncd    /opt
fslv00              jfs        8     8     1    closed/syncd  /data
loglv01             jfslog     1     1     1    closed/syncd  N/A
```

As you can see, the file system log takes up little space on the disk. Usually, it occupies just one physical

partition. When creating a JFS2 file system, it is possible to define an *inline log*. In this case, the log is inside the file system and hence there does not need to be a separate logical volume for the logs.

The log is such an important part of the file system that invalid entries in it can cause the file system to fail to mount. If such a problem occurs, it may be necessary to format the log, thereby removing all entries from it and thus re-creating it. This can be done using the *logform* command.

Enhanced Journaled File System

With the increase in the number of files stored by users and the size of those files, the capacity and performance limits of the JFS had become increasingly problematic. The improved version of the file system - the *Enhanced Journaled File System (JFS2)* - was released to overcome those problems. The construction of the JFS2 is very similar to that of the JFS. The JFS features described above are also valid for the JFS2, albeit with the following differences:

- **Directory organization** - The JFS2 uses two types of directory organization, one for directories that have a small number of files and another for directories with a large number of files:
  - For small directories, a description of their contents is kept in the i-node of the directory. This eliminates the need to use an additional disk block and thus reduces the number of disk reads for searching and referencing files.
  - For large directories, all entries (files and directories) located in the directory are indexed using a B+ tree. The key to the index is the file name. Compared to a traditional directory structure that does not use sorting, this greatly speeds up the search for files in the directory.
- **Block size** - The JFS2 allows you to create file systems of 512-, 1024-, 2048-, and 4096-byte blocks. Block size manipulation effectively replaces the block fragmentation mechanism used in the previous version of the file system. It also allows you to customize the file system to better suit your system and the associated application requirements.
- **Dynamic allocation of i-nodes** - The i-node disk space is dynamically allocated as needed and then dynamically released. This eliminates the need to specify the maximum number of files that can be created when creating a file system.
- **Maximum file and file system size** - The maximum JFS2 size has been increased.
- **Compression** - The newer version of the file system removes the ability to create compressed file systems. Due to the ever-increasing capacity of the disks and their decreasing price, this feature has become less important.

These and other differences are detailed in Table 8.1.

Table 8-1: Differences between the JFS and JFS2.

Feature	JFS2	JFS
Block size	512–4096 B	4096 (fragments 512–4096)
Maximum file size	16 TB	2 GB (JFS) 64 GB (JFS - Large File Enabled)
Maximum file system size	32 TB	1 TB
Number of I-nodes	Dynamic, without restrictions	Specified, set when creating the file system
Directory organization	B+ tree	Linear
Compression	NO	YES
The default owner	root:system	sys:sys
Extended Access Control List (ACL)	YES	YES
Limits (Quotas)	YES (from AIX-5.3)	YES
Encryption	YES (Encrypted File System)	NO
Snapshots	YES	NO

Adding a file system

There are two main types of file systems that you can create:

- **JFS (Journaled File System)** - Not typically used due to its limitations, but still available. Within the JFS you can create:
  - **"Usual" JFS** - Compatible with the JFS file system used in AIX 3. The file size in this case cannot exceed 2 GB.
  - **Compressed JFS** - All data are compressed automatically before the write to disk and then automatically decompressed when read. This allows you to store more data in the file system at the cost of the processor load.
  - **JFS with large file support** - Available from AIX 4.3. This file system allows you to create files of approximately 64 GB (64 GB minus 124 MB). It was the first response to the need to create files larger than 2 GB. Another response was the creation of JFS2.
- **JFS2 (Enhanced Journaled File System)** – Improved version of JFS, used by default starting from AIX 5. The maximum file size you can create on it is 32 TB.

You can create a file system on a predefined logical volume or without an existing logical volume. In the latter case, when creating a file system, a logical volume will be created on one of the disks. It's a good idea to use the first version, since you have full control over the disks on which a logical volume will be created. So, you have control over where the file system will be created.

Adding a JFS2 file system

By default, all file systems, including those created automatically during the installation of the operating system, are created as JFS2 file systems. The process of creating a file system from a *smit crjfs2std* menu is shown in Figure 8-6.

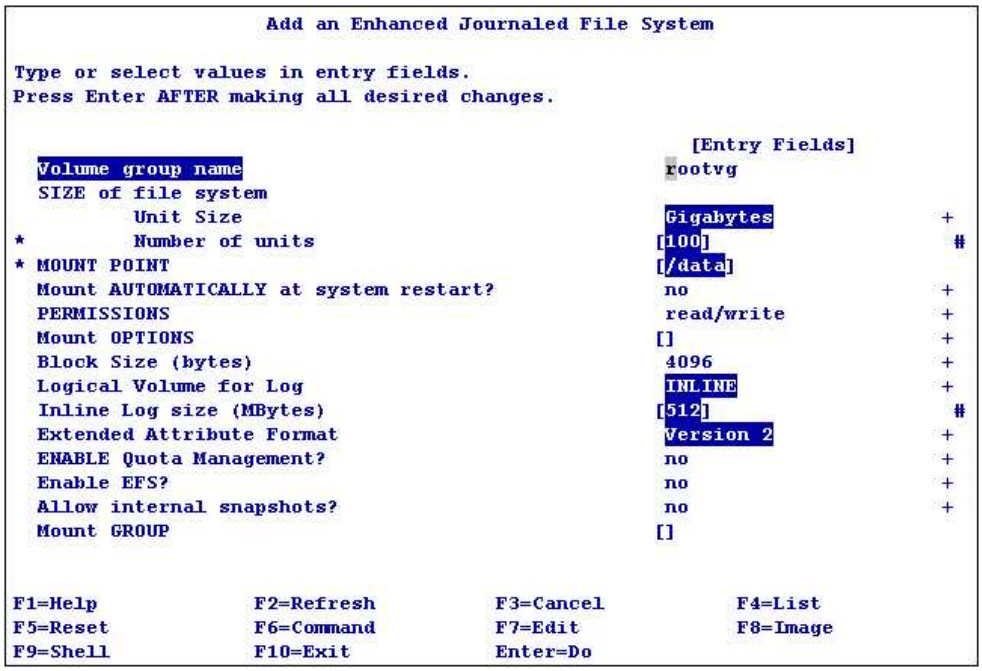

Figure 8-6: smit crjfs2std - Add an JFS2 file system.

Once the file system is created, the information is added to */etc/filesystems* in the form shown in the following output:

```
# grep -p "/data:" /etc/filesystems
/data:                                  # The directory that is the mount point.
        dev             = /dev/fslv00   # Automatically created logical volume.
        vfs             = jfs2
        log             = INLINE
        mount           = false   # No automatic mounting of the FS after the restart.
        options         = rw
        account         = false
```

Key parameters:

- **Size of file system** - Specifies the size of the file system, the *number of units* to allocate, and the *unit size*. You can specify the size of the file system with the accuracy of the block. In fact, it is allocated using whole physical partitions and, as a result, the size is rounded up to fill the whole partition. The above example adds a 100 GB file system.
- **Logical Volume for Log** - Determines whether a log should be part of a file system or whether it should be located on a dedicated logical volume. For a log to be a part of the file

system, select *INLINE*; otherwise, select an existing logical volume defined as a log. The *INLINE* log in the JFS2 file system is new compared to the usual JFS, which could only use a log created in a separate logical volume.

- **Inline Log size (Mbytes)** - The size of the inline log that will be created (if an inline log is selected). By default, the operating system creates a log of the size: *file_system_size / 256*. Usually, there is no need to modify it.
- **Extended Attribute Format** - This parameter first appeared in AIX 5.3. It specifies whether the file system should use additional attributes and support the NFS4 access control list that was introduced with the NFS version 4 protocol. There are two possible settings:
  - **Version 1** - The default setting that is compatible with AIX 5.2 and earlier versions. In this case, a standard access control list (AIXC - AIX Classic) known from earlier versions of the system is used. More information about this list is provided later in this chapter.
  - **Version 2** - The file system created will support the NFS4 list that is used with NFS version 4.
- **ENABLE Quota Management?** - Specifies whether you can set limits for individual users in the file system. This option appeared in JFS2 which was implemented in AIX 5.3.
- **Enable EFS?** - Enables the ability to encrypt files on your file system. For more information about encryption, see the security chapter. This parameter can be set dynamically during the lifetime of the file system.
- **Allow internal snapshots?** - Specifies how to create snapshots for the file system.
  - **yes** - Means to build snapshots within the file system.
  - **no** - Means to build snapshots for the file system in a separate, dedicated logical volume.

Alternatively, you can create a file system with the *crfs* command.

Adding a compressed file system

You can only create a compressed file system in the older version, that is, JFS. When creating it, keep in mind that there is no easy way to convert from a compressed file system to an uncompressed (or vice versa). The only way to do this is to create a new file system with the proper properties and then copy the data or restore it from the backup.

The following example demonstrates the creation of a file system on a previously prepared logical volume. This approach has a significant advantage - when you create a logical volume, you fully control the disks on which that logical volume will be created. This allows you to optimize the I/O performance; therefore, you should use this method.

You can simply create a compressed file system using the *smit crjfslvcomp* menu, as shown in Figure 8-7.

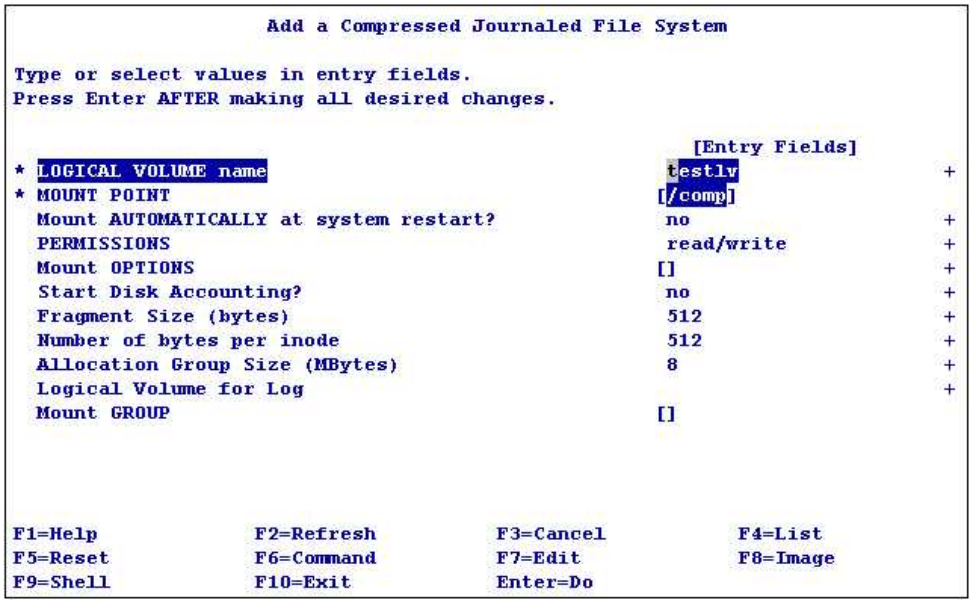

Figure 8-7: smit crjfscomp - Adding a compressed file system.

Key parameters:

- **LOGICAL VOLUME name** - Indicates the previously created logical volume on which the file system will be created. The file system will fill the entire logical volume.
- **PERMISSIONS** - Indicates the access rights to the file system that will be set by default:
  - **read/write**.
  - **read only**.
- **Mount OPTIONS** - The parameters that will be assigned by default to the *mount* command when mounting this file system.
- **Fragment Size (bytes)** - This parameter was described in the section concerning data blocks.
- **Number of bytes per inode** and **Allocation Group Size (MBytes)** - These parameters were described in the section concerning allocation groups.

Compressed file system properties

Compressed file systems use the LZ algorithm to compress data. In order to work properly, block fragmentation is required (the *fragment size* option set during file system creation must be less than 4096 B). Fragmentation is required due to the way compressed data is stored. This method involves inserting compressed data blocks into smaller areas (i.e., block fragments).

As mentioned earlier, the newer version of the file system (JFS2) does not support compression. The capacity gain and efficiency loss that you can expect when using a compressed file system are shown in the following examples.

We save the file containing the text data. The file is 500 MB in size, and it is copied to two identical file systems: one compressed, the other uncompressed. To determine the duration of the operation and the

amount of system resources consumed by the operation, we use the *time* command, which shows us:

- The real duration of the command;
- The amount of CPU time used by the user process that triggers the task; and
- The amount of CPU time used to complete the tasks generated by the running user process.

```
# time cp file.dbf /nocomp   # Copying a file to the file system without compression.

real    0m2.21s
user    0m0.00s
sys     0m0.87s

# time cp file.dbf /comp   # Copying a file to a compressed file system.

real    0m4.98s
user    0m0.00s
sys     0m0.97s
```

Copying a large, typical file to a compressed file system takes about twice as much time as copying it to an uncompressed file system. It should be noted that if more files of smaller sizes were copied, the result would be worse.

Interestingly, the *time* command shows a small difference in the processor load between the operation on the compressed file system and that on the non-compressed file system. In fact, the write to the compressed file system consumes much more system resources, and thus the processor time. The slight difference is due to the fact that the time needed to wait for the input and output operations is added to the system load. As we write uncompressed data, we perform a larger number of I/O operations, so the system performs these operations for longer. However, in the case of normal system operation, the time that this process waits for the input/output operations would be consumed by another process that does not require such operations. Thus, sys would shew a lower value.

To check the speed and use of system resources when reading previously prepared files, we will use the *dd* command. In our example, it reads the file block by block:

```
# time dd if=/nocomp/file.dbf of=/dev/null bs=4096
119541+1 records in.
119541+1 records out.

real    0m2.97s
user    0m0.16s
sys     0m0.86s

# time dd if=/comp/file.dbf of=/dev/null bs=4096
119541+1 records in.
119541+1 records out.

real    0m4.20s
user    0m0.15s
sys     0m1.41s
```

Reading a sample file from a compressed file system takes nearly twice as long, and it consumes more than twice as many CPU resources.

As shown in the following output, the gain on capacity is almost triple. This is not as fantastic a result as for a text file, but it is worth noting. The profit may be different for different types of files, and the

average result that can be achieved is the size of the occupied space lowered by half.

```
# du -sm /nocomp/file.dbf.
466.96  /nocomp/file.dbf   # Uncompressed file - Size is 466.96.

# du -sm /comp/file.dbf
167.98  /comp/file.dbf#   # Compressed file - Size after compression is 167.98.
```

As you can see in the examples above, the gain derived from compression is not very significant. In addition, it can only be used for the JFS, which has a lot of restrictions, including the maximum file size. However, in some cases, such as storing multiple text files, it is important to consider using it.

File system operations

Every file system that exists in AIX is listed by default in the */etc/filesystems* file. For example:

```
/testfs:
        dev             = /dev/fslv01 # LV on which the FS is created.
        vfs             = jfs2        # FS type.
        log             = /dev/hd8    # LV on which the FS log is located.
        mount           = false       # Should the FS be mounted during system startup.
        options         = rw          # Type of access to the FS.
        account         = false       # Accounting.
```

This file contains the same parameters that were given when creating the file system. You can obtain similar information using the *lsfs* command:

```
# lsfs
Name            Nodename   Mount Pt               VFS   Size   Options Auto Accounting
/dev/hd4        --         /                      jfs2  688128  --         yes  no
/dev/hd1        --         /home                  jfs2  32768   --         yes  no
/dev/hd2        --         /usr                   jfs2  5210112 --         yes  no
/dev/hd9var     --         /var                   jfs2  360448  --         yes  no
/dev/hd3        --         /tmp                   jfs2  196608  --         yes  no
/dev/hd11admin  --         /admin                 jfs2  262144  --         yes  no
/proc           --         /proc                  procfs --     --         yes  no
/dev/hd10opt    --         /opt                   jfs2  65536   --         yes  no
/dev/livedump   --         /var/adm/ras/livedump  jfs2  524288  --         yes  no
/dev/fslv00     --         /data                  jfs2  229376  rw         yes  no
/dev/testlv     --         /comp                  jfs   1605632 rw         yes  no
/dev/lv00       --         /nocomp                jfs   1277952 rw         yes  no
```

After you create a file system, you can change a lot of its parameters. You can perform most of the changes using the *SMIT* utility or the *chfs* command. The basic operations that can be performed on a file system are:

- Mounting and unmounting;
- Changing the size; and
- Defragmentation.

Mounting and unmounting

File systems can be automatically mounted at boot time. This occurs with all the important AIX file systems. If the file systems are not mounted at boot time, then you need to use the *mount* command to mount them to an existing directory:

```
# mount /testfs   # File system /testfs will be mounted as described in /etc/filesystems.
```

This is the simplest use of the mount command. You may wonder how AIX knows what file system to mount. The answer is simple. AIX knows that each file system is described in the */etc/filesystems* file, so it checks that file for the information it needs. When it finds an entry, it has all the data needed to mount it. Therefore, it is usually not necessary to specify other parameters in the *mount* command.

You can mount all the unmounted file systems, which are described in the */etc/filesystems* file, using one command:

```
# mount -a
```

You can also mount remote file systems using the NFS or SMB protocol. For details on how to do so, please consult the manual (*man mount*).

Unmounting the file system involves a similar process, except that the *umount* command is used:

```
# umount / testfs   # FS /testfs will be unmounted.
```

Resize

Both the JFS and JFS2 file systems can be extended "on the fly" without any break in access to them. However, you can only reduce JFS2 "on the fly," starting with AIX version 5.3.

The file system extension command is the same for both the JFS and JFS2.

Extending file system - You can do this on a working system using the *chfs* command, for example:

```
# chfs -a size=40000 /testfs   # Extension of the FS size up to 40,000 blocks of 512 B each.
# chfs -a size="+4M" /testfs   # Addition of 4 MB to the FS.
```

As you can see in the examples above, by default the file system size is increased by the number of disk blocks specified. You can specify the units in which the size of the area to be added is determined. For example, typing "M" immediately after the number stands for megabytes, while "G" stands for gigabytes.

When creating a file system, its size is rounded up to the size of the full physical partitions. The same occurs when the file system is extended. Even if you increase the file system by one disk block, the system rounds that size up to that of the entire physical partition. The physical partition may have a size ranging from 1 MB to 132 GB, so the rounding may be significant.

Reducing file system size - Up to AIX 5.2, the only way to reduce the file system was to remove the old one and create a new one with a smaller size. In AIX 5.3 and later, you can reduce the JFS2 file system "on the fly" in the same way as you can extend it, for example:

```
# chfs -a size = 40000 / testfs # Resizes the file system to 40,000 blocks of 512 B each.
# chfs -a size = "- 40M" / testfs # Reduces the file system by 40 MB.
```

The first example reduces the size of the file system to 40,000 disk blocks (if larger). The new size is rounded up so that it consists of the full physical partitions. The operation will succeed if the existing data can fit into the size specified in the command.

The second example reduces the file system size by 40 MB by rounding down to the full physical partitions. If the partitions were 16 MB, this would reduce the file system by two partitions (i.e., 32 megabytes).

When reducing the file system size, the operating system performs three steps:

1. Checks if there is sufficient space in the file system to release the logical partitions. If there is enough free space to release at least one, the system moves to step two.
2. Moves the data blocks from the logical partitions that are to be deleted to those that are to remain in the file system.
3. Removes the empty logical partitions.

Defragmentation

Defragmentation is most important for file systems (only JFS) built on fragments smaller than the block size. Compressed file systems are part of this group. If the file system has defined fragmentation, then the operating system always tries to save the ends of files in the fragments of blocks. The purpose of such a procedure is to ensure that the small (less than 4 KB) end of the file does not occupy the entire file system block. This allows you to save a few ends of files in one block, and the files hence occupy less disk space. This is shown in Figure 8-8.

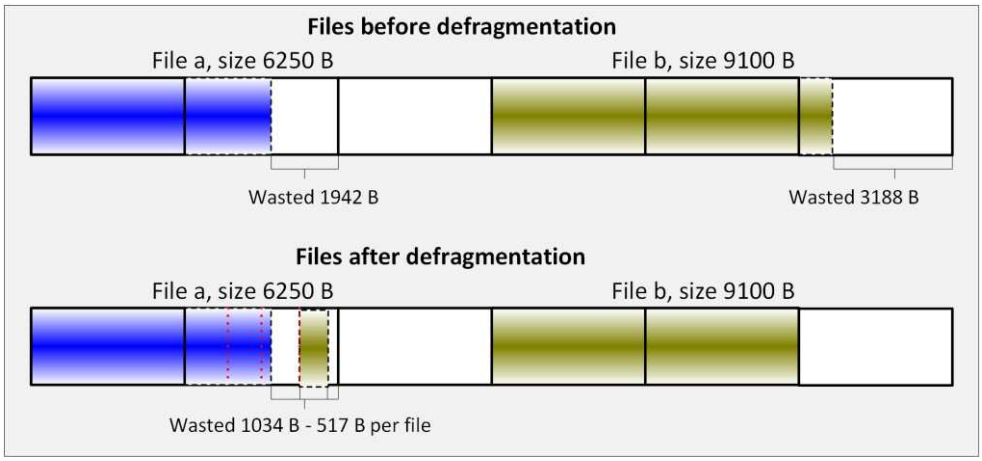

Figure 8-8: File system defragmentation.

As you can see in the figure above, there were five file system blocks occupied prior to defragmentation. Performing defragmentation has reduced the occupancy to four blocks.

Another benefit of placing several file ends in one block is the fact that it limits the number of I/O operations. The system always reads from disks with full blocks. If it wanted to operate on both files before defragmenting, it would have to read five file system blocks. However, when operating on files after defragmentation, it must read only four blocks.

The defragmentation process can also be run on file systems without fragmentation. However, the only thing that the system will do in such a case is to organize the data needed to obtain a continuous area of free blocks.

You should perform defragmentation on the mounted file system. You can do so during normal system operation, with the only side effect being the high disk load. You can use the *defragfs* command to perform defragmentation and evaluate the results you obtain. The defragmentation results are best explained using an example. In this case, an example file system is */frag*:

```
# df | grep -E "Filesystem|/frag"
Filesystem    512-blocks      Free %Used    Iused %Iused Mounted on
/dev/lv00         524288    339168   36%     3334     6% /frag
```

The */frag* file system occupies 524,288 512-byte disk blocks. It is a fragmented file system wherein the fragments have a size of 512 B. Before you start any defragmentation, you can see the results that you can achieve:

```
# defragfs -s /frag
/frag filesystem is 0 percent fragmented
Total number of fragments                : 524288
Number of fragments that can be migrated : 417
```

The above command provides us with some valuable information:

- The percentage of file system fragmentation.
- The total number of fragments - This is the total size of the file system in fragments. The size of the fragments in our case is 512 B.
- The number of fragments that can be migrated to other disk areas and merged so that they would occupy fewer allocation blocks.

If you require more information about the results that the *defragfs* command can provide, you can use the *-r* option:

```
# defragfs -r /frag
Statistics before running defragfs:
Number of free fragments                     : 298043
Number of allocated fragments                : 226245
Number of free spaces shorter than a block   : 197
Number of free fragments in short free spaces : 488

Statistics after running defragfs:
Number of free spaces shorter than a block   : 164
```

```
Number of free fragments in short free spaces : 416

Other statistics:
Number of fragments moved      : 417
Number of logical blocks moved : 120
Number of allocation attempts  : 104
Number of exact matches        : 25
```

The above operation can take a long time for large file systems, since it requires a detailed review of the file layout. As a result, information is divided into three sections that tell us:

- What the current level of fragmentation in the system is;
- What the result of defragmentation will be; and
- What operations will be performed at the time of execution.

The above example shows that there are 197 blocks (4 KB each) that are not fully filled. The free fragments number in these blocks are 488. After defragmentation, it is expected that there will be 164 blocks with free fragments, and the number of free fragments in those blocks will be 416. By arranging the fragments in the smallest possible number of blocks, you can expect some additional free space in the file system.

In the third section, you will see that 417 fragments and 120 allocation units (4 KB) will be moved, while 25 allocation units will be completely filled with the ends of files. If you know the effects of the *defragfs* command, you can decide whether to use it. If so, it is performed as follows:

```
# defragfs /frag
Statistics before running defragfs:
Number of free fragments                    : 298043
Number of allocated fragments               : 226245
Number of free spaces shorter than a block  : 197
Number of free fragments in short free spaces : 488

Statistics after running defragfs:
Number of free spaces shorter than a block  : 164
Number of free fragments in short free spaces : 416

Other statistics:
Number of fragments moved      : 417
Number of logical blocks moved : 120
Number of allocation attempts  : 104
Number of exact matches        : 25
```

The result is in line with expectations. It is important to know that in AIX, as in other UNIX systems, defragmentation is rarely used. By writing or modifying data, the operating system is constantly trying to optimize the records. Therefore, there will never be much fragmentation within file systems, and defragmentation will not significantly improve system performance.

The only way to perform full defragmentation is to remove all the files from the file system and then copy or restore them from the backup.

Other operations

Removal - Make sure that the file system is not mounted (for example, using the *df* command) before deleting.

```
# rmfs /testfs      # Removing the file system /testfs.
```

Calling this command:

- Deletes the file system;
- Deletes the logical volume on which it was located; and
- Deletes the file system entries from */etc/filesystems*.

Checking - If you are not sure whether the file system is working correctly, you can check its status using the *fsck* command. This command checks the file system and, depending on the parameters used, either fixes the errors automatically or asks the administrator for confirmation. To repair the file system, you must first unmount it. If you need to check the file system without fixing it, you can run the *fsck* command on the running file system:

```
# fsck /dev/fslv01   # Checks the filesystem located on the LV named fslv01.
```

Standard file system permissions

All file and directory permissions are defined in a standard UNIX manner. It is best to show them using the *ls -e* command. This command works in the same way as *ls -l*, except that it also tells you whether extended file attributes are set:

```
# ls -e smit.log
-rw-r--r--+    1 root     system        24753 Dec 14 12:11 smit.log
```

As a result of using this command, in the first column you can distinguish several sections:

-|rwx|rwx|rwx|+ - These sections show (starting from the left) the file type, owner permissions, group permissions, permissions of the rest of the users, and whether to use extended attributes (ACLs).

The first section, which is represented by a single character, specifies the file type. A number of file types are available:

- ***-*** - Usual file.
- ***d*** - Directory.
- ***l*** - Symbolic link.
- ***s*** - Socket file.
- ***c*** - Character device.
- ***b*** - Block device.
- ***p*** - FIFO file (first in, first out).

The next three sections specify the permissions for the file owner, group of users, and other users. The permissions have similar but not identical meanings for files and directories. This is shown in Table 8-2.

Table 8-2: Access permissions.

Permission	File	Directory
r	Read permission (required *x* for the directory that contains the file)	Permission to browse files in the directory
w	Write/delete permission (required *x* or *w* for the directory that contains the file)	Permission to create a file in the directory
		Permission to rename a file in the directory
		Permission to delete a file from the directory
x	Execute permission (required *x* for the directory that contains the file)	Required for reading, writing, and executing a file in the directory

There are three additional properties associated with file permissions, which are known as bits:

- SUID - The bit set for executable files. It is identified by the letter "s" in the owner permissions section. It indicates that the file (executable program) will run with the owner's permission, not the permission of the user who runs it. For example:

```
# ls -l /usr/bin/su
-r-sr-xr-x    1 root     security      40047 Mar 07 2016  /usr/bin/su
```

  The running of the */usr/bin/su* file by any user in the security group will cause the program to run with the owner privileges, which will be *root* in this case. SUID has no meaning for directories.

- SGID - The bit identified by the letter "s" in the group permissions section. It indicates that the file (executable program) will run with the group permissions of that file, not the permissions of the primary group of users that run it. For example:

```
# ls -l /usr/bin/timex
-r-xr-sr-x    1 bin      adm            8678 Jan 21 2016  /usr/bin/timex
```

  The running of the */usr/bin/timex* program by any user will cause the program to run with the *adm* group permissions, regardless of the primary group of the user who runs it.

  SGID set to the directory means that each file and directory created in it will inherit the group of the parent directory. Without the set SGID bit, the newly created files and directories are in the primary user group of the owner. This privilege is propagated down the directory structure.

- SVTX (Sticky bit) - The bit identified by the letter "t" in the permissions section of other users. The bit has no meaning for files. When placed on the directory, it means that the files located there can only be removed by their owners. After installing the operating system, only the */tmp* directory has the sticky bit set:

```
# ls -l / | grep tmp
drwxrwxrwt    7 bin      bin            4096 Dec 15 04:00 tmp
```

At this point, it is worth mentioning something that may not seem obvious. A user who does not have the "w" permission for the file can delete that file if he has the "w" permission for the directory where the file is located.

You can use three commands to manage file permissions:

```
# /usr/bin/chown    # Change the owner of the file.
# /usr/bin/chgrp    # Change the group.
# /usr/bin/chmod    # Change the file permissions.
```

You can use the *chmod* command using letters or numbers to change permissions. Both methods are shown in the outputs below as well as in Table 8-3

Table 8-3.

```
# /usr/bin/chmod o+r file1   # Letter notation - Adding read permission for all users.
# /usr/bin/chmod 0750 file1   # Number notation - Modifying the permissions according to the table.
```

Table 8-3: File permissions, numeric notation.

Special bits			Owner			Group			Others		
SUID	**SGID**	**SVTX**	**r**	**w**	**x**	**r**	**w**	**x**	**r**	**w**	**x**
4	2	1	4	2	1	4	2	1	4	2	1
`/usr/bin/chmod 0750 file1  # Change permissions as follows:`											
-	-	-	4	2	1	4	-	1	-	-	-
0			7			5			0		

Access Control Lists (ACLs)

In a traditional UNIX system, the only permissions for files and directories that can be granted are read, write, and execute. These permissions can be granted to the owner, group, or all users. They are often sufficient for basic applications.

The applications that are currently in use do not usually require users to access the command line. Information sharing is typically based on the rules that are defined by the application. However, traditional UNIX file and directory permissions are sometimes not sufficient. Access Control Lists are useful in such a case. In AIX, there are two types of ACLs available:

- AIXC (AIX Classic) - A traditional AIX ACL.
- NFS4 - ACL introduced in AIX 5.3 with Network File System version 4.

AIXC (AIX Classic)

AIXC has existed for a long time in AIX. It enables more granular operations on privileges than traditional UNIX mechanisms.

There are three commands for ACL operations:

- **aclget** - Read permissions for a file or directory.
- **acledit** - Edit permissions for a file or directory.
- **aclput** - Modifies permissions from the command line.

By default, ACL is disabled. It can be activated for individual files and directories. This is done by editing the list and replacing the word "disabled" with "enabled", for example:

```
# aclget /smit.log
*
* ACL_type   AIXC
*
attributes:
base permissions   # The base file permissions.
    owner(root):  rwx
    group(system):  r--
    others:  r--
extended permissions   # Below this row are the ACL permissions.
    disabled   # Disabled/enabled
```

It is important to note that if you want to use the *acledit* command, you need to have the *EDITOR* environment variable set. This variable should include the full path to the default text editor on the system.

By enabling ACL on a file, you are not disabling its basic permissions. You only extend or restrict those permissions (*deny* keyword). When you change the permissions of an ACL-enabled file, do not use the *chmod* command with a numeric notation (for example, *# chmod 750 /file1*), since this will disable the ACL for the file.

Extended permissions

There are three keywords that can be used to define extended permissions. After each of these words, permissions to read, write, or execute are written as one or more characters. For example, *rwx*. Next come the names of the users or groups that these permissions apply to, for example, u*: user_name*, *g: group_name*. The keywords are:

- ***permit*** - Allows user or group access to a file or directory.
- ***specify*** - Works like a *permit* but defines access more precisely. The nature of this precision will be explained in a specific example later in this chapter.
- ***deny*** - Denies access to a file or directory to specific users or groups of users. *Deny* has a higher priority than *permit* and *specify*, so if there are conflicting records about the permissions, the dominant record will be *deny*.

An example of a file that uses extended permissions:

```
# aclget /var/message
*
* ACL_type   AIXC
*
attributes:
base permissions
    owner(stud3):  rwx
    group(students):  r--
    others:  r--
extended permissions
    enabled
    permit   -w-     u:stud1,g:physics
    specify  -wx     u:stud2,g:physics
    deny     --x     g:students
```

A line starting with the word *permit* gives privileges to write to the file (*-w-*) to the user *stud1* if he belongs to the group *physics*.

A line starting with *specify* gives privileges to write and run the file (*-wx*) to *stud2* if he belongs to the group *physics*. In other words, if *stud2* is not in the *physics* group, he will not have those permissions. He will only have read permissions as defined in the *base permissions* section.

A line starting with *deny* removes permissions to run files from all the users that belong to the *students* group. Note that the file owner (i.e., *stud3*) will also not have permission to run this file. However, as the owner of the file, he will always be able to change the permissions.

Modification of permissions

You can make changes to extended permissions using the *acledit* command. This allows you to edit permissions in the default system text editor. The *EDITOR* environment variable indicates the text editor. After editing the permissions, save the file. Once the file has been saved, the system will ask whether to update the ACL when you exit the editor. The *acledit* tool is shown in Figure 8-9.

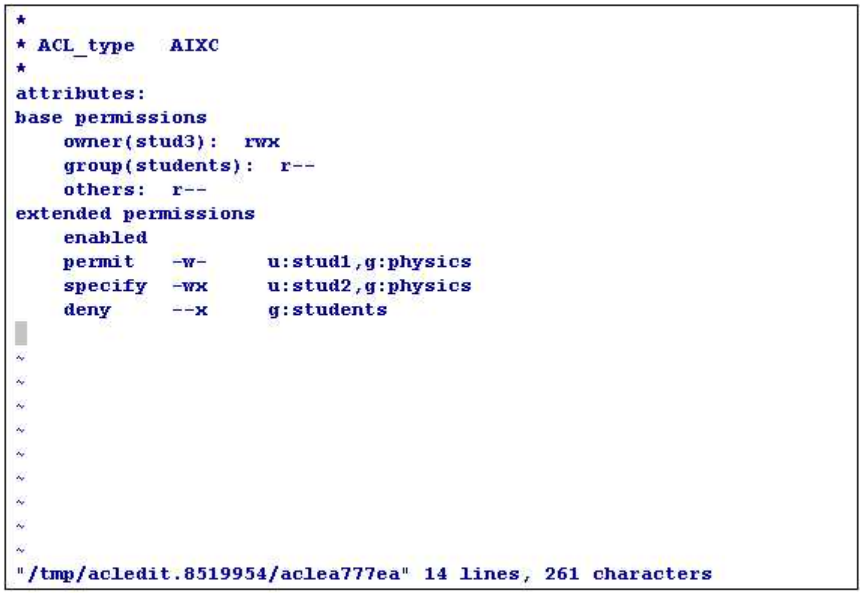

Figure 8-9: Editing the Access Control List - acledit.

Unfortunately, this means of granting permissions is quite laborious. You also need to remember how to write the permissions. In some cases, it is easier to use the *aclget* and *aclput* commands. Using a combination of the two, you can copy the permissions for one file to another. There are two methods of using the commands.

Copying permissions, first method:

```
# aclget message1   # Checking the permissions for the file message1.
*
* ACL_type   AIXC
*
attributes:
base permissions
    owner(root):  rwx
    group(system):  r--
    others:  r--
extended permissions
    disabled

# aclget message | aclput message1   # Copying the permissions from the message file to the message1
file.

# aclget message1   # Checking the copy result.
*
* ACL_type   AIXC
*
attributes:
base permissions
    owner(root):  rwx
    group(system):  r--
    others:  r--
extended permissions   # Extended permissions were copied.
    enabled
    permit   -w-     u:stud1,g:physics
    specify  -wx     u:stud2,g:physics
```

```
    deny     --x     g:students
```

Copying permissions, second method:

```
# aclget -o rights message   # Saving the permissions for the file "message" to the file "rights."

# cat rights   # Checking the resultant file - It can also be edited.
*
* ACL_type   AIXC
*
attributes:
base permissions
    owner(root):  rwx
    group(system):  r--
    others:  r--
extended permissions
    enabled
    permit   -w-     u:stud1,g:physics
    specify  -wx     u:stud2,g:physics
    deny     --x     g:students

# aclput -i rights message1   # Assigning permissions to the file message1.
```

If you add the -R parameter and specify a directory, you give the above permissions in a recursive manner to the directory and its contents, for example:

```
# aclput -i rights -R messages   # Assigning permissions to all the files and subdirectories in the
"messages" directory.
```

NFS4

In AIX 5.3, along with Network File System version 4, the NFS4 Access Control List has been introduced. Although this list was created for managing the access permissions for network resources, AIX can use it locally. This ACL is very similar to that of the NTFS file system that is used on Windows, and it offers more possibilities to define permissions than AIXC.

To use this list, whether locally or remotely, the file system must support it. AIX only supports it in the JFS2 file system if it was created with the *Extended Attribute Format* set to *Version 2*. If not, you can change the *Extended Attribute Format* parameter dynamically during the file system operation (*smit chfs* menu).

AIXC is the default access control list for all files. If you used the AIXC access control list and want to change it to the NFS4 list, then you need to convert it. You can do this using the *aclconvert* command (the *-R* parameter allows the recursive conversion of the directory). This command can convert lists in both directions.

The following example shows the AIXC and NFS4 lists as well as how to convert from AIXC to NFS4:

```
# aclget test   # Access Control List (AIXC) of a sample file
*
* ACL_type   AIXC
*
attributes:
base permissions
    owner(root):  rw-
    group(system):  r--
    others:  r--
extended permissions
    disabled

# aclconvert -t NFS4 test   # Conversion of the above list to NFS4.

# aclget test   # The same list as above when converted to NFS4.
*
* ACL_type   NFS4
*
*
* Owner: root
* Group: system
*
s:(OWNER@):     a       rwpRWaAdcCs
s:(OWNER@):     d       xo
s:(GROUP@):     a       rRadcs
s:(GROUP@):     d       wpWxACo
s:(EVERYONE@):  a       rRadcs
s:(EVERYONE@):  d       wpWxACo
```

On both AIXC and NFS4 access control lists, you can use the same commands, namely *acledit*, *aclget*, and *aclput*. The means of using these commands was described during the presentation of the AIXC list.

Format of access permissions

The textual representation of the list contains a series of lines with the following structure (ACE = Access Control Entries):

IDENTITY ACE_TYPE ACE_MASK ACE_FLAGS

1. IDENTITY

The IDENTITY format is: IDENTITY_type:(IDENTITY_name or IDENTITY_ID or IDENTITY_who)

The meanings of the individual fields are explained in Table 8-4.

Table 8-4: NFS4 ACL - Values that the IDENTITY_type field can take.

IDENTITY_type	Description
u : user	In this case, IDENTITY_name is the username, while the IDENTITY_ID is the ID of the user.
g : group	In this case, IDENTITY_name is the name of the group, while the IDENTITY_ID is the group ID.
s : special who string	In this case, IDENTITY_who is one of three possible special values: OWNER@ - file owner; GROUP@ - the group to which the file belongs; or EVERYONE@ - everyone.

Examples:

s:(OWNER@): - Indicates that the row represents the permissions for a file of its owner.

s:(GROUP@): - Indicates that the row represents the permissions for a file of its group.

2. ACE_TYPE

A single letter represents the ACE_TYPE:

a : allow
d : deny
l : alarm
u : audit

3. ACE_MASK

The ACE_MASK consists of single letters, each of which represents a specific permission. The individual permissions are detailed in Table 8-5.

Table 8-5: NFS4 ACL - Possible values of the ACE_MASK.

ACE_MASK	Description
r	The right to read a file or list the contents of a directory.
w	The right to modify the contents of a file or add a new file to a directory.
p	The right to add data to a file or create a new subdirectory.
R	The right to read the attributes of a file or directory.
W	The right to change the attributes of a file or directory.
x	File execution right.
D	The right to delete files or subdirectories of a given directory.
a	The right to read the base attributes (not related to the access control list) of a file or directory.
A	The right to change the base attributes (not related to the access control list) of a file or directory.
d	The right to delete a file or directory.
c	The right to read the access control list of a file or directory.
C	The right to modify the access control list of a file or directory.
o	The right to change the ownership of a file or directory.
s	File access permissions locally on the server (read and write synchronization).

4. ACE_FLAGS

The ACE_FLAGS specify the inheritance of the access control list. They are represented by two-letter parameters, as shown in Table 8-6.

Table 8-6: NFS4 ACL - Possible values of the ACE_FLAGS.

ACE_FLAGS	Description
fi	Indicates that this entry applies to any newly created file (not a directory).
di	Indicates that this entry applies to any newly created subdirectory.
oi	Indicates that this entry does not apply to the current directory. However, it applies to newly created files or directories (as indicated above).
ni	Indicates that this entry applies to newly created files or subdirectories in the directory. However, subdirectories will not pass this entry to their "descendants."

For a complete overview, consider the sample NFS4 ACL entry:

```
s:(OWNER@):   a       rwpRWaAdcCs    fidi
```

- s: (OWNER @): - The entry refers to the file owner's permissions.
- a - Grants permissions ("d" would indicate revoking permissions).
- rwpRWaAdcCs - Individual permissions:
  - r - The right to read a file or list the contents of a directory.
  - w - The right to modify the contents of a file or add a new file to a directory.
  - And so on, according to the table that shows the possible ACE_MASK values.
- fidi - Indicates that this entry applies to any newly created files and subdirectories.

Additional information

For the NFS4 ACL, there are two types of users with special file permissions:

- User with an ID of 0 (i.e., *root*). This user has all file permissions by default, regardless of the entries in the ACL. The only exception is the right to execute the file.
- File owner. He always has the following permissions for his files, regardless of the ACL:
  - c (ACL reading);
  - C (ACL change);
  - a (reading base file permissions); and
  - A (changing base file permissions).

If you are using NFS4 ACL, do not change the base permissions using the *chmod* command. This action disables the access control list and, as a result, you lose information about extended permissions. The only case in which you can use the *chmod* command is to set the bits: *SUID* (*chmod u + s* or *u-s*), *SGID* (*chmod g + s* or *g-s*), or *SVTX* (*chmod + t* or *-t*).

File system snapshots

Starting with AIX 5.2, you can create a file system snapshot. This is a mechanism that was previously used in large disk arrays. It has been adapted for use in AIX primarily in order to increase its backup capabilities.

It only works with the JFS2 file system. It allows operations (read only) on files and directories that are in the same state as they were in at the time of snapshot creation, regardless of the operations that were later performed on them.

By creating a regular file system backup using any standard AIX tool, we can copy files in a specific order to the selected location. This may be, for example, a file, another file system, or a tape. If the file system is large, it may take up to several hours for the backup to run. As a result, the first file will be saved a few hours earlier than the last file, which may cause the data in the backup to be inconsistent. This occurs when these two files were in some kind of relationship.

The solution to this problem is to create a file system snapshot that will allow access to files from exactly one point in time. The backup should be done on this snapshot.

In AIX 7.x, you can perform a variety of operations on the file system snapshot, which are shown in Figure 8-10.

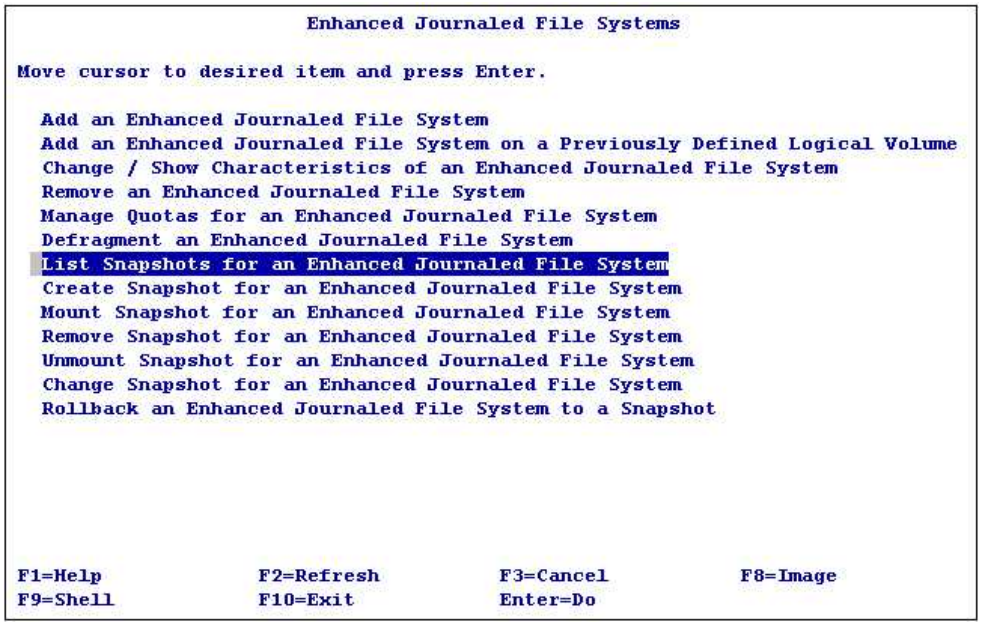

Figure 8-10: Smit - Operations on the file system snapshot.

The meaning of each snapshot option:

- **List Snapshots for an Enhanced Journaled File System** - Displays the snapshot information of a file system. It provides information on the time the snapshot was created, the logical volume on which it resides, and the size and usage of that volume. This is equivalent to the *snapshot* command:

```
# /usr/sbin/snapshot -q /usr
Snapshots for /usr
Current  Location          512-blocks        Free Time
*        /dev/fslv01           229376      227328 Thu Dec 15 05:53:00 CST 2016
```

- **Create Snapshot for an Enhanced Journaled File System** - Creates a file system snapshot. More information about this option is given in the next section.
- **Mount Snapshot for an Enhanced Journaled File System** - This is traditionally done using the *mount* command with the *-o snapshot* parameter. Data access is only possible after the snapshot has been mounted. It is always mounted in read-only mode.
- **Remove Snapshot for an Enhanced Journaled File System** - The equivalent of the *snapshot -d /dev/logical_volume* command.
- **Unmount Snapshot for an Enhanced Journaled File System** - The equivalent of the *umount /filesystem* command.
- **Change Snapshot for an Enhanced Journaled File System** - Allows you to change the size of the snapshot created in a separate logical volume. The equivalent of the *snapshot -o size = "snapshot_size" /dev/logical_volume* command.
- **Rollback an Enhanced Journaled File System to a Snapshot** - Restores a file system state from a previously created snapshot. The file system must be unmounted in order to perform this operation.

The only operation that is worth further discussion is the creation of a file system snapshot. A description of this operation is hence provided in the following section.

Creating a snapshot

You can create a file system snapshot in the file system space if you created the FS with the *Allow internal snapshots* option. If it was created without this option, the only way forward is to create a snapshot in a separate logical volume. It can be created on a pre-prepared logical volume, or a logical volume can be created automatically while a snapshot is created. In the latter case, a logical volume with the default name is created, for example, *fslv00*. In such a case, you do not have any influence over the placement of this volume. The process of creating a snapshot using the SMIT menu is shown in Figure 8-11.

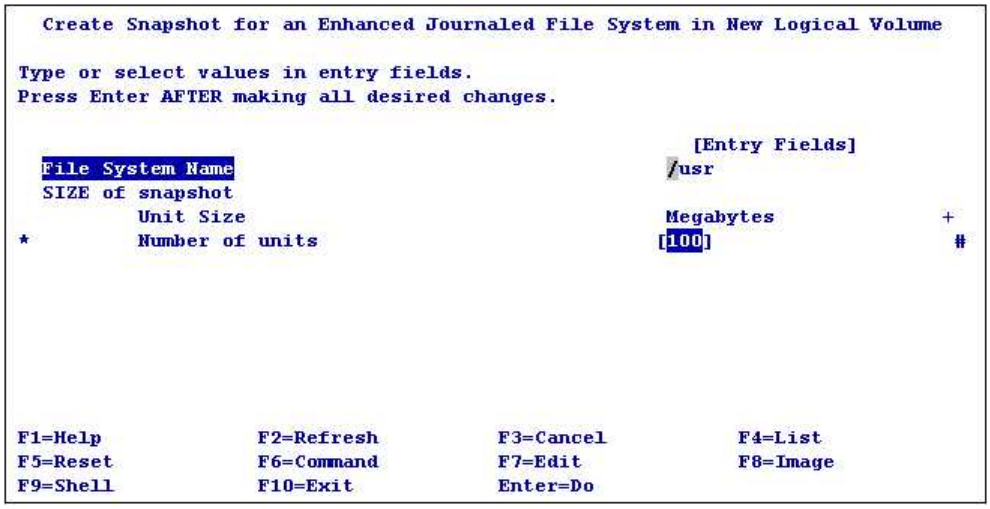

Figure 8-11: Smit - Create a file system snapshot.

There are only two parameters that you need to enter here:

- **File System Name** - The name of the file system to which the snapshot is to be created.
- **Number of units** - The size of the snapshot in selected units (512-byte blocks, megabytes, gigabytes). Due to the way the snapshot works, for typical file systems it is sufficient to give it a size some 2–6% of the size of the source file system. However, this is highly dependent on the number of changes made to this file system as well as how long the image will remain.

An alternative way to create a snapshot is to use the command line:

```
# snapshot -o snapfrom=/usr  -o size=100M.
```

The process of creating a snapshot

At the start of the operation for creating a snapshot, in order to ensure consistency, write operations to the source file system are temporarily paused. On the logical volume on which the snapshot is to be created, support structures are created. The resulting structures are responsible for mapping information from the source file system.

In this case, mapping indicates that if a given block of the source file system has not been modified, it will not be stored in the snapshot. There will only be an indication to the same block in the source file system.

If a block is read from a snapshot, it will be retrieved from the source file system. If you change the contents of that block in the source file system, it will be copied to the snapshot. Using this mechanism, the creation of a snapshot takes only a few seconds depending on the size of the file system, and the logical volume for a snapshot can be much smaller than the source file system.

Yet, maintaining a file system snapshot results in a greater load on I/O operations when modifying disk blocks in the source file system. The blocks, which are to be modified in the source file system for the first time, must be copied to the snapshot.

To more clearly illustrate how snapshots work, we will use an example. It is based on the */usr* file system:

```
# df | grep -E "Filesystem|/usr"
Filesystem    512-blocks      Free %Used    Iused %Iused Mounted on
/dev/hd2         1048576    137216   87%    11688    41% /usr
```

To demonstrate the consumption of time and resources that occurs during snapshot creation, we use the *time* command. This command was previously used in the compression section.

```
# time snapshot -o snapfrom=/usr  -o size=10M
Snapshot for file system /usr created on /dev/fslv01

real    0m0.80s
user    0m0.22s
sys     0m0.10s
```

As you can see, creating a snapshot of this file system took 0.8 seconds. The */dev/fslv01* logical volume was created, and it was used for a snapshot. Although it cannot be seen in the above display, the size of the logical volume is 64 MB rather than the declared 10 MB. This is because the smallest unit that can be allocated to a logical volume is the physical partition, which in this case is 64 MB in size.

During the creation of the snapshot, no write was made to the */usr* file system. All the scheduled writings were waiting for the snapshot creation process to finish.

The most important part of a snapshot is its map, which is stored in the superblock. The function of the map is to store the state of all the mapped file system blocks. When creating a snapshot, no blocks are copied from the source file system. The only task is to create a map that will contain the address of the source file system blocks. The map stores the following information:

- Addresses of the blocks of the source file system that were in use at the time of snapshot creation (containing some data).
- Addresses of the blocks of the source file system that were in use at the time of snapshot creation (containing some data), which have been modified or deleted after the snapshot was created.
- Addresses of the blocks copied from the source file system to the snapshot. They contain copies of those blocks of the source file system that have been modified or removed after the snapshot was created.

To take a closer look at the image you created, you need to mount it. After doing so, the image will be available in read-only mode:

```
# mount -v jfs2 -o snapshot /dev/fslv01 /snap

# df -k | grep -E "Filesystem|/usr|/snap"
Filesystem    1024-blocks      Free %Used    Iused %Iused Mounted on
/dev/hd2           524288     73216   87%    11688    40% /usr
/dev/fslv01         65536     65152    1%        -     -  /snap
```

Immediately after creation, the snapshot of the 500-megabyte file system, which is used at 87%, takes only 384 KB. These few hundred kilobytes represent a place for a snapshot map. This is because, until now, no source system block has been modified or deleted. After removing the */usr/snap.bid* file, which has a size of about 10 MB, from the source file system, the situation will change radically.

```
# rm /usr/tmp.bid

# df -k | grep -E "Filesystem|/usr|/snap"
Filesystem    1024-blocks      Free %Used    Iused %Iused Mounted on
/dev/hd2           524288     83680   85%    11687    37% /usr
/dev/fslv01         65536     54272   18%        -     -  /snap
```

The snapshot increased by more than 10 MB, that is, by the size of the deleted file. In order to maintain the consistent state of the snapshot, before deleting the file, the system copied all the blocks of the file from the source file system to the snapshot. This activity took much longer than the process of deleting the file. However, the snapshot still contains data from the time of its creation, regardless of the modifications that have occurred in the source file system.

It is important to note that creating files on the source file system does not involve any operations on its snapshot, since this does not interfere with the source file system state from the time of snapshot creation.

If a snapshot is only created in order to create a backup of the file system, you can use a much simpler method. This method is to use the *backsnap* command, which creates a snapshot of the file system, makes backups of it using the *backup* command, and then removes the snapshot. You can see how the command works in the following output:

```
# backsnap -R -m /snap -s size=64M -i -f/tmp/backup /var
Snapshot for file system /var created on /dev/fslv01
Mount volume 1 on /tmp/backup.
        Press Enter to continue.
rmlv: Logical volume fslv01 is removed.
```

Used parameters of the *backsnap* command:

- ***-R*** - Deletes the created snapshot after making a copy.
- ***-m /snap*** - Indicates the directory */snap* as the location where the snapshot is to be mounted.
- ***-s size=64M*** - Specifies the size of the snapshot.
- ***-i*** - This is an option passed to the *backup* command that runs on the snapshot after it is created. It instructs the *backup* command to retrieve the file names to back up from the standard input.
- ***-f /tmp/backup*** - Indicates the file where the backup is to be stored. You can specify a tape drive that is usually represented by */dev/rmt0*.

Chapter 9. The Paging Space

The paging space is associated with the concept of virtual memory, although this concept actually has a broader meaning. Instead of the term "paging space," the term "swap space" is often used. Its equivalent exists in almost every modern operating system.

The paging space is an area (usually a disk) that the system can use as an extension of its RAM. It allows you to run programs and create structures whose size exceeds the available RAM size. If you try to do this without having a paging space, the actions taken will at best fail and, at worst, lead to the unstable operation of the system.

This space should be adapted to the requirements of the applications working within the operating system. Application providers often provide the environment specification along with recommended settings, which also include information about the required paging space. If you do not have the specific information, you can use the generic rules for setting the size of the paging space. You can determine the size of the required space by the amount of RAM in the system. With a low memory size, up to 8 GB, it is best to create a space that is twice as large as the available RAM. With more memory, the space is usually created to match the size of the physical memory. With large amounts of memory, you can create a paging space that is half the size of the physical memory or less.

The above rules are very general. If you create too large a paging space, aside from wasting disk space, it will not negatively affect the system. If you create a small space, it will not negatively affect the system so long as you control the memory usage. In terms of general rules, it is best to configure the system so that the paging space is not used at all, or at least used as little as possible. Using this space may slow down the system and its applications. However, it must exist so that any unexpected demand for additional memory can be met.

How it works

The operating system works on the memory as well as in the paging space using a fixed size unit. These units are known as pages, and they are either 4 KB or 64 KB in size. The operating system can also operate on 16 MB or 16 GB memory pages, although it requires additional configuration, while 4 KB and 64 KB pages are managed automatically. The fixed page size facilitates the exchange of information between the RAM and the paging space. For the AIX operating system, there are two types of memory pages:

- **Computational pages** - All pages that process data and are thus subject to swapping. This means that if a page must free up RAM due to an insufficient number of free pages, it will be saved in the paging space. In a great simplification, you can see such pages as all memory pages (data, stack, shared memory, etc.), except for pages representing the file system cache.
- **Non-computational pages** - All memory pages not directly related to the processing of information and therefore not subject to swapping. In a great simplification, you can see them as memory pages representing the file system cache.

When the number of free RAM pages decreases below the limit it should not exceed (as defined by the

minfree parameter of the *vmo* command), then the process of releasing some of the occupied pages takes place. The operating system now has a choice: free up the memory occupied by cached files or by data. The type of pages to be freed is determined by the *lru_file_repage* parameters (not available starting from AIX 7.1) as well as the *minperm* and *maxperm* parameters of the *vmo* command.

lru_file_repage - This parameter is not available from AIX 7.1. In earlier versions of AIX, the operating system decided what kind of memory pages to release as additional needs arise. Possible settings:

- 1 - The operating system releases computational and non-computational pages without special preferences, depending on the *minperm* and *maxperm* parameters:
  - If the number of non-computational pages is greater than or equal to the *maxperm*, non-computational pages are released.
  - If the number of non-computational pages is between the *minperm* and the *maxperm*, both computational and non-computational pages are released.
  - If the number of non-computational pages is less than or equal to the *minperm*, computational pages are released.
- **0** - The operating system releases pages according to the *minperm* and *maxperm* parameters, although it focuses on releasing non-computational pages.
  - If the number of non-computational pages is greater than or equal to the *maxperm*, non-computational pages are released.
  - If the number of non-computational pages is between the *minperm* and the *maxperm*, non-computational pages are released. Occasionally, the release of computational pages may occur.
  - If the number of non-computational pages is less than or equal to the *minperm*, the pages of both types are released.

Starting with AIX 7.1, the system works as described for *lru_file_repage = 0*.

Releasing pages occupied by the file cache (non-computational pages) is the simplest and most resource efficient (I/O, CPU time) option. If the file page has not been modified after loading into the memory, it is simply marked as free as it is released, and no further action is required. The system can do this without losing information, since the file is permanently on the disk and the system only removes the volatile representation of the file page. If the file page was modified after loading into the memory, then it is written to the disk prior to release. This does not add an extra overhead to the operating system, since sooner or later such a page would be written to the disk. Only after this action is performed the page is marked as free.

Releasing the memory pages occupied by data (computational pages) is more laborious for the system. Data pages have no representation on the disk. To release such a page from the RAM, you must write it to a paging space. After this operation, and before any operation on this page, it must be moved back to the RAM.

The operating system releases the least used pages. Nevertheless, if you use a large paging space, you may experience trashing. This means continually rewriting pages from the RAM to the paging space and from the paging space to the RAM. This situation drastically slows down the system, and it can lead to a total blockage. Therefore, you should control the consumption of memory and not allow excessive use of the paging space.

AIX can work with multiple paging spaces. This is particularly important when you have a small amount of RAM and are hence forced to use the paging space as a memory extension. In this case, the system balances the load between all the paging spaces. If you place the paging space on multiple disks, you can accelerate this type of operation. The situation of writes to the paging space when there are several such spaces available is shown in Figure 9-1.

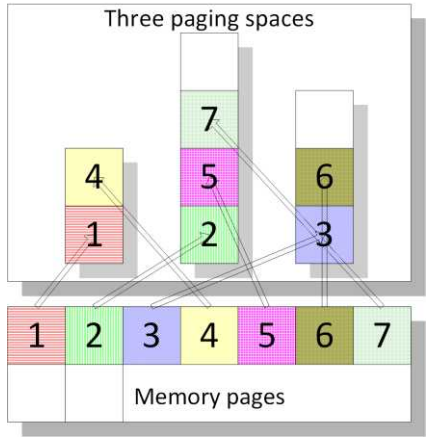

Figure 9-1: Writing pages of memory to several paging spaces.

The system, for as long as it can, writes the same number of pages in each space. If one of the spaces runs out of space, it writes the pages to the others. When deactivating a single space, the data are rewritten to the other spaces. This can take the system a long time, and the duration of this operation depends on the number of blocks to be rewritten.

Checking the paging space

In AIX, at least one paging space must both exist and be active. After installing the system, there is always a *hd6* space.

```
# lsps -a   # Checking the paging spaces.
Page Space Physical Volume   Volume Group    Size %Used   Active   Auto  Type   Chksum
hd6          hdisk0            rootvg         512MB     1   yes    yes    lv       0
```

The above output indicates that the *hd6* is located on the hdisk0 that belongs to the rootvg volume group. Its size is 512 MB and 1% of its area is currently being used. The space is active, and it is activated by default during system startup. The type column is the type of paging space. There are two possible types:

- **lv** - This means that the paging space is located on a logical volume. This type is commonly used, since the configuration is most efficient.
- **nfs** - This means that the paging space is a file shared by the Network File System. Typically, this method is not used due to the lower speed of this type of space and the likelihood of losing connectivity with the server, which renders the file available. This setting was applicable for AIX-based diskless workstations.

The *Chksum* column determines whether we use checksums for a given paging space, which can improve its reliability.

An important aspect concerning the paging space is the issue of monitoring. The consumption of the entire paging space, or its excessive use, negatively affects the operation of the system. You can easily check the capacity and usage of the paging space using any of the monitoring tools, for example, the *lsps* discussed earlier. Another important aspect is performance. If you notice a decrease in the performance of an application on your operating system, one of the main parameters to check is the use of the paging space. You can easily obtain this information using the *vmstat* command:

```
# vmstat 1   # Continuous monitoring with a 1 s interval.

System configuration: lcpu=8 mem=2048MB ent=2.00

kthr    memory              page              faults              cpu
----- ----------- ------------------------ ------------ -----------------------
 r  b   avm   fre   re   pi  po  fr   sr  cy  in   sy  cs   us sy id  wa    pc    ec
 2  0 252303  8894   0    0   0   0    0   0   10   480 182  0  0  99  0   0.01   0.4
 2  0 252303  8894   0    0   0   0    0   0   12   33  175  0  0  99  0   0.00   0.1
```

In this case, you should pay particular attention to the page section as well as the *pi* (page in) and *po* (page out) parameters. They show the number of writes and reads of memory pages from the paging space. If there are values constantly showing for these parameters, it usually indicates performance problems. This suggests that the reason for the slowdown is the lack of sufficient memory for the current system work.

It is worth noting that the high occupancy of the paging space does not necessarily cause performance problems, so long as the pages are not exchanged between it and the RAM. However, the constant use of large amounts of paging space without any work with these pages may suggest leaks of application memory. It may also suggest that unnecessary processes have been started and, further, that their (not used) memory is being written to the paging space.

Adding paging space

After installing AIX, the paging space is located on the logical volume */dev/hd6*:

```
# lsvg -l rootvg | grep -E "TYPE|paging"
LV NAME             TYPE       LPs     PPs     PVs  LV STATE      MOUNT POINT
hd6                 paging     32      32      1    open/syncd    N/A
```

There may be any number of paging spaces. When a new space is added, the system distributes the load evenly across all the available spaces using the round robin algorithm. To add a new space, use the *mkps* command or *smit mkps*. Creating a paging space of four physical partitions, which is activated immediately after creation and always during system startup, looks like this:

```
# mkps -a -n -s 4 rootvg hdisk0
paging00   # Added a space with a standard name.
```

Key parameters of the *mkps* command:

- ***-a*** - Activate the space every time the system is started.
- ***-n*** - Immediate activation.
- ***-s 4*** - Determine the size of the space (four logical partitions).

After the command is run, a new 64 MB paging space (four logical partitions of 16 MB) appears:

```
# lsps -a   # Checking the paging spaces.
Page Space Physical Volume   Volume Group    Size  %Used   Active  Auto Type Chksum
paging00        hdisk0         rootvg         64MB    0      yes     yes  lv     0
hd6             hdisk0         rootvg        512MB    1      yes     yes  lv     0
```

Removing paging space

At least one paging space must always exist and be active within the operating system. Therefore, you can only remove spaces when you have several available. Prior to removing a space, you must first deactivate it using the *swapoff* command; otherwise, the remove operation will fail. Removing the paging space occurs as follows:

```
# lsps -a   # Checking the paging spaces.
Page Space Physical Volume   Volume Group    Size %Used   Active  Auto Type Chksum
paging00        hdisk0         rootvg         64MB    0     yes     yes  lv     0
hd6             hdisk0         rootvg        512MB    1     yes     yes  lv     0

# swapoff /dev/paging00   # Deactivation of the paging space.

# rmps paging00   # Removing the paging space paging00.
rmlv: Logical volume paging00 is removed.

# lsps -a
Page Space Physical Volume   Volume Group    Size %Used   Active  Auto Type Chksum
hd6             hdisk0         rootvg        512MB    1     yes     yes  lv     0
```

The above example shows how to remove one space if you have two or more of them available. If there is only one space within the system, it cannot be removed or deactivated.

```
# lsps -a
Page Space Physical Volume   Volume Group    Size %Used   Active  Auto Type Chksum
hd6             hdisk0         rootvg        512MB    1     yes     yes  lv     0

# swapoff /dev/hd6   # Unsuccessful attempt to deactivate.
0517-081 swapoff: Cannot deactivate paging space /dev/hd6.
: There is not enough space in the file system.
```

Other operations on the paging space

The paging space can be manipulated using many commands, including *lsps* (list paging space), *swapon*, *swapoff*, *mkps* (make paging space), *rmps* (remove paging space), and *chps* (change paging space). In addition to the operations mentioned in the previous section, we can use them for:

- Activation and deactivation;
- Increasing the size; and
- Decreasing the size.

Activation and deactivation

These operations can be performed at any time, provided that one paging space is always active. The deactivation of the space is necessary before it can be removed. If you do not do so, then the attempt to remove the space will fail.

Examples of activation and deactivation:

```
# lsps -a   # Checking the paging spaces.
Page Space Physical Volume   Volume Group    Size %Used   Active  Auto Type Chksum
paging00        hdisk0         rootvg        64MB    0     yes     yes  lv     0
hd6             hdisk0         rootvg       512MB    1     yes     yes  lv     0

# swapoff /dev/paging00   # Deactivation of the paging space.

# lsps -a
Page Space Physical Volume   Volume Group    Size %Used   Active  Auto Type Chksum
paging00        hdisk0         rootvg        64MB    0     no      yes  lv     0
hd6             hdisk0         rootvg       512MB    1     yes     yes  lv     0

# swapon /dev/paging00   # Activation of the paging space.

# lsps -a
Page Space Physical Volume   Volume Group    Size %Used   Active  Auto Type Chksum
paging00        hdisk0         rootvg        64MB    0     yes     yes  lv     0
hd6             hdisk0         rootvg       512MB    1     yes     yes  lv     0
```

As you can see in the output above, both spaces are initially active. You can identify this fact by the word *yes* in the *Active* column. Then, the *paging00* space is deactivated using the *swapoff paging00* command. If the system stores a large amount of data in the paging space, the operation may take some time. The reason for this is that pages written in *paging00* must be rewritten to *hd6* before *paging00* is deactivated. The subsequent listing (*lsps*) of the paging spaces shows that *paging00* is not active. Activation performed using they *swapon* command is always a quick process.

You can specify whether the space is to be activated when the system is started. You can check this setting by verifying the *Auto* column using the *lsps -a* command. To automatically activate the *paging00* space during system startup, run the following command:

```
# chps -a y paging00
```

Otherwise:

```
# chps -a n paging00
```

Increasing the size

You can increase the size of the space at any time. You can do so on an active and loaded paging space using the *chps* command with the *-s* parameter:

```
# lsps -a
Page Space Physical Volume   Volume Group    Size  %Used   Active  Auto Type Chksum
hd6             hdisk0         rootvg       512MB    1     yes      yes  lv     0

# chps -s 6 hd6   # Adding six logical partitions to the hd6 paging space.

# lsps -a
Page Space Physical Volume   Volume Group    Size  %Used   Active  Auto Type Chksum
hd6             hdisk0         rootvg       608MB    1     yes      yes  lv     0
```

Six physical partitions were added to the *hd6* paging space in the above example. Each of the physical partitions has a size of 16 MB.

The paging space is usually a logical volume (lv type). Therefore, it can be extended using the standard *extendlv* command, which increases the size of the logical volume.

```
# extendlv hd6 6   # Adding six physical partitions to the hd6 logical volume.
```

However, this way of changing the size of the space is not recommended.

Shrinking

Starting with AIX 5.1, you can reduce the size of the paging space using one command: *chps -d*. You can do this even if you only have one space. Executing this command starts a series of steps that were performed manually in earlier versions of AIX:

- Creates a temporary paging space known as *paging00*. If this already exists, it creates a paging space with another unused number (*mkps* command).
- Changes the device on which the system should drop its state in the event of a crash (dump) to a temporary paging space (*sysdumpdev* command). This is only the case when reducing the size of the space intended to drop the system state on it. You can check this using the *sysdumpdev* command. In newer versions of the system, the dump space is created as an additional logical volume in the rootvg group.
- Removes the space to be reduced (*rmps* command).
- Creates this space again, assigning it a smaller size (*mkps* command).

The above steps are illustrated in the following example:

```
# chps -d 2 hd6   # Reducing the hd6 paging space by two logical partitions.
shrinkps: Temporary paging space paging01 created.
shrinkps: Paging space hd6 removed.
shrinkps: Paging space hd6 recreated with new size.
shrinkps: Resized and original paging space characteristics differ,
        check the lslv command output.
```

Backup

Today's advanced backup systems are usually centrally controlled across an organization. These systems typically work with multiple operating systems and different types of databases, where they serve to improve data backups. From an operating system perspective, they are usually incomplete. Mostly, they allow file backups, although they do not allow to do a complete backup of the operating system in an easy and efficient way. You can use them to back up a file containing the system image (*mksysb*), but to create this file and then restore the system, you must use native AIX tools. This chapter outlines the relevant tools that are provided with the operating system.

The efficient creation of a backup is one of the primary tasks of the operating system. As with any UNIX system, AIX lets you use standard tools such as *tar* or *cpio* to back up files. These tools offer the advantage that the copies you create can be restored to another operating system than the one on which they were created. However, as universal tools, they do not consider the specifics of the system; thus, they have less potential than the tools typically created for a particular operating system.

With standard tools, you can efficiently create a copy of the data and restore it back to the system if needed. The copies, although they will not contain certain system-specific information (sparse files, encrypted files), will contain all the primary data. The situation is different in the case of backing up the operating system itself. In such a case, rebuilding the system would require the prior installation of the system on the server, the creation of appropriate structures and file systems, and later the recovery of files from the backup.

To avoid having to undertake so many steps and the associated time wastage, specific backup tools have been created. They enable the easy and rapid recovery of the operating system, including all the disk structures.

A number of AIX-specific tools significantly improve backup performance:

- ***mksysb*** - To back up the system volume group (*rootvg*).
- ***savevg*** - To back up any volume group.
- ***backup*** - To back up any files.

It is additionally important to note the capabilities of the Network Installation Manager (NIM). It is a valuable utility in relation to both the installation of the system and its backup and recovery. It allows you to remotely backup and restore any system (*mksysb*) as well as volume groups with file contents (*savevg*).

Mksysb

This tool is used to create a system backup, that is, a backup of a *rootvg* volume group. This group includes the system logical volumes, file systems, and other structures necessary for AIX. It is good practice to keep non-system files in a separate volume group. This way, you have fewer files to restore after a crash and you can hence restore your operating system faster.

A very important feature of the tool is the ability to create a backup in the form of a bootable tape or

disc, which enables the easy and quick recovery of the operating system. A backup file can also be restored over the network using the NIM utility. When you create a backup directly onto a tape or disc, restoring your operating system is very easy. You simply have to change the boot order in such a way that the first one is a backup tape drive or DVD, and then start the server. From the backup media, the kernel for service tasks will be run, and through the menu you will be able to restore the system.

The system that you are restoring in this way can be modified, for example, in terms of its file systems' sizes. You can also restore it on another server and, more importantly, the server does not have to be the same as the one on which the backup was made. To have such flexibility in relation to the restore, you must install the system with the *Enable System Backups to install any system* option. Selecting this option (which is selected by default) installs all the possible drivers to the existing devices on the Power Systems platform.

Main features of the mksysb tool:

- Used only for the backup of the *rootvg* volume groups - A similar tool is used for the backup of other volume groups, namely *savevg*.
- Saves information about the volume group, the logical volumes and their locations, the policies that are associated with them, and the files.
- Saves paging space definitions.
- Does not back up unmounted file systems.
- Files are saved using the *backup* command.

Structure of the data on the tape

The system backup has more or less the same structure regardless of the medium on which it was saved, although there may be minor differences due to the specifics of different types of drives. The detailed structure of a backup can be important, mainly in the case of a copy made on tape, due to the sequential access to such data. Therefore, the structure of the backup medium will be shown using the example of a tape.

The information on the backup tape made using the *mksysb* tool has a special arrangement that divides the various data stored there into several sections, as shown in Figure 0-1.

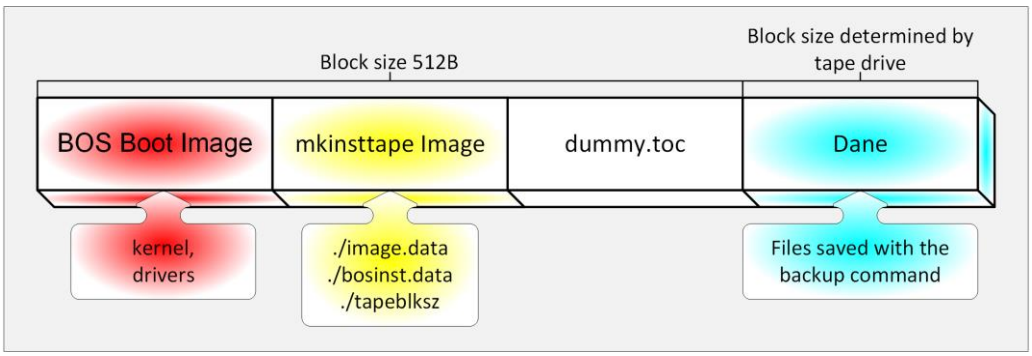

Figure 0-1: Mksysb - Data on the tape.

Individual sections of the backup tape:

- **BOS boot image** - Contains the system's bootable kernel and the drivers required to restore the system. This section is created using the *bosboot* command. When booting from the tape, the kernel included in this section is started. It takes control and provides a menu with which we can decide on further actions.
- **mkinsttape image** - Here is the file that defines the structures to be created when the system is restored *(image.data*) as well as the file that decide how the restore process should work (*bosinst.data*). The most important files are *image.data*, *bosinst.data*, and *tapeblksz*.
    - ***image.data*** - Describes the rootvg volume group and the structures that are in it. It contains a detailed definition of all the parameters that determine the rootvg volume group, the logical volumes created, and the file systems created on those logical volumes. This is a text file created using the *mkszfile* command. Depending on how the command was invoked during the backup creation, the file is rebuilt (*mksysb -i*), or else a previously created version of the file is used (located in the / directory). As this file is persistent, you can modify it before performing a backup. That way, you can customize it and then use it with the *mksysb* command. After restoring, you will have a customized system built according to your custom *image.data* file.
    - ***bosinst.data*** - Describes how the installation program on the backup tape should behave. In other words, this file can contain the answers to questions that are asked when the system boots from the backup; thus, it automates the system recovery process. The restore is preceded by questions, so that you can restore the backup with some modifications.
    - ***tapeblksz*** - Contains information about the size of the block used to write the fourth section (i.e., the data).
- **Data** - This section is the largest. Here, all the volume group files are stored. The files are saved using the *backup* command. This makes it possible to restore individual files from the backup. To restore individual files, use the *restore* command for the fourth tape section where the files are stored. An example command listing the contents of the backup is as follows:

```
restore -Tvs 4 -f /dev/rmt0.1.
```

Alternatively, use the appropriate SMIT menu (*smit restmksysb*).

Backup creation

You can create a backup during normal system operation. However, it is recommended that you set the option *Disable software packing of backup* to *yes*, which will disable data compression prior to saving to tape. This will prevent the possible inconsistency of the backup, which can occur when the application or system modifies the file during its compression.

The possibilities for creating a system backup are displayed in the SMIT menu (*smit mksysb*), as shown in Figure 0-2.

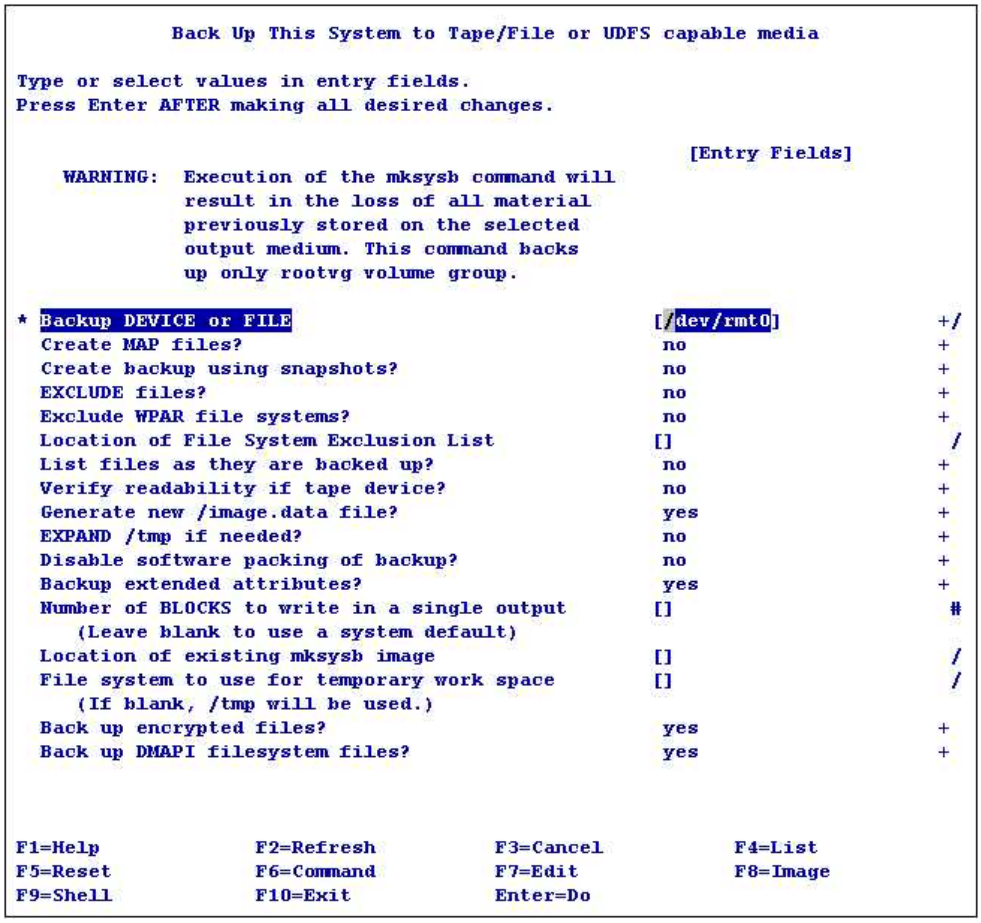

Figure 0-2: Smit mksysb - Create a system backup.

Running a backup defined in this way calls the *mksysb* command with the appropriate parameters.

Key parameters to specify on the menu:

- **Backup DEVICE or FILE** - Select the device on which to back up (*/dev/rmt0* - typical device representing the tape drive, */dev/cd0* - typical device representing the optical drive).
- **Create Map files?** (*mksysb -m*) - Writes additional logical volume information. It allows you to reproduce logical volumes in exactly the same physical location on the disk as where they were when you created the backup.
- **Create backup using snapshots?** - Specifies whether to use snapshots during backup. The snapshots are described in the section on file systems.
- **EXCLUDE files?** (*mksysb –e*) - Bypasses the files listed in */etc/exclude.rootvg* during backup.
- **Verify readability if tape device?** (*mksysb -V*) - After making a copy, checks the correctness of the recording made on the tape. This option checks only the physical correctness of the records. It does not check that the backup is bootable or that the data on the tape is consistent.
- **Generate new/image.data file?** (*mksysb -i*) - Determines whether the system should create a new */image.data* file, or whether to use an existing one. This option is useful if you want the backup to contain definitions of the disk structures that are different to those on the original

system. In such a case, edit the contents of the existing file, and then perform a backup with the option set to *no*.

- **Expand/tmp if needed?** (*mksysb -X*) - The */tmp* file system is used during backup to temporarily store the boot image. This boot image is saved to the medium on which the system backup is being created. This option determines whether the */tmp* file system should be enlarged if there is no space in it. This does not matter if you are using good practice, since you will ensure that there is always free space in the */tmp* file system. Keep in mind that various system utilities very frequently use this file system.
- **Disable software packing of backup?** (*mksysb -p*) - Specifies whether to compress the files before saving. Disabling compression is recommended in two cases:
  - When the target drive itself has compression mechanisms (in the case of tape drives, this is the rule).
  - When files are modified during backup. In this way, you can prevent errors that may result from changing the contents of the file while its compression is in progress.
- **Backup extended attributes?** - This option saves additional file attributes in the backup, such as AIXC and NFS4 ACL.
- **Number of BLOCKS to write in a single output** (*mksysb -b block_size*) - Specifies what block size the system uses to write the backups to tape.
- **Location of existing mksysb image** - Specifies the location of the mksysb backup file that we want to transfer to the tape and creates a bootable backup.

When creating a typical system backup, there are several key actions, which are presented in Figure 0-3.

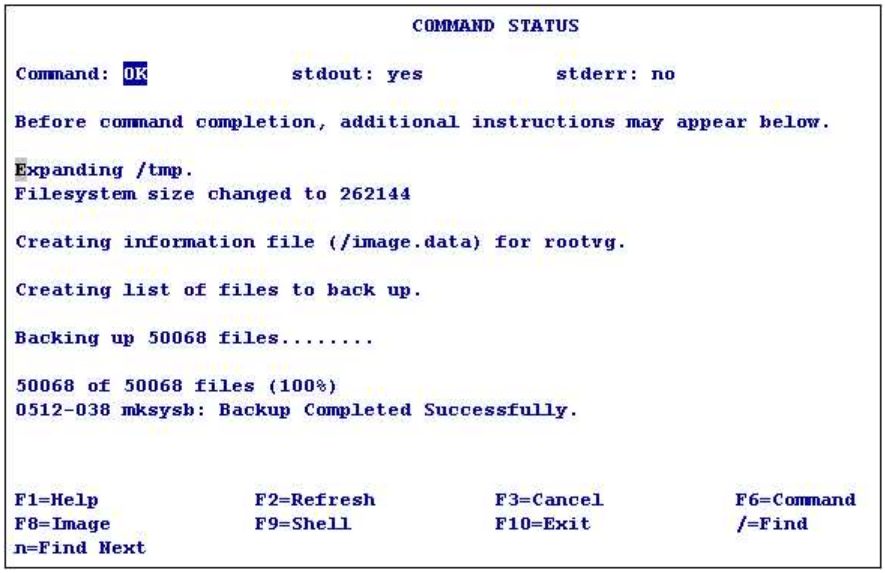

Figure 0-3: smit mksysb - Create a system backup - Status window.

- **Expanding/tmp** - The */tmp* file system is enlarged. This is due to the setting of the *Expand/tmp if needed* parameter. The message means that there is no space for the temporary files created by *mksysb*, so the temporary file system has been enlarged.
- **Creating information file (/image.data) for rootvg** - Creating a file that describes the rootvg volume group and the structures within it. This file is one of the first to be saved on

a medium. During system recovery, one of the first steps is to create structures (logical volumes, file systems, etc.) on disks according to their description in the file.

- **Creating tape boot image** - This entry does not appear on the figure above because the backup was made to a file, not to a tape. It would appear in the case of a tape backup. It would indicate the creation of a copy of the system kernel (using the *bosboot* command). A copy of the kernel would be saved on the backup medium. This kernel controls the further operation of the server when it is started from the backup medium.
- **Creating a list of files to back up** - Only after creating a full list of files does mksysb perform their backups. This means that when you create a backup on a normally running system, there may be errors that indicate some files to be missing. You do not usually need to worry about these messages because, as you know, many temporary files appear in the system and then disappear after a while. If you know what the files are, then it's a good idea to exclude them from the backup so that the backup ends without warning.
- **Backing up** - Backup files according to the list prepared before. This is done using the backup command. It is described later in this chapter.

Backup restoration

You can restore the system via the network using a NIM server, a bootable optical media (DVD), or a tape created with the mksysb utility. To do this using a tape, set the appropriate boot order (bootlist command or SMS mode), insert the backup medium into the drive, and start the server. The system will boot from the medium and, interactively, using the implemented menu, restore the system. If there are any problems with booting from the backup medium, then you can boot from the installation DVD and start up the system recovery process.

After starting the server from the backup medium, select *Start Maintenance Mode for System Recovery* and then *Install from a System Backup*. At this point, the system asks you to select the device to restore from (usually */dev/rmt0* for the tape drive, */dev/cd0* for the DVD drive). After making the selection, it is still possible to exert an influence on how to restore the backup through the restore menu. This is shown in Figure 0-4.

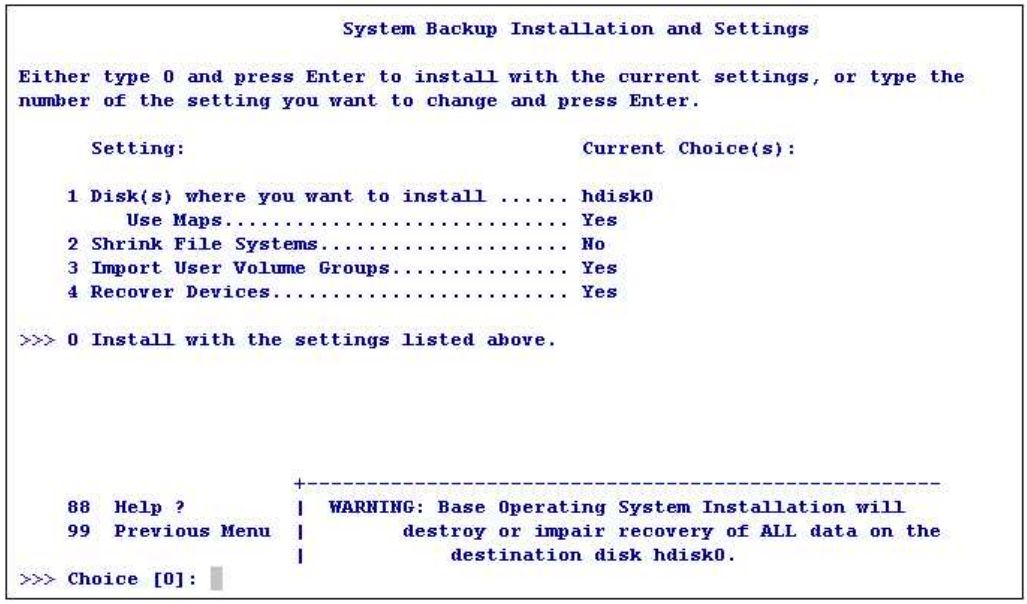

Figure 0-4: Restoring the system from a backup.

Parameters you can modify:

- **Disk(s) where you want to install** - The disk on which to restore the system. If there is a disk in the system that is not allocated to any volume group, you can restore the system to it. This way, you can have two copies of the operating system on your server: original and restored to an extra disk. You can run any of these systems by changing the boot order.
- **Use maps** - This option determines whether the logical volumes created during the restore are to be located in exactly at the same disk location as they were when creating the backup.
- **Shrink File Systems** - Determines whether to reduce all the file systems to the minimum size. The minimum size means the smallest size that will hold all the files on a given file system.
- **Import User Volume Groups** - Specifies whether the volume groups on other disks are to be

imported to the restored system. Other volume groups can be imported at any time using the *importvg* command, so this option is of little importance.

- **Recover Devices** - Determines whether to restore the device database. If the backup is restored on the server on which it was created, the option should be set to *yes*; otherwise, it should be set to *no*.

During system recovery, the contents of the target disks are destroyed. The structures of the restored system are then created: a volume group, logical volumes, and file systems. Only after they have been created the files are copied from the backup medium.

A backup created with the mksysb tool can also be used for "fresh" installations of the system on other servers. This installation mode is often referred to as cloning.

Parameters of the restore

The customized restore of the system is very useful feature. As mentioned earlier, information concerning the structures to be created during system recovery is included in the *image.data* file. By modifying that file, you can influence what structures are to be created, their size, and other parameters.

To perform a "customized restore," restore the system based on the modified *image.data* file. The complete "customized restore" is available when using the NIM server as a backup restore system. In this case, you are able to replace the *image.data* file with its modified version and then restore the system backup according to the parameters listed there. For backups on tape or optical media, you must first modify */image.data* and then create a full backup without the *-i* option. This way, you will save the new backup with the "old" modified *image.data* file. This solution is most commonly used when you need to restore a system that is mirrored to a server that does not have enough resources to create a mirror.

Before modifying a file for use with a NIM server, it must first be read from the backup medium. The *smit restmksysb* menu is the easiest way to do this.

The *image.data* file consists of several sections. They contain parameters for the same backup, volume group, physical volumes, logical volumes, and file systems. Here are some examples:

Volume Group Parameters:

```
vg_data:
        VGNAME= rootvg
        PPSIZE= 16
        VARYON= yes
        VG_SOURCE_DISK_LIST= hdisk0
        QUORUM= 2
        ENH_CONC_CAPABLE= no
        CONC_AUTO= no
        BIGVG= no
        TFACTOR= 1
        CRITVG= no
```

Parameters for Physical Volumes (disks):

```
source_disk_data:
```

```
        PVID= 00f62177fc930906
        PHYSICAL_LOCATION= U8233.E8B.062177P-V10-C3-T1-L8100000000000000
        CONNECTION= vscsi0//810000000000
        LOCATION=
        SIZE_MB= 8192
        HDISKNAME= hdisk0
```

Logical Volume Parameters:

```
lv_data:
        VOLUME_GROUP= rootvg
        LV_SOURCE_DISK_LIST= hdisk0
        LV_IDENTIFIER= 00f6217700004c0000000158f791b446.4
        LOGICAL_VOLUME= hd4
        VG_STAT= active/complete
        TYPE= jfs2
        MAX_LPS= 512
        COPIES= 1
        LPs= 21
        STALE_PPs= 0
        INTER_POLICY= minimum
        INTRA_POLICY= center
        MOUNT_POINT= /
        MIRROR_WRITE_CONSISTENCY= on/ACTIVE
        LV_SEPARATE_PV= yes
        PERMISSION= read/write
        LV_STATE= opened/syncd
        WRITE_VERIFY= off
        PP_SIZE= 16
        SCHED_POLICY= parallel
        PP= 21
        BB_POLICY= relocatable
        RELOCATABLE= yes
        UPPER_BOUND= 32
        LABEL= /
        MAPFILE= /tmp/vgdata/rootvg/hd4.map
        LV_MIN_LPS= 11
        STRIPE_WIDTH=
        STRIPE_SIZE=
        SERIALIZE_IO= no
        FS_TAG=
        DEV_SUBTYP=
```

File System Parameters:

```
fs_data:
        FS_NAME= /
        FS_SIZE= 688128
        FS_MIN_SIZE= 353484
        FS_LV= /dev/hd4
        FS_JFS2_BS= 4096
        FS_JFS2_SPARSE= yes
        FS_JFS2_INLINELOG= no
        FS_JFS2_SIZEINLINELOG= 0
        FS_JFS2_EAFORMAT= v2
        FS_JFS2_QUOTA= no
        FS_JFS2_DMAPI= no
        FS_JFS2_VIX= yes
        FS_JFS2_EFS= no
```

The parameters in the majority of sections are mostly the same ones that are defined when creating individual structures using the SMIT tool or appropriate commands. If you modify the parameters directly in the file, you should be consistent. That is, as you make changes in one place, you must

anticipate what other changes you should make. For example, when you reduce the size of a physical partition, you must consider that the size of the file systems will be also reduced (logical volumes are made up of a specified number of physical partitions). Such a change may result in a lack of space for all the files. Therefore, in this case, you should increase the number of logical partitions and thus the physical partitions per logical volume.

When the *image.data* file has been modified, you can start the restore using the Network Installation Manager. You can also specify a modified *bosinst.data* file that describes how to restore the backup. In this way, you can automate the restore or cloning process. You can configure the *bosinst.data* file to restore automatically without interacting with the administrator. Sample *bosinst.data* file content is available in the operating system on the path: */var/adm/ras/bosinst.data*.

```
control_flow:   # Definition of the parameters that affect the interactive installation.
    CONSOLE = Default
    INSTALL_METHOD = overwrite
    INSTALL_EDITION = enterprise
    PROMPT = yes
    EXISTING_SYSTEM_OVERWRITE = yes
    INSTALL_X_IF_ADAPTER = yes
    RUN_STARTUP = yes
    RM_INST_ROOTS = no
    ERROR_EXIT =
    CUSTOMIZATION_FILE =
    INSTALL_TYPE =
    BUNDLES =
    SWITCH_TO_PRODUCT_TAPE =
    RECOVER_DEVICES = Default
    BOSINST_DEBUG = no
    ACCEPT_LICENSES =
    ACCEPT_SWMA =
    DESKTOP = NONE
    INSTALL_DEVICES_AND_UPDATES = yes
    IMPORT_USER_VGS =
    ALL_DEVICES_KERNELS = yes
    GRAPHICS_BUNDLE = yes
    SYSTEM_MGMT_CLIENT_BUNDLE = yes
    OPENSSH_CLIENT_BUNDLE = yes
    OPENSSH_SERVER_BUNDLE = yes
    FIREFOX_BUNDLE =
    KERBEROS_5_BUNDLE = no
    SERVER_BUNDLE = no
    ALT_DISK_INSTALL_BUNDLE = no
    REMOVE_JAVA_5 = yes
    HARDWARE_DUMP = yes
    ADD_CDE = no
    ADD_GNOME = no
    ADD_KDE = no
    ERASE_ITERATIONS = 0
    ERASE_PATTERNS =
    MKSYSB_MIGRATION_DEVICE =
    TRUSTED_AIX = no
    TRUSTED_AIX_LSPP =
    TRUSTED_AIX_SYSMGT =
    SECURE_BY_DEFAULT = no
    ADAPTER_SEARCH_LIST =

target_disk_data:   # Indication of the disk on which the restore/install should be performed.
    PVID = 00f62177fc930906
    PHYSICAL_LOCATION = U8233.E8B.062177P-V10-C3-T1-L8100000000000000
    CONNECTION = vscsi0//810000000000
    LOCATION = none
```

```
    SIZE_MB = 8192
    HDISKNAME = hdisk0

locale:   # Language settings.
    BOSINST_LANG = en_US
    CULTURAL_CONVENTION = en_US
    MESSAGES = en_US
    KEYBOARD = en_US
```

Maintenance mode

A backup created with the *mksysb* tool is more than just a regular backup. It allows you to do much more than just restore data. As with the installation DVD, this backup can be considered to be a rescue medium. With it, you can get into the system through a "back door" and solve any problems. Service mode is especially useful for correcting critical configuration errors that cause the system to crash when it is started. It is also very useful for changing a forgotten *root* password.

When you start the server from the backup tape or the installation DVD and select *Start Maintenance Mode for System Recovery*, a service menu appears. This is shown in Figure 0-5.

```
                              Maintenance

Type the number of your choice and press Enter.

>>> 1 Access a Root Volume Group
    2 Copy a System Dump to Removable Media
    3 Access Advanced Maintenance Functions
    4 Erase Disks
    5 Configure Network Disks (iSCSI)
    6 Select Storage Adapters
    7 Install from a System Backup



    88  Help ?
    99  Previous Menu

>>> Choice [1]:
```

Figure 0-5: System backup - Maintenance mode.

This menu provides the following options:

- **Access a Root Volume Group** - Starts the system and mounts the main file systems from the *rootvg* volume group. The next step after selecting this option is to select the volume group that you want to get to. Of course, you want to get to the *rootvg* group, but the system in this phase is not able to determine which single volume group is the *rootvg*. The system only sees the group IDs and the disks that belong to them. If there are several different volume groups on the server, you need to determine which one is the rootvg group. Figure 0-6 shows a single disk system, so there is only one volume group to choose from.

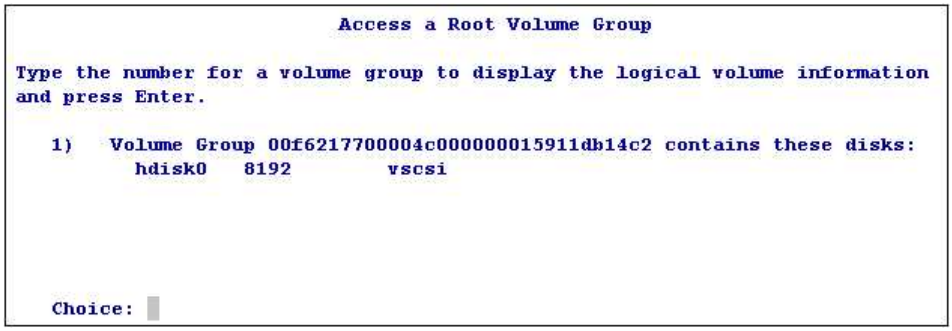

Figure 0-6: Maintenance mode - Select a volume group to access.

When you select a volume group, its logical volumes are displayed (Figure 0-7).

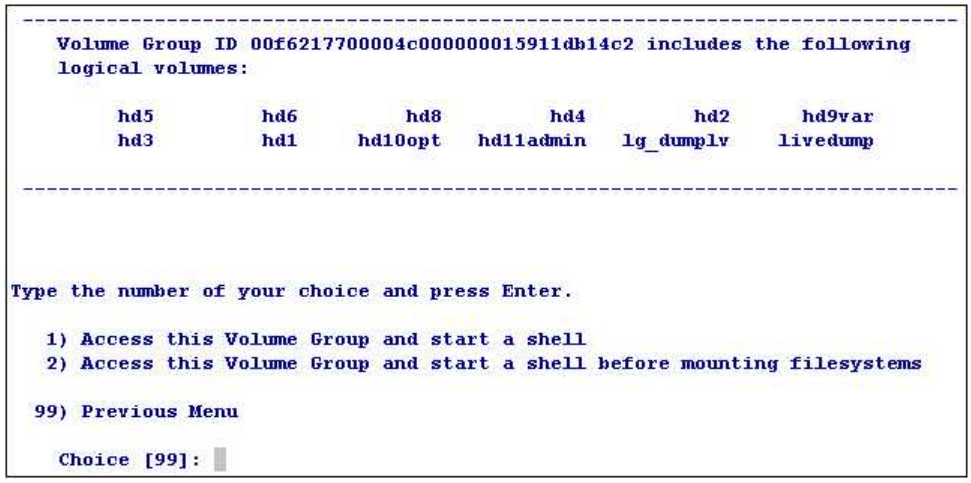

Figure 0-7: Maintenance mode - Information about the content of the volume group.

If you see the names of logical volumes, you can easily determine whether it is a rootvg group. The rootvg group always has the logical volumes listed in Table 0-1.

Table 0-1: Logical volumes in rootvg.

Logical volume	File system
hd1	*/home*
hd2	*/usr*
hd3	*/tmp*
hd4	*/*
hd5	Boot Logical Volume
hd6	Paging Space
hd9var	*/var*
hd10opt	*/opt*

If you find that the wrong group was selected, go back to the previous menu and correct the selection. In the next step, you need to choose whether to mount the file systems or not. The whole operation ends with access to the command line, from which you can make any modifications.

- **Copy a System Dump To Removable Media** - Copy the system state from the moment of failure to an external medium. In the event of a major failure, the system saves its state at this point. This record allows you to check the cause of the failure. It is a good idea to copy a system dump to an external medium and then restore the system to normal operation.
- **Access Advanced Maintenance Functions** - Access advanced features. In practice, this option is rarely used.
- **Erase Disks.**
- **Configure Network Disks (iSCSI)** - Allows you to configure iSCSI disks and verify their work.
- **Select Storage Adapters** - Allows you to select a storage adapter that shows which disks you can choose to install/restore.
- **Install from a System Backup** - Restore the system. This option is described at the beginning of this section.

Savevg

This is used to create backups of any volume group. Typically, it is used to create backups of all the volume groups except *rootvg*. You can use the *savevg* tools to back up *rootvg*, although the backup would be not bootable, so it is far less useful than the previously mentioned backup made with the *mksysb* tool.

In general, *savevg* works in the same way as *mksysb*, except that it does not create a bootable backup.

Structure of the data on the tape

Savevg does not allocate separate sections in which to store parameters describing the structure of the volume group. The entire backup is in one section, which is the files saved by the *backup* command. The files that describe the structures are written first. So, they can be read first, and the volume group can be restored based on them. The files that describe the structure of the volume group are:

- ***Name_VG.data*** - *Name_VG* is the name of the volume group, which means that the file for the volume group *testvg* will be called *testvg.data*. This file is most important for backups created using *savevg*. This is equivalent to the *image.data* file described earlier. The structure of both files and their rules (modification possibilities) are very similar. In practice, this file from the file group described here is necessary to restore the volume group. The rest of the files listed below are auxiliary files.
- ***filesystems*** - Copy of the */etc/filesystems* file. After the volume group has been restored with the necessary structures, the *savevg* command adds the restored file systems to the */etc/filesystems* file. This information is taken from this file.
- ***tapeblksz*** - A file that contains the size of the block that was used to write the backup to the tape. This information may be necessary for reading the tape with a backup.

- ***backup.data*** - A file that informs you which file systems are part of the backup. It also informs you about the file system sizes that the system will create when you restore volume groups using the *Shrink File System* option.

The above files are saved on the tape, and they are also located in */tmp/vgdata/vg_name/*.

Creating a backup

The process of creating a backup is the same as with *mksysb*. You can do this via the smit *savevg* menu, as shown in Figure 0-8.

```
                Back Up a Volume Group to Tape/File or UDFS capable media

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

                                                        [Entry Fields]
    WARNING:  Execution of the savevg command will
              result in the loss of all material
              previously stored on the selected
              output medium.

* Backup DEVICE or FILE                                [/backup/savevg.new]    +/
* VOLUME GROUP to back up                              [data]                  +
  List files as they are backed up?                     no                     +
  Generate new vg.data file?                            yes                    +
  Create MAP files?                                     no                     +
  Create backup using snapshots?                        no                     +
  EXCLUDE files?                                        no                     +
  Location of File System Exclusion List                []                      /
  EXPAND /tmp if needed?                                no                     +
  Disable software packing of backup?                   no                     +
  Backup extended attributes?                           yes                    +
  Number of BLOCKS to write in a single output          []                      #
     (Leave blank to use a system default)
  Verify readability if tape device?                    no                     +
  Back up Volume Group information files only?          no                     +
  Back up encrypted files?                              yes                    +
  Back up DMAPI filesystem files?                       yes                    +

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

Figure 0-8: smit savevg - Back up a volume group.

Running such a defined backup will trigger the *savevg* command with the appropriate parameters. When you create a backup using *savevg*, you specify the same parameters as in the case of *mksysb*. The main difference is that you must additionally specify the name of the volume group that is to be backed up.

When creating a backup, the first thing that is done by the system is the creation of information files, that is, *name_VG.data*, *backup.data*, *filesystems*, and *tapeblksz*, in the */tmp/vgdata/NameVG/* directory. Then, these files and all the other files from this volume group are saved on the backup medium (refers to mounted file systems).

Restore the backup

You can restore the volume group using the *restvg* command or the *smit restvg* menu. Before you restore a backup, you can change the parameters of the structures that will be created during the process. This is the same as for a system backup, except that the *name_VG.data* file is modified rather than *image.data*.

To do this, you need to copy */tmp/vgdata/nameVG/name_VG.data* or its equivalent from the backup. Then, edit the file and, in the *Remake a Volume Group* menu, enter the path to this modified file in the *Alternate vg.data file* field.

The method of restoring the backup using SMIT (*smit restvg*) is shown in Figure 0-9.

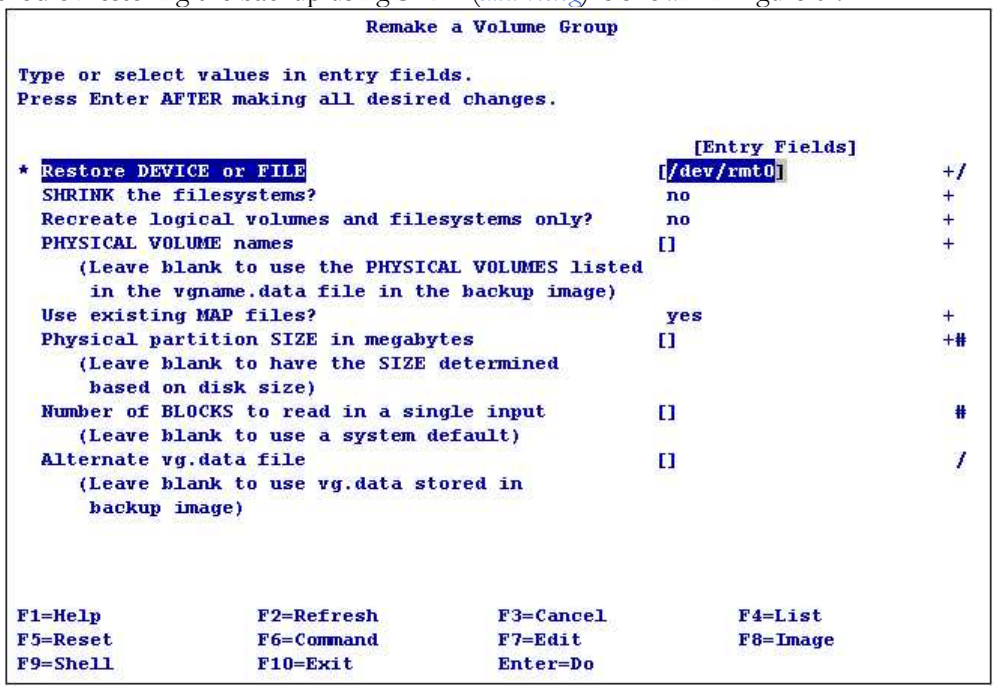

Figure 0-9: smit restvg - Restoring a volume group.

Accepting the defined menu will call the *restvg* command with the appropriate parameters, which in our case will be:

```
# /usr/bin/restvg -q -f'/dev/rmt0'
```

Key parameters:

- **SHRINK the filesystems?** - Determines whether to recreate the logical volumes and file systems shrunk to the minimum size that fits the data.
- **Recreate logical volumes and filesystems only?** - Specifies whether to restore only structures without files or with files.
- **PHYSICAL VOLUME names** - Indicates the physical volumes on which to restore the volume group.
- **Use existing MAP files?** - Specifies whether to place the logical volumes on exactly the same

physical partitions as they were before the backup (only works on backups created using the Create MAP files option set).

- **Alternate vg.data file** - Specifies the path to the alternate file that contains the backup metadata (that is, to the modified *vg.data* file). Do not use the */tmp/vgdata/group_name/VG.data* file as an alternative. You should copy it to another location first, since when you restore a volume group, one of the initial steps that the system takes is to restore the metadata files. This means that the */tmp/vgdata/groupname/name_VG.data* file is replaced with the same file from the backup, and the volume group structures are then restored according to that file.

Backup and restore tools

Backup and *restore* tools are AIX-specific. They are used by *mksysb* and *savevg* to backup files, although they can also be used directly. They allow you to back up files in the same way as standard UNIX tools such as *tar* or *cpio*. However, they offer the advantage that they can create incremental backups, support sparse files, and save extended file attributes.

The contents of regular files that we usually deal with are completely written to disk. In the case of sparse files, file system blocks filled with null values (NULL) are skipped when writing. This is shown in Figure 0-10.

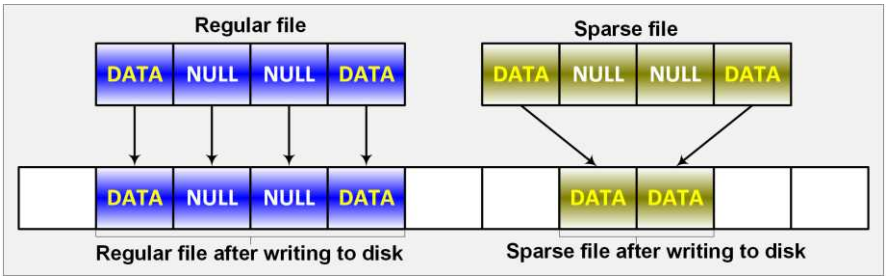

Figure 0-10: Comparison of a regular and a sparse file.

When using such a means of writing, sparse files can take up less disk space than regular files. In addition, operations performed on sparse files can prove more efficient, since they require fewer disk operations.

Backups made using the *backup* command consider the specifics of the sparse files, so they can take up less space than backups made using other tools. Restoring such backups using the *restore* command will save the files in a sparse form. In the case of backup using traditional commands such as *tar* or *cpio*, the stored files become regular files.

With these tools, you can create backups in two ways:

1. Backup by file name

This allows you to create backups of any files. The command must be called with the *-i* parameter in order to retrieve the file names from the standard input. Only the specified files are stored. If a directory name is specified in a standard input, only the directory (without content) will be saved.

When using this method, you typically associate a backup command with the find command. It is important to determine whether you use a relative path (e.g., *./home*) or an absolute path (e.g., */home*). In the former case, you can restore files to any directory, while in the latter case, the files you restore will be created in the directory from which they were downloaded to the backup. A better solution is to use relative paths, since they give you more flexibility. Examples:

Backup of the whole */home* system file:

```
# find /home -print | backup -if /dev/rmt0
# Backup to the tape (rmt0).
# "-i" - The names of files to back up will be read from the standard input (from the find command).
```

Backup selected files with compression:

```
# find . -name "bac*" | backup -ivpf archive   # Backup of the files specified by the find command.

# backup -ivpf archive <filelist   # Backup of the files specified in the "filelist" file.
```

Parameters used:

- ***-i*** - The names of files to back up are read from the standard input (parameter required for backup by file name).
- ***-f file_or_device*** - Specifies the file or device on which to write the backup.
- ***-v*** - Displays information about the backup process.
- ***-p*** - Compression of files (compress files only up to 2 GB).

2. Backup by i-node

This type of backup allows you to back up entire file systems. It does not allow you the ability to back up individual files, but instead gives you the ability to create incremental backups. When you create a backup of this type, you must specify a level (0 to 9) on which to create it. The levels work so that when you have an x-level backup and then create a new one at x + 1 level, only those files that have been modified since the x-level backup are written.

Restoring the entire file system should be done in order, starting from the lowest numbered backup. It is assumed that the level 0 backup is a full backup. If you perform a level 0 backup and after some time a level 1 backup, then the restoration of the entire file system should proceed in the same way: first restore backup level 0, then level 1.

Information about the date, time, and level of each incremental backup is stored in */etc/dumpdates*. Sample content:

```
# cat /etc/dumpdates
/dev/rhd1 0 Mon Dec 19 03:00:20 2016   # Level 0 file system /dev/rhd1 backup.
/dev/rhd1 1 Mon Dec 19 05:00:52 2016   # Level 1 file system /dev/rhd1 backup.
```

```
# lsfs /dev/hd1   # File system hd1 (/home).
Name      Nodename   Mount Pt    VFS   Size    Options    Auto Accounting
/dev/hd1    --        /home      jfs2  32768   --         yes  no
```

If the */etc/dumpdates* file is empty, a backup at any level will write all the files in the file system.

This type of backup is recommended for unmounted file systems. Although this is not a physical limitation and you can back up a mounted file system, but you will not be 100% sure of its correctness. Any modification of the files during the backup may cause their integrity to be compromised. In addition, this type of backup can only be performed on JFS and JFS2 file systems. Examples:

```
# backup -0 -uf /dev/rmt0 /home   # Full backup of the file system /home (level 0).

# backup -1 -uf /dev/rmt0 /home   # Incremental backup of the file system /home (level 1).
```

Parameters used:

- ***-0***, ***-1***, - Indicates the level of the backup (the levels range from 0 to 9, backup level 0 is known as full a backup, since all the files of that file system are always backed up on this level).
- ***-u*** - Writes information about this backup to */etc/dumpdates*. This parameter is required for incremental backups.
- ***-f filename_or_devicename*** - File or device on which the backup is performed.

You can use the *restore* command to browse and restore the backup content. You can also use it to restore the entire backup or extract individual files (this applies to both types of backups). Examples:

Checking backup contents:

```
# restore -Tvf / dev / rmt0
```

Restoring the */home* file system from the tape:

```
# restore -xvf / dev / rmt0 / home
```

Parameters used:

- ***-T*** - Displays information about the backup content.
- ***-v*** - Displays additional information about the command execution status.
- ***-f filename_or_device*** - Indicates the file or device on which the backup is located.
- ***-x*** - Restores the specified files (in the above case, the whole */home* file system).

Standard UNIX system tools

You can use the standard tools available on all UNIX systems to back up AIX. A backup created with these tools is universal and independent of the system. This means that it can be moved between different UNIX systems. It can even be moved between UNIX and Windows, so long as you install the appropriate tool in the second system. These standard tools are *tar* and *cpio*.

Tar

Tar is a traditional backup method used by UNIX systems. With it, you can create backups in files and on tape. AIX 7.1 with TL 3 and earlier versions let you back up files up to 8 GB in size. In newer versions, that is, above AIX 7.1 TL3, this restriction has been removed.

As with the backup utility, there are two ways you can back up your files:

- **absolute paths** - You must restore the files to their original destination.
- **relative** - You can restore the files anywhere in the file system.

Tar supports the backup of extended file attributes (ACLs), but it does not support sparse files. A sparse file is backed up and restored to a regular file. This means that empty values are also written to the backup and, once restored, the file becomes regular.

Examples:

Creating a backup of the *`/home`* directory to a tape device *`/dev/rmt0`*:

```
# tar -cvf /dev/rmt0 /home
```

Listing files from the *`tar`* archive:

```
# tar -tvf /dev/rmt0
```

Restore files from the archive:

```
# tar -xvf /dev/rmt0 /home/user
```

Parameters used:

- ***`-c`*** - Backup creation.
- ***`-v`*** - List all the processed files.
- ***`-f filename_or_device`*** - Indicates the file or device to which the backup should be written.
- ***`-t`*** - List files from the archive in the order they occur.
- ***`-x`*** - Restore specific files or directories (in the above example, it restores the */home/user* directory along with the contents).

Cpio

Like *tar*, *cpio* can also be used to transfer data between systems. It allows you to create files up to 2 GB in size (due to XPG/4 and POSIX.2 standards). Again similar to *tar*, it does not support sparse files.

Examples:

Back up files from the current directory and subdirectories:

```
# find . -print | cpio -ov >/tmp/backup.cpio
```

Checking backup content:

```
# cpio -itv </dev/rmt0
```

Restore the files from the backup located in /tmp/backup.cpio:

```
# cpio -idv < /tmp/file.cpio
```

Parameters used:

- ***-o*** - Reads the file names from the standard input and creates their backup on the standard output.
- ***-i*** - Reads the backup from the standard input and restores the files.
- ***-t*** - Checks the contents of the backup.
- ***-d*** - Creates directories, if needed.
- ***-v*** - Displays the names of the processed files.

Chapter 10. Server and Operating System Startup

AIX only works on IBM Power Systems servers. Therefore, the system startup is closely related to how these machines work. Power Systems servers are typically connected to HMCs that allow server management (one server can be connected to two consoles, and each console can support multiple servers). Most servers are virtualized, and systems are created on the LPARs created from server resources. This scenario, which is shown in Figure 10-1, is described later in this chapter.

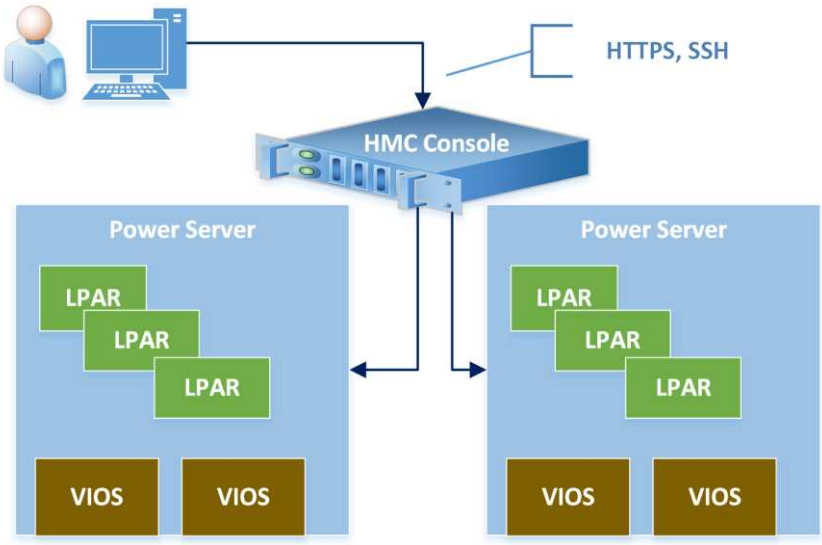

Figure 10-1: Servers and HMC.

If the server has been used and configured with the HMC and shut off from the power supply, the startup phases will look like this:

1. **Connect the server to the power supply**

When you connect the server to the power it is initialized. The service processor is started. Its role is to manage the server and provide communication with the HMC. The service processor code is part of the server firmware. At the end of this phase, the server is in the "Power Off" state, and this information is displayed on the HMC. At this point, you can access the ASM (Advanced System Management) tool from the console. The ASM tool is an interface wherein you can make basic server configurations, although it is used very rarely. The default user and password for ASM are *admin/admin*. Administrators often do not change the password, which leads to lower server security. Examples of the configuration options are shown in Figure 10-2.

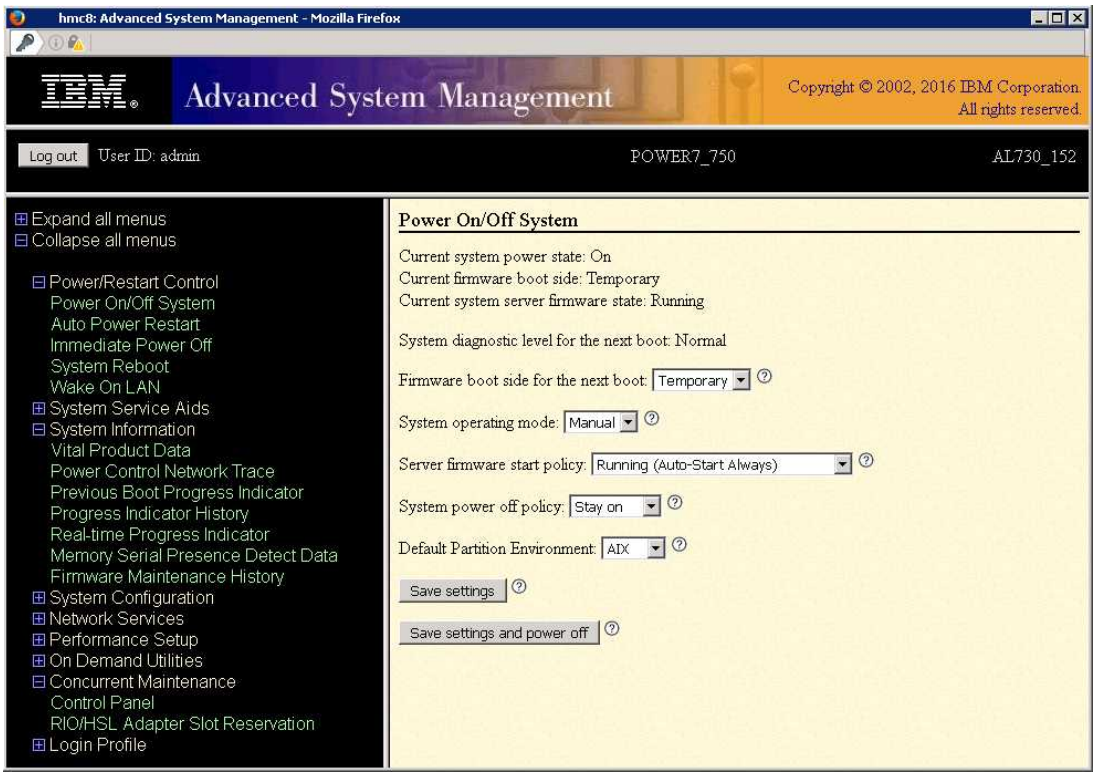

Figure 10-2: Advanced System Management (ASM).

2. Startup in standby mode

The next step is to start the server from the HMC. Startup results in the initialization of the server resources (CPUs, memory) and the transition to *standby* mode. This information appears in the description of the server shown on the HMC. In standby mode, no LPARs are started. This mode indicates that the server is functional and ready to run (launch, create partitions, or further configure from the HMC).

3. Startup in operating mode

The server goes into this mode following the launch of the first LPAR. This means that the server is fully booted and at least one partition is running on it. This mode is active until the server is shut down or a serious failure occurs that results in a change of state. Even if you close the final partition on the server, its status will be reported as *Operating*.

LPAR activation

The next step is to start the LPAR. It can be run in several ways, as shown in Figure 10-3.

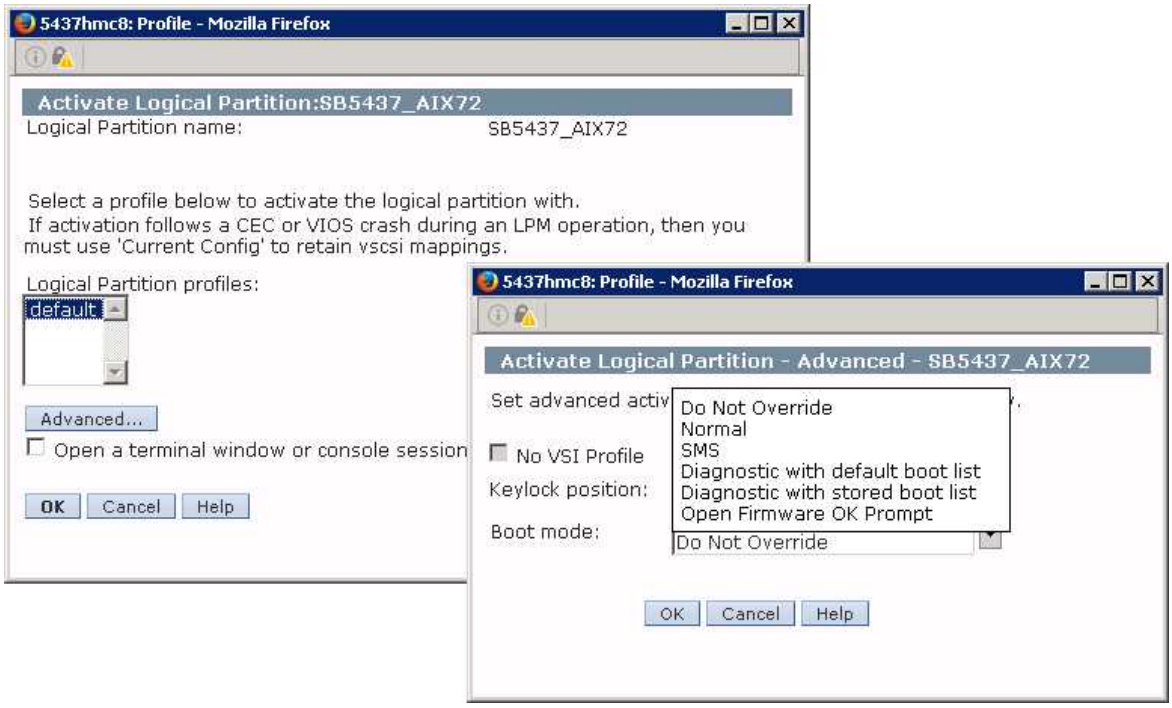

Figure 10-3: Ways to start the LPAR.

During normal operation with the system, the LPARs are run in *Normal* mode, which means that the operating system is started normally. You usually leave this mode alone and do not modify its settings unless you need it. The individual settings:

- **Normal** - Standard system startup.
- **SMS** - Launching into SMS mode. This is described later in this chapter.
- **Diagnostic with default boot list** - Launching the system in diagnostic mode using the default boot list. In this mode, you can diagnose the system with similar capabilities as with the *diag* utility, or else you can run the system in *Single User* mode. The options that are available when running in diagnostic mode are shown in Figure 10-4.

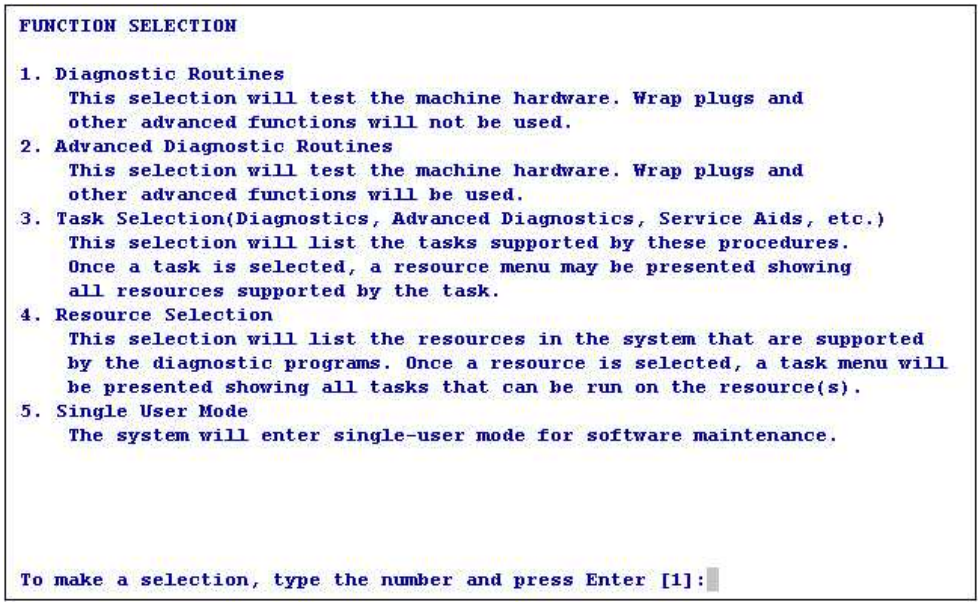

Figure 10-4: Activation of the LPAR - Diagnostic mode.

In this mode, you can perform further operating system repairs if required. Not all the repair options are available in *Single User* mode, mainly because the file systems are mounted in this mode. Some failures require access to unmounted file systems in order to perform a repair. Without such access, the system may not be able to mount file systems and run in *Single user* mode. In that case, the *Maintenance* mode is helpful. This mode requires the system to boot from an external medium, such as a mksysb backup on tape or DVD, or from a *SPOT* resource from a NIM server. The most important differences between the *Single User* and *Maintenance* modes are detailed in Table 10-1 below.

Table 10-1: Differences between the Single User and Maintenance modes.

Single User	Maintenance
The system is started from the local disk in single user mode. If the system has a problem starting normally, it may not start in this mode.	The operating system is booted from an external source (DVD, tape, NIM server), and it can operate independently from the operating system that is on local disks.
Quicker and easier to access - You do not have to look for the medium you plan to boot from.	More difficult and slower access - You need to configure the boot from an external source.
You need to know the root password to operate in single user mode.	You do not need to know the root password to get into the system. This way you can also change the root password.
Disks and devices are visible as after normal system startup.	Disks and devices can be renumbered (renamed) due to booting from another source.

- **Diagnostic with stored boot list** - Booting the system in diagnostic mode using the boot list that is set for that system (*bootlist* command or SMS mode). The way this option works is in line with the option described above.
- **Open Firmware OK Prompt** - Boots the LPAR to Open Firmware mode (prior to the

operating system startup). IBM engineers occasionally use it to verify LPAR environmental information. In fact, it is used very rarely. An example of VSCSI disk validation follows:

```
0 > ioinfo   # Executing the ioinfo command after starting in "Open Firmware" mode.

!!! IOINFO: FOR IBM INTERNAL USE ONLY !!!
This tool gives you information about SCSI,IDE,SATA,SAS,and USB devices attached to the system

Select a tool from the following

 1. SCSIINFO
 2. IDEINFO
 3. SATAINFO
 4. SASINFO
 5. USBINFO
 6. FCINFO
 7. VSCSIINFO

q - quit/exit

==> 7   # Option 7 - VSCSIINFO.
VSCSIINFO Main Menu
Select a VSCSI Node from the following list:
  #  Location Code               Pathname
------------------------------------------------------------
  1. U8233.E8B.062177P-V10-C3-T1  /vdevice/v-scsi@30000003
  2. U8233.E8B.062177P-V10-C4-T1  /vdevice/v-scsi@30000004

  q - Quit/Exit

==> 1   # Select the first VSCSI device.
VSCSI Node Menu
VSCSI Node String: /vdevice/v-scsi@30000003
------------------------------------------------------------

1. List Attached VSCSI Devices
2. Select a VSCSI Device

3. Enable/Disable VSCSI Adapter Debug flags

q - Quit/Exit

==> 1   # Choice of device listing options:
1. 8100000000000000 - 8192 MB Disk drive (bootable)
2. 8200000000000000 - CDROM drive
```

LPAR activation - SMS mode

As mentioned above, one of the modes in which you can run the LPAR is SMS (*System Management Services*) mode. In this mode, you can pre-configure the system, mainly in the context of the source of the boot and the boot order. You can access this mode by running the LPAR with the corresponding option shown earlier in this chapter or by pressing the *1* key during the normal boot. The second situation is shown in Figure 10-5.

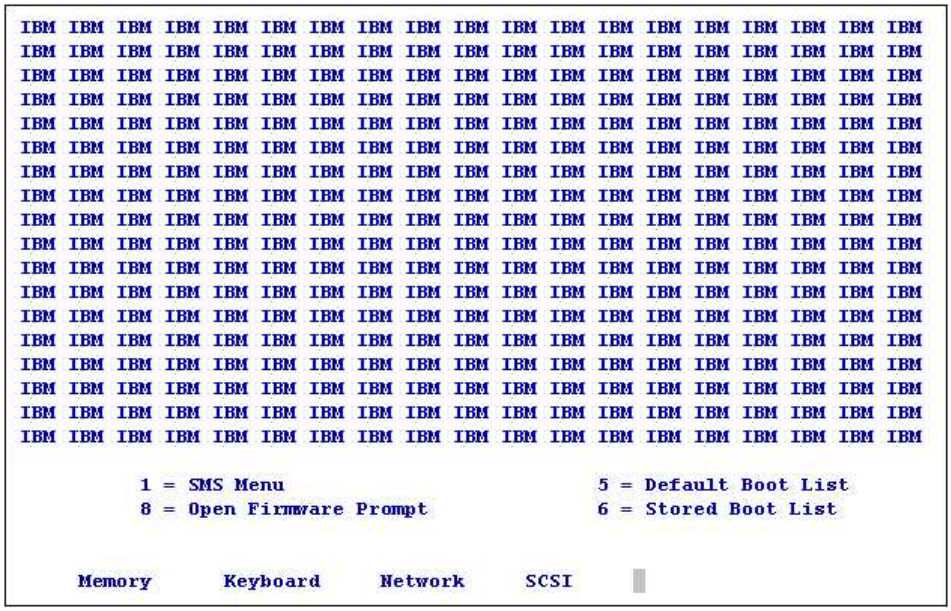

Figure 10-5: Access to SMS mode.

After entering SMS mode, the menu shown in Figure 10-6 is displayed.

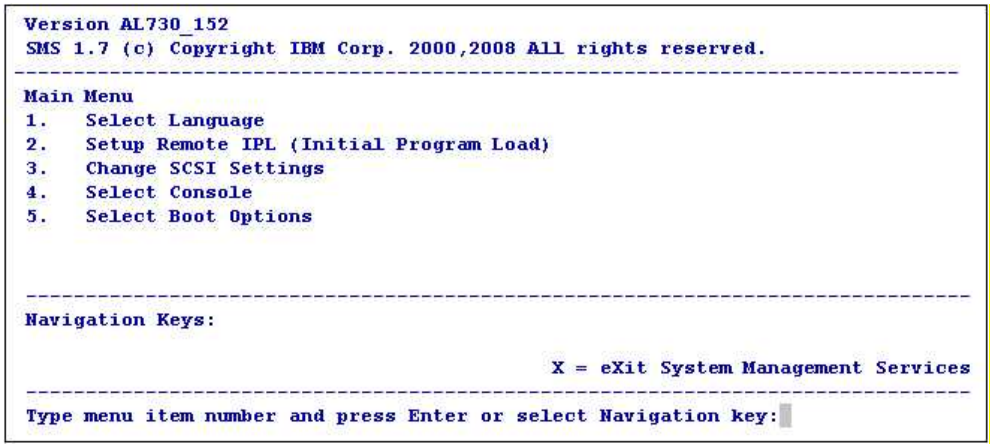

Figure 10-6: SMS mode.

The main goal of SMS mode is to enable the proper booting of the system. This mainly involves setting the appropriate boot list in the *Select Boot Options* option, along with a few other useful features:

- **Network adapter settings** - *Setup Remote IPL (Initial Program Load)*. You can set the IP address, default gateway, mask, and NIM server address on the network adapter, so that you can boot the system from the network. With this option, you can also check the connection by performing a ping operation. This option further allows you to set iSCSI-related parameters.
- **Activation of WWN addresses for FC adapters of a given LPAR**. *Select Boot Options> SAN Zoning Support*. This option allows the FC adapter to log on to the SAN. This allows the storage

guys to view the WWN addresses and configure the SAN network and storage devices accordingly prior to starting the LPAR installation.

Booting the operating system

When you exit SMS mode, AIX starts. If the operating system is already installed on the disk, the next steps look like this: the address of the logical volume that contains the base system components needed to run it is read from the first physical sector of the boot disk. The content of the logical volume is read into the memory and to the virtual disk created in it. One of the items included in this logical volume is the kernel, which takes control of the further process of booting the system. The Logical Volume is called the *Boot Logical Volume* (BLV). It contains the following components:

- Soft *Read-Only Storage* (ROS) - This is part of the code needed during the server startup process. The name *Read-Only Storage* stems from the fact that this code complements the server's initiation process as performed by the firmware. Here are the initialization procedures required by AIX that are not included in the server firmware.
- AIX kernel - This is a kernel specially prepared to perform system initialization. After performing a few tasks, the appropriate kernel will boot from the disk with the system installed.
- ODM database - Device database, containing information about the devices needed to start the system.
- *rc.boot* script - A script used to boot the system. It is called during the system startup phase.
- Basic Commands - Commands needed during the boot process.

After the BLV is loaded into the virtual disk, the system passes control to the loaded kernel. The system phase then begins, which is actually divided into another three phases:

Phase one:

The most important task during the first phase is to prepare the system to activate the *rootvg* volume group. The *init* process starts from the virtual disk. It launches the *rc.boot* script, which is also acquired from the BLV. The script is started with parameter 1. As a result, it loads the device database needed to activate the *rootvg* volume group.

Phase two:

The most important task during the second phase is to activate the *rootvg* volume group. The *rc.boot* script is run with parameter 2. This activates the *rootvg* volume group. Then, the *root* file system (/), which is located on */dev/hd4*, is temporarily mounted on the virtual file system in the */mnt* directory. Next, the following file systems are temporarily mounted: */usr* (device */dev/hd2*) and */var* (device */dev/hd9var*).

Then, the primary paging space (*/dev/hd6*) is activated. The next steps combine a copy of the device database (ODM) from the virtual file system with its equivalent from the *rootvg* volume group. The temporary mounted file systems */usr* and */var*, after they have fulfilled their temporary roles, are unmounted. The *root* file system of the *rootvg* volume group is mounted instead of the *root* file system from the virtual disk. The corresponding */var* and */usr* file systems are mounted at the appropriate

mount points.

The events of the first and second phases are not displayed on the console. You can find them in the boot log, which is available after booting (*/var/adm/ras/bootlog*).

Phase three:

Phase three involves processing the */etc/inittab* file. The file structure and processing rules are very similar in all UNIX systems.

One of the most important scripts called from */etc/inittab* is the *rc.boot*. This time it is called with parameter 3 - */sbin/rc.boot 3*. When viewing the content, you may find that it causes, among other things:

1. Mounting the */tmp* directory.
2. Background synchronization of the *rootvg* volume group. This is done using the *syncvg rootvg* command, and it only applies if there are logical volumes in the *rootvg* group that work in the mirror. During synchronization, the system compares the metadata of each physical partition with the metadata of its copies. If the compared partitions are different, the physical partition recognized by the system as invalid is overwritten with the contents of the partition considered valid. This process can be time consuming, especially in cases of starting after an incorrect shutdown.
3. Running the configurator *cfgmgr* – To configure the devices in the system.
4. Configuration of the console.
5. Updating the device database stored in the BLV, with information from the devices database of the running the system. This is done using the *savebase* command.

The */etc/inittab* file calls many programs and scripts that configure the system. These scripts, like any other, can be freely modified. In addition, you can add other scripts or programs to the above file. Knowing the structure of the */etc/inittab* file is very important, since it allows you far-reaching control over how your operating system boots.

Structure and use of the /etc/inittab file

The */etc/inittab* file is responsible for the starting processes when the operating system starts. This file is also periodically, that is, every 60 seconds, checked by the *init* process. The system checks whether the contents of the file have changed and decides whether to take any action. If new lines are added to the file, then the next time the system checks the file, they will be processed.

Each line of the file consists of four fields separated by a ":" (colon):

Identifier : RunLevel : Action : Command

To aid with the easier understanding of the descriptions of each field, one sample line will be interpreted:

rctcpip:23456789:wait:/etc/rc.tcpip > /dev/console 2>&1 # Start TCP/IP daemons

The *rctcpip* identifier is to be processed on each run level (*RunLevel* field) listed in the second field of the line (i.e., 2, 3, 4, 5, 6, 7, 8, 9). The word "wait" in the third field means that the system will wait for the script */etc/rc.tcpip* to finish before further processing the */etc/inittab* file.

Meaning of individual fields:

Identifier

It uniquely identifies a given line. It must be unique throughout the file. It is used to indicate a line when modifying a file using special commands.

RunLevel

It indicates the run level at which the entry is to be processed. In AIX, the run levels range from 0 to 9. In addition, there is something you can call special levels or signals passed to the *init* process. They are represented by the letters a, b, h, S, s, M, m, Q, and q.

The system always runs at a certain run level. The default level is indicated by an entry in */etc/inittab* that contains the word "*initdefault*:"

```
init:2:initdefault:
```

AIX operates on level 2 by default. This means that all the entries that contain "*2*" in the second field must be processed at system startup.

The run level can be changed during system operation. You can do this using the *telinit* command. In case of a level change, all the processes running at the previous level (as specified in the */etc/inittab* file) that do not have an entry corresponding to the new level are stopped. The system sends a "SIGTERM" signal to terminate the operation. If this is not enough, it sends a "SIGKILL" signal, which kills the process.

The default meaning of each level:

- ***0***, ***1*** - Levels reserved for future use. In other UNIX systems, switching to level 0 shuts down the system, while switching to level 1 causes the system to start up in service mode.
- ***2*** - AIX default run level. The system runs at this level after startup - Access to all system services is guaranteed.
- ***3–9*** - These levels do not have a default function and can be used freely. If you want to use these levels, then you need to create the appropriate entries in the */etc/inittab* file that define its functionality. In other UNIX systems, switching to level 6 usually causes the server to restart.
- ***S***, ***S***, ***M***, ***m*** - These levels represent a service/maintenance run level. At this level, only necessary processes are running. You do not have access to the network or network services. Access to the system is only possible directly from the console.
- ***a***, ***b***, ***c***, ***h*** - Levels that you can use to run additional services. Calling these levels (*telinit* command) triggers processes defined at this run level, although it does not close the processes defined at the previous run level. It also does not change the run level.
- ***Q, q*** - Sends a signal to the *init* process instructing it to read the */etc/inittab* file again.

- *N* - Stops processes before the "respawn." This term is explained below, along with a description of the third field of the entries in */etc/inittab*.

Action

This specifies how the system behaves when processing the entries in the system boot context, when changing the boot level, or when the file is processed at a certain boot level. There are a number of possible entries:

- **respawn** - Instructs the system to start the process if it is not running. The system does not wait for the process to finish, but instead continues to process the */etc/inittab* file. If the process is terminated or interrupted, the system will start it again at the next periodical review (every 60 seconds). An example of this behavior can be seen in the entry for the cron subsystem. If you kill the cron process, the system will restart it in a maximum of 60 seconds, as defined below:

```
cron:23456789:respawn:/usr/sbin/cron
```

- **wait** - Causes a one-time start of the process. The system waits for it to finish before continuing to process the */etc/inittab* file. After the initial processing of this line, each subsequent periodic processing of the file ignores it. Use such entries if you want to use this mechanism to start services in a specific order.
- **once** - Causes a one-time start of the process. The system does not wait for it to finish, but instead continues processing the file.
- **boot** - Causes the process to be run once when the system first reads the */etc/inittab* file. The system does not wait for it to finish but continues processing the file. This entry should be used with the default run level.
- **bootwait** - Causes the process to be run once when the system first reads the */etc/inittab* file. The system waits for it to finish before continuing to process the */etc/inittab* file.
- **powerfail** - Runs the process only when the kernel process has received a "SIGPWR" signal indicating power problems. The system waits for it to finish before continuing to process the */etc/inittab* file.
- **off** - If the process associated with this entry is running, the system interrupts it. The system sends a "SIGTERM" signal to terminate the process. If that is not enough, after 20 seconds it sends a "SIGKILL" signal, which kills the process. In other words, this entry ensures that the process is not run.
- **ondemand** - Works the same way as "respawn," except that it is used with sub-levels (a, b, or c).
- **initdefault** - The line is processed once when the system first reads the */etc/inittab* file. It indicates the run level of the system after the entire boot process. If the run level field is blank, this is interpreted as *0123456789*. This means that the running system will be at run level *9*, since it always uses the highest of these levels. If there is no such line in the */etc/inittab* file, the system will prompt you to specify a run level at startup.
- **sysinit** - The line is processed before the console is taken over by the system. This method is only used to let the system ask for the run level where it should work. The system waits for it to finish before continuing to process the */etc/inittab* file.

Command

A program or script that runs because of an entry processing. It is important to note that if you want the system to keep the process running on (*respawn*) or running off (*off*) permanently, you should run the program directly instead of running it from the startup script. In the case of a script, the mechanism will not work because the system will not be able to identify the processes that it should monitor.

Default contents of /etc/inittab:

This file can be fully customized to suit your needs, although it is rarely modified. You usually add additional lines to it. By default, after installing AIX, the */etc/inittab* file has the following contents:

```
#cat /etc/inittab
# All unnecessary information has been omitted.
init:2:initdefault:
brc::sysinit:/sbin/rc.boot 3 >/dev/console 2>&1 # Phase 3 of system boot
powerfail::powerfail:/etc/rc.powerfail 2>&1 | /usr/bin/alog -tboot > /dev/console # Power Failure
Detection
tunables:23456789:wait:/usr/sbin/tunrestore -R > /dev/console 2>&1 # Set tunables
securityboot:2:bootwait:/etc/rc.security.boot > /dev/console 2>&1
rc:23456789:wait:/etc/rc 2>&1 | /usr/bin/alog -tboot > /dev/console # Multi-User checks
srcmstr:23456789:respawn:/usr/sbin/srcmstr # System Resource Controller
rctcpip:23456789:wait:/etc/rc.tcpip > /dev/console 2>&1 # Start TCP/IP daemons
aso:23456789:once:/usr/bin/startsrc -s aso
rcnfs:23456789:wait:/etc/rc.nfs > /dev/console 2>&1 # Start NFS Daemons
fbcheck:23456789:wait:/usr/sbin/fbcheck 2>&1 | /usr/bin/alog -tboot > /dev/console # run
/etc/firstboot
cron:23456789:respawn:/usr/sbin/cron
clusterconf:23456789:once:/usr/sbin/clusterconf
piobe:2:wait:/usr/lib/lpd/pioinit_cp >/dev/null 2>&1  # pb cleanup
cons:0123456789:respawn:/usr/sbin/getty /dev/console
qdaemon:23456789:wait:/usr/bin/startsrc -sqdaemon
writesrv:23456789:wait:/usr/bin/startsrc -swritesrv
uprintfd:23456789:respawn:/usr/sbin/uprintfd
shdaemon:2:off:/usr/sbin/shdaemon >/dev/console 2>&1 # High availability daemon
trustedboot:2:wait:/etc/rc.trustedboot > /dev/console 2>&1 # Get trusted log and start TCSD
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6
l7:7:wait:/etc/rc.d/rc 7
l8:8:wait:/etc/rc.d/rc 8
l9:9:wait:/etc/rc.d/rc 9
naudio2::boot:/usr/sbin/naudio2 > /dev/null
rcwpars:2:once:/etc/rc.wpars > /dev/console 2>&1 # Corrals autostart
logsymp:2:once:/usr/lib/ras/logsymptom # for system dumps
pfcdaemon:2:once:/usr/bin/startsrc -s pfcdaemon  > /dev/null 2>&1
perfstat:2:once:/usr/lib/perf/libperfstat_updt_dictionary >/dev/console 2>&1
diagd:2:once:/usr/lpp/diagnostics/bin/diagd >/dev/console 2>&1
artex:2:wait:/usr/sbin/artexset -q -c -R /etc/security/artex/config/master_profile.xml >
/dev/console 2>&1
clcomd:23456789:once:/usr/bin/startsrc -s clcomd
xmdaily:2:once:/usr/bin/topasrec -L -s 300 -R 1 -r 6 -o /var/perf/daily/ -ypersistent=1 2>&1
>/dev/null #Start local binary recording
ctrmc:2:once:/usr/bin/startsrc -s ctrmc > /var/ct/ctrmc-inittab.err 2>&1
ha_star:h2:once:/etc/rc.ha_star >/dev/console 2>&1
```

Selected actions in the standard */etc/inittab* file:

- **brc** - The third phase of system startup, that is, running the script */sbin/rc.boot 3*.
- **powerfail** - Runs a script that supports power problems. The actions it takes differ depending on the situation: it logs the problem, sends information to all logged in users about the problem, and in case of serious problems, it causes the system to shut down properly.
- **rc** - Performs the basic tasks necessary for server startup, including activation of non-rootvg volume groups, activation of the paging space, configuration of the device to dump the system state during a crash (Dump device), and checking and mounting file systems.
- **srcmstr** - Starts *srcmstr* (*System Resource Controller - SRC*). This daemon is responsible for handling subsystems on AIX. The subsystems and SCR are described in a separate chapter.
- **rctcpip** - Starts TCP/IP-related daemons such as *inetd*, *sendmail*, *syslogd*, and many others.
- **cron** - Starts the *cron* daemon responsible for scheduling tasks.
- **l2**, **l3**, **l4**, **l5**, **l6**, **l7**, **l8**, **l9** - Run the */etc/rc.d/rc* script with the appropriate parameter that points to the current boot level.

Lines with the identifiers l2, l3, l4, l5, l6, l7, l8, and l9 are very useful. They run the script */etc/rc.d/rc* when entering the appropriate boot level or when exiting the level (change run level, restart the system). This script, depending on the level at which it is run (indicated by the parameter), runs all the scripts defined to run at that level. These scripts are stored in different locations, depending on which level you want to run:

level 2 = */etc/rc.d/rc2.d*

level 3 = */etc/rc.d/rc3.d*

level 4 = */etc/rc.d/rc4.d*

...

level 9 = */etc/rc.d/rc9.d*

All the scripts starting with "K" and "S" will run at the entry to a given level. Startup takes place in alphabetical order, in two phases. First, the scripts that start with "K" with the parameter "stop," then the scripts that start with "S" with the parameter "start." The scripts that begin with "S" (start) are used to start services, while the scripts that start with "K" (kill) are used to close services. A similar mechanism works in most UNIX and LINUX systems. This is the most commonly used mechanism for automatic service startup.

Operations on /etc/inittab

The */etc/inittab* file is a plain text file. You can modify it using any text editor, although this is not recommended. If you make a mistake during manual editing, you can cause the system to crash while it boots.

To reduce the risk of corruption or the possibility of making an error while editing a file, special commands have been created:

- ***lsitab*** - Displays the contents of the file on the screen, for example:

```
# lsitab rctcpip  # Displays a line with the identifier "rctcpip".
rctcpip:23456789:wait:/etc/rc.tcpip > /dev/console 2>&1 # Start TCP/IP daemons
```

- ***mkitab*** - Adds a line, for example:

```
# mkitab „monitor:23:respawn:/usr/sbin/monitor”
```

- ***chitab*** - Changes a line, for example:

```
# chitab „monitor:2345:respawn:/usr/sbin/monitor” # Changes the run levels.
```

- ***rmitab*** - Deletes a line, for example:

```
# rmitab monitor   # Deletes an entry with a "monitor" ID.
```

Actions undertaken using these commands ensure that the file structure is preserved.

Chapter 11. Problem Determination

The *syslogd* daemon runs on any UNIX system. Its role is to receive various types of messages, including error messages, and to save them in the locations specified in the configuration file. On many UNIX systems, you can view information about various events in your system here.

On AIX, there is also a *syslogd* that you can use. However, when it comes to monitoring common errors, there is a daemon specially designed for this purpose (i.e., *errdemon*). This daemon constantly monitors the system and saves the reported errors to its log.

Due to the AIX-specific *errdemon*, this operating system has one of the more transparent error reporting mechanisms of all UNIX systems. The error reporting mechanism is supplemented by the *diag* diagnostic tool. With this tool, you can always confirm whether a bug continues to occur or test suspicious LPAR hardware components.

Error daemon

Errdemon is a constantly running component of the system. It monitors the special file */dev/error*, where the error messages appear. In the event of a failure, the daemon writes the corresponding information to the log, and it compares it with the contents of its database. The comparison helps to identify the additional components of the system from which to gather information in order to better pinpoint the characteristics of the error. After collecting the information, the error and the additional information are logged in the log file (*/var/adm/ras/errlog* by default).

The error daemon can generate a notification when certain errors occur. Defined notifications are sent via the corresponding script that is triggered in response to the event. Scripting allows you great flexibility in terms of implementing notifications. They can be implemented in a variety of ways, such as a console entry (the most popular and easiest method), email, or SMS (if you have the appropriate SMS gateway).

Errdemon is started automatically when booting with the */usr/lib/errdemon* command. When the system calls this command, the daemon runs with the default parameters: file name, size, and the amount of memory allocated to its buffer. Alternatively, it can be run with custom parameters, for example:

```
# /usr/lib/errdemon -B buffer_size -i file_name -s file_size
```

The new sizes given in this way will become the default values. They will be set every time the system or daemon is restarted. You can use the *-l* flag to read the configuration that *errdemon* is running:

```
# /usr/lib/errdemon -l
Error Log Attributes
------------------------------------------
Log File                /var/adm/ras/errlog
Log Size                1048576 bytes
Memory Buffer Size      32768 bytes
Duplicate Removal       true
Duplicate Interval      10000 milliseconds
Duplicate Error Maximum 1000
```

```
PureScale Logging       off
PureScale Logstream     CentralizedRAS/Errlog
```

The first three parameters are the location of the file, its maximum size, and the size of the buffer used by the *errdemon*. The log file is written with the round robin algorithm, which means that when it takes up the whole area defined for the log, subsequent entries will overwrite the oldest ones. In such a case, there will never be a log overflow, and the latest error messages will always be available.

The three lines of the above output starting with the word "Duplicate" control the number of entries that appear in the log in the event of a serial occurrence of the same error. They cause all the occurrences of the same error (no more than 1000) that occur within an interval of no more than 10,000 milliseconds (interval between consecutive occurrences of the error) to be recorded as one error. This means that thousand first occurrence of the error will be treated as a separate error. this is the same as the occurrence of the same error after 10,001 milliseconds. These parameters protect against the flooding of the log with one repetitive error, and they result in a clearer error log.

Usually, there is no need to modify *errdemon* and hence it is left with the default parameters. However, it is important to ensure that its logs are periodically archived. This way, you will have access to older information that would normally be overwritten. A practical case, where old errors are overwritten with the new ones is the loss of one copy of the mirror. This causes many errors, since the loss of copies for each physical partition is reported separately at the first attempt to perform the operation on it.

The error log file is a binary file. Therefore, there is a special command, namely *errpt*, for reading it.

Reading the error log (errpt)

The *Errpt* command provides you with many possibilities for displaying information. Due to their number, only the most important and most frequently used methods will be presented.

The command, by default, displays the information obtained from the current error log file. By using the *-i filename*, you can specify another file to read the information from. This option is required to read the archived error log files.

Running *errpt* without any parameters returns a short list of all the errors, for example:

```
# errpt
IDENTIFIER TIMESTAMP  T C RESOURCE_NAME  DESCRIPTION
B6267342   1221034416 P H cd0            DISK OPERATION ERROR
B6267342   1221034416 P H cd0            DISK OPERATION ERROR
BFE4C025   1214193016 P H sysplanar0     UNDETERMINED ERROR
DE84C4DB   1214043116 I O ConfigRM       IBM.ConfigRM daemon has started.
A6DF45AA   1214043116 I O RMCdaemon      The daemon is started.
69350832   1214043016 T S SYSPROC        SYSTEM SHUTDOWN BY USER
9DBCFDEE   1214043116 T O errdemon       ERROR LOGGING TURNED ON
192AC071   1214041816 T O errdemon       ERROR LOGGING TURNED OFF
```

Due to the small amount of content, the information obtained in this way only allows you to check the general condition of the system. Each column provides the following data:

- **IDENTIFIER** - Error identifier. Each type of error has its own identifier. It does not identify

a specific occurrence of the error, but instead indicates its type.

- **TIMESTAMP** - You can read the date and time of the error occurrence. The data is in the **MMDDHHmmYY** format, where:
  - **MM** - Month;
  - **DD** - Day;
  - **HH** - Hour;
  - **mm** - Minute; and
  - **YY** - Year.
- **T - (Type)** - The type of error. The following types are possible:
  - **TEMP** - Temporary error. This means that there were errors, but they occurred for only a short time, after which the situation returned to normal. This type of error is often ignored following verification, since it do not affect the correct operation of the system.
  - **PERM** - Permanent error. Reports a permanent problem with hardware or software. This type of error should be of interest, since it may have a significant impact on the functioning of the system.
  - **INFO** - An entry that informs about an event that is of importance to the system, but is not an error.
  - **PERF** - An entry indicating a decrease in the performance of the device. An example of a situation where this error occurs is the attaching of an older disk to the SCSI bus on which newer disks based on the newer and faster SCSI standard are working. In this case, the bus speed will be reduced to the highest speed acceptable by all the disks and a PERF error will be reported.
  - **PEND** - A warning that a possible failure of a component will soon occur. The system informs you that the behavior of the device allows you to assume that its life is ending, and you need to think about its replacement.
  - **UNKN** - The type of error cannot be determined.
- **C (Class)** - The error class. There are the following classes:
  - **H** - Hardware errors;
  - **S** - Software errors;
  - **O** - Information entries made by an operator using an *errlogger* command; and
  - **U** - Unspecified class.
- **RESOURCE_NAME** - The source of the problem, or the source that reports the problem
- **DESCRIPTION** - A brief description of the error.

For a detailed list of all the errors, use the *-a* parameter. The *-j* option allows you to select a specific type of error to display, for example:

```
# errpt -aj B6267342   # Detailed information on errors with the identifier B6267342

---------------------------------------------------------------------------
LABEL:          SC_DISK_ERR2
IDENTIFIER:     B6267342

Date/Time:       Wed Dec 21 03:44:56 CST 2016
Sequence Number: 16
Machine Id:      00F621774C00
Node Id:         aix72c1
Class:           H
Type:            PERM
WPAR:            Global
Resource Name:   cd0
```

```
Resource Class:  cdrom
Resource Type:   vopt
Location:        U8233.E8B.062177P-V11-C3-T1-L8200000000000000


Description
DISK OPERATION ERROR

Probable Causes
DASD DEVICE

Failure Causes
DISK DRIVE
DISK DRIVE ELECTRONICS

        Recommended Actions
        PERFORM PROBLEM DETERMINATION PROCEDURES

Detail Data
PATH ID
           0
SENSE DATA
0A00 2800 0000 0110 0000 0200 0000 0000 0000 0000 0000 0000 0102 0000 7000 0200
0000 000A 0000 0000 3E01 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0001 1000 0000 0000 0000 0000 0000 0000 0083 0000
0012 0018 0017
```

Additional information obtained using the *-a* flag:

- **Sequence Number** - The number of the error entry, which is its unique identifier.
- **Node Id** - The system name (equivalent to the name obtained using the *uname -n* command).
- **Resource Class, Resource type** - The class and type of device that is the source of the error.
- **Location** - The location of the device. You can read information such as the number of the PCI slot to which the device is attached. In the example above, cd0 is a virtual drive in the LPAR with the ID 11 (V11) that is pinned to slot 3 (C3), and its LUN ID is L8200000000000000. For physical components, refer to the server documentation for the meaning of the location code.
- **Probable Causes, Failure Causes** - Hardware or software components that may cause an error.
- **Recommended Actions** - The recommended actions to take in order to correct a problem or to verify it. The *PERFORM PROBLEM DETERMINATION PROCEDURES* listed in the example usually mean that the administrator should perform the analysis. In the case of physical devices, he should also perform detailed diagnostics of the affected physical devices using the *diag* tool.
- **Detail Data, SENSE DATA** - Detailed error information. This information may be important to IBM engineers investigating the issue. The information is also used by the *diag* tool when testing a physical component for which an error has been logged (this is the case when you perform the *problem determination* diagnostic from the *diag* menu). In the above example, the Detail Data indicates the path identifier for the device, which can help us to determine the VIOS by which the device is shared.

The *errpt* command has many flags that allow you to accurately determine which errors have occurred

and how to display them. The above usage scenarios show only a small portion of its capabilities, but they are sufficient to obtain basic error information. You can use the *errpt* by using the SMIT menu: *smit error*.

Clearing the error log (errclear)

After long-term use, logs grow to large sizes. Large log sizes make it harder to analyze the logged data, so it is important to periodically clean them. To do so, use the *errclear* command. Before using it, it is best to archive the contents of the log. The easiest way to do this is to use the *errpt -a* command and redirect the result to the file. In this way, we will obtain an archive file in a textual format, which will allow you to view it using any text editor on both the server and any workstation.

The only required parameter for the *errclear* command is the number of days back, counting from the current day, for which the entries are to be retained (older ones will be deleted). In addition, you can specify what types of errors are to be removed, which are to be preserved, and so on. Example usages of the command follow:

```
# errclear 30 # Deletes all entries older than 30 days.

# errclear -d S 30 # Removes all S-class errors (software) older than 30 days.

# errclear -j 192AC071 3 # Removes all errors with the id. 192AC071 older than three days.

# errclear -l 24 0 # Deletes an error with the sequence number 24.
```

The system also automatically cleans the error log. The entries in the configuration file of the cron daemon, which belongs to the root user, are responsible for this task. Cron is responsible for performing specific tasks at a given time, and the entries for cleaning the error log look like this:

```
# crontab -l | grep err
0 11 * * * /usr/bin/errclear -d S,O 30   # Removes all software errors older than 30 days.
0 12 * * * /usr/bin/errclear -d H 90   # Removes all hardware errors older than 90 days.
```

Syslogd

As mentioned earlier, the *syslogd* daemon monitors various types of events that occur on the system and logs their occurrences to a specified location within the system. This is usually the file, system console, or monitoring tool to which the event information is routed through the network.

The *syslogd* daemon is started automatically from the */etc/rc.tcpip script*. When run, it creates a */etc/syslog.pid* text file containing its process number. At startup, the daemon gets information from */etc/syslog.conf*. Its configuration can be customized by modifying the configuration file. All added entries will be activated after the restart of the daemon or after the configuration is refreshed. This can be done in two ways:

1. *refresh -s syslogd* - This is a typical AIX method because it uses the *System Resource Controller (SRC)*, a simplified method for controlling services. The SRC is discussed in a separate chapter.
2. *kill -HUP `cat / etc / syslog.pid`* - The command sends a signal number 1 to the *syslogd* process

(the number is read from */etc/syslog.pid*). The *kill* command should work on any UNIX system.

/etc/syslog.conf file

Each entry in the configuration file must consist of two parts. The first part specifies the source of the messages to be handled (facility) as well as their priority. The second part contains the destination of the log. In addition, each entry can contain a "rotation," which is an optional part that allows you to log the events to multiple files in a rotational way.

The structure of a typical entry.

facility.priority destination rotation

The following facilities are available:

- ***kern*** - Kernel.
- ***user*** - User level.
- ***mail*** - Mail subsystem.
- ***daemon*** - System daemons.
- ***auth*** - Security or authorization.
- ***syslog*** - *syslogd* daemon.
- ***lpr*** - Line-printer subsystem.
- ***news*** - News subsystem.
- ***uucp*** - uucp subsystem.
- ***local0*** through ***local7*** - Local use.
- ***** - All facilities.

The following priorities are available:

- ***emerg*** - Specifies emergency messages (LOG_EMERG). These messages are not distributed to all users. LOG_EMERG priority messages can be logged into a separate file for review.
- ***alert*** - Specifies important messages (LOG_ALERT), such as a serious hardware error message. These messages are distributed to all users.
- ***crit*** - Specifies critical messages not classified as errors (LOG_CRIT), such as improper login attempts. LOG_CRIT and higher-priority messages are sent to the system console.
- ***err*** - Specifies messages that represent error conditions (LOG_ERR), such as an unsuccessful disk write.
- ***warning*** - Specifies messages for abnormal, but recoverable, conditions (LOG_WARNING).
- ***notice*** - Specifies important informational messages (LOG_NOTICE). Messages without a priority designation are mapped into this priority message.
- ***info*** - Specifies informational messages (LOG_INFO). These messages can be discarded, but they are useful in terms of analyzing the system.
- ***debug*** - Specifies debugging messages (LOG_DEBUG). These messages may be discarded.
- ***none*** - Excludes the selected facility. This priority level is only useful if it is preceded by an entry with an * (asterisk) in the same selector field.

The following destinations are available:

- ***filename*** - Write to a file. The parameter is the full path to the file where the data are to be written.
- ***@host_name*** - Sends information to the monitoring tool or the *syslogd* daemon on another system. The parameter is the hostname where specific messages are forwarded. The *syslogd* on this host decides what to do next with this message. The name must be preceded by *@*.
- ***user[, user][...]*** - List of users to whom the message is sent.
- ***** - Message sent to all users.

Rotation keywords:

- ***rotate*** - Indicates the use of rotation. If no further parameters are specified, the default is: *size = 1m*, *files = unlimited*.
- ***size*** - The size in kilobytes (KB) or megabytes (MB) that the file can reach. A file that reaches this value is archived by renaming. The new name is created by adding a period and the next number to the original name. The first archived file is named *filename.0*. The minimum file size that can be specified is 10 KB.
- ***time*** - The time after which the file will rotate. The number given here is the number of hours, days, weeks, months, or years. When the file reaches a certain age, it is archived, and a date is added to its name in the format *YYYYmmmdd: HH: MM: SS*, where:
  - *YYYY* - Year;
  - *mmm* – Three-letter name of the month;
  - *dd* - Day;
  - *HH* - Hour;
  - *MM* - Minute; and
  - *SS* - Second.
- ***files*** - Determines the number of files in the rotation to which the messages are written. If this parameter is not specified, there is no limit to the number of files. The minimum number you can specify is two.
- ***compress*** - Determines whether the archived files are to be compressed.
- ***archive*** - Specifies the directory to which the files are to be copied.

Examples:

- ****.warning;mail.none /dev/console*** - All messages with the warning priority or higher (except those sent by the mail subsystem) are displayed on the system console.
- ***kern.crit /tmp/kernel.log rotate size 1m files 3*** - All messages from the kernel that have a critical or higher priority are to be written to */tmp/kernel.log*. The file should be rotated (it should be archived sequentially under the names *kernel.log.0*, *kernel.log.1*, and *kernel.log.2*) after reaching a size of 1 megabyte.
- ***lpr. * @printsrv*** - All messages from the print subsystem are passed to the *syslogd* daemon running on a server named *printsrv*.

Diag utility

Diag is the main diagnostic tool in AIX. It enables the detailed testing of the selected hardware in order to help diagnose an incorrect operation or detect possible failures. It also allows you to perform many other maintenance activities that are described later in this chapter. Most diagnostic tasks can be performed during system operation. However, in certain situations, the *diag* may require exclusive access to a resource so as to perform a full diagnostic. This is the case if you want to test:

- SCSI adapters, with disks that contain an active paging space;
- disks with an active paging space;
- memory;
- processors, or
- some other devices.

In the case of testing the above-mentioned devices, only a part of their functionality will be tested during the normal operation of the system. For complete testing, you must start the system in service mode (single user) or run the *diag* tool directly from the diagnostic disc. Only with the latter option you can be 100% sure that the test of the selected components is complete.

You can boot the system in service mode using the *shutdown -m* command. It is important to note that network services do not work in this case. Therefore, access to the system is only possible through a directly connected console.

To start the *diag* tool from the diagnostic DVD, change the boot order so that the optical drive is at the beginning. You can do this via the bootlist command, for example, *bootlist -m normal cd0 hdisk0 hdisk1*. After changing the sequence, restart the server, leaving the diagnostic DVD in the drive. AIX and the *diag* tool will be launched from the disc. In this mode, access is given to all devices and any diagnostics can be performed.

The *diag* tool has several parameters that can be used in order to automate actions by providing them on the command line. It frees the user from having to navigate through the various tool menus, since it immediately calls the appropriate function. Sample parameters:

- ***-A*** - Advanced tests.
- ***-B*** - Basic tests.
- ***-c*** - Runs the tool in silent mode. The result is displayed on the standard output.
- ***-d device*** - The device to be tested.
- ***-e*** - Analysis of errors recorded for the device.
- ***-E days*** - The number of days back that *diag* should consider when analyzing the error log.
- ***-S group*** - Tests an individual device group (1 - base system devices, 2 - I/O devices, 3 - asynchronous devices, 4 - graphics devices, 5 - SCSI devices, 6 - data storage devices, 7 - communication devices, and 8 - multimedia devices).
- ***-s*** - Start diagnostics on all devices.
- ***-T task*** - Executes a specific task.

Examples:

```
# diag -d fcs0 -c   # Diagnostics of the fcs0 fiber channel adapter.
Starting diagnostics
Testing fcs0
Ending diagnostics.

# diag -AecS 5   # Advanced SCSI diagnostics, including an error log analysis.
Starting diagnostics
Testing fcs0
Testing fscsi0
Testing fcs1
Testing fscsi1
Ending diagnostics.
```

Diag features

When using the *diag* tool, you typically use the menu rather than using the parameters that are available on the command line. When *diag* is started, the first menu is the function selection menu, which is shown in Figure 11-1.

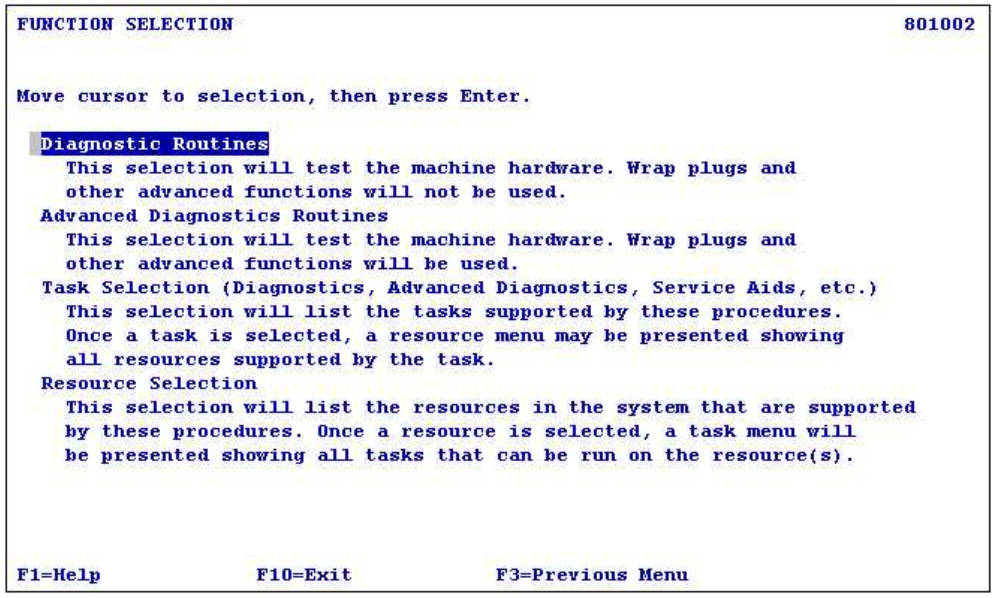

Figure 11-1: Diag utility - Function selection.

There are four possible functions:

- **Diagnostic Routines** - Basic diagnostic test mode. This is convenient for the user, since in this mode he is not forced to interact with the system. The user is not asked during the test procedure to disconnect devices or connect special test equipment. The disadvantage of this mode is the possibility of missing certain hardware faults due to carrying out only basic tests.
- **Advanced Diagnostics Routines** - Advanced test mode. Some tests require the user to connect or disconnect the appropriate device, or to connect the "*wrap plugs*." In this mode,

accurate tests are performed. If the tests are successful, you can be sure that the tested components are working correctly.

- **Task Selection (Diagnostics, Advanced Diagnostics, Service Aids, etc.)** - A section with a lot of tools for performing service tasks.
- **Resource Selection** - This menu provides the same functionality as the *Task Selection* menu, but it is more user friendly. It allows you to first select the device on which you want to operate, and then the action you want to perform. The activity is selected from the list of actions that can be performed on the device. This greatly limits the number of possibilities, thereby facilitating their selection.

Diagnostics

If you choose the mode you want to perform diagnostics for, whether it be basic or advanced, there are two options:

- **System Verification** - A system check without an error log analysis. Use this option after corrective action, or in case there are no errors in the log.
- **Problem Determination** - A system check with an error log analysis. This option is used when an error appears in the error log. In the event of anomalies in the work of any device, *detailed data* are written to the log. They provide additional data for the *diag* utility to analyze.

When you select one of these options, *diag* displays the selection screen of the devices on which you want to run the test. You can select one device or all the devices at once. This is shown in Figure 11-2.

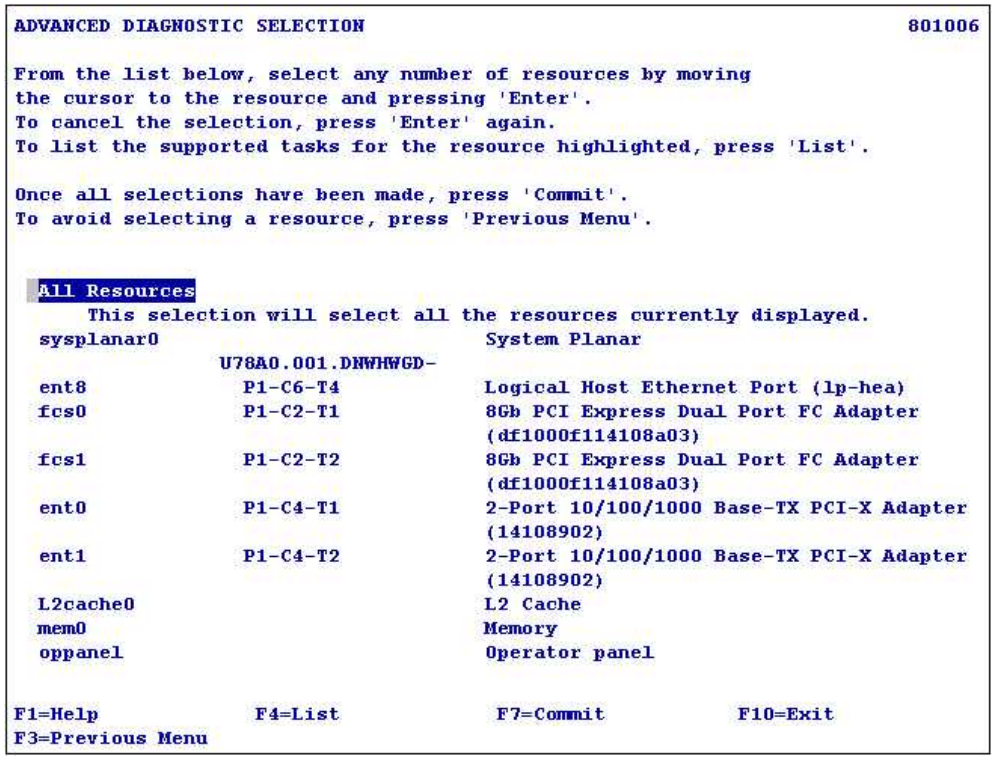

Figure 11-2: Diag - Selection of devices to test.

Additional functions of the diag

The *Task Selection* and *Resource Selection* menus significantly extend the functionality of the *diag* utility. There are a number of available options:

- **Display or Change Bootlist** - Displays or modifies the bootlist from the menu.
- **Microcode Tasks** - Upgrades the device with the new firmware.
- **Hot Plug Task** - A menu that allows you to use hot plug devices. These devices can be added and removed from the server during its normal operation. With these options, you can identify a slot or disk by lighting a diode, removing a device from the server, or adding a new device.
- **Delete Resource from Resource List** - Deletes devices.
- **RAID Array Manager** - Creates, modifies, and removes RAID (if there are the appropriate controllers and disks).

There are several additional features available in this tool. Some of them are very useful, so it is good practice to review them in order to thoroughly understand what you can achieve using this utility.

Chapter 12. Network Management

Network management is a very broad topic. Indeed, I could describe in this chapter a whole series of protocols belonging to different network layers, and many network devices and tools supported by AIX. However, it is most important to know about the basic configuration based on Ethernet and IP, as well as about the basic network services. These topics are hence described in this chapter.

Basic configuration

You can perform the basic configuration using the installation assistant after the operating system installation. You can also do it at any time via the *smit mktcpip* menu. When you start the menu, the first thing to do is to select the interface you want to configure. This is shown in Figure 12-1.

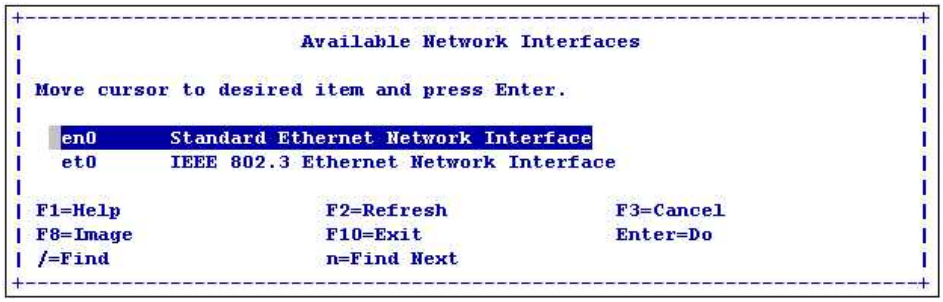

Figure 12-1: Smit mktcpip - Basic network configuration - Interface selection

In the menu above, it is possible to configure two interfaces, while the system has one network adapter. This is because Ethernet adapters can use two types of frames:

- Ethernet II (standard Ethernet).
- IEEE 802.3.

Therefore, the above figure should be interpreted as follows. The *ent0* adapter can be configured to use frames:

- Ethernet II - Device *en0*,
- IEEE 802.3 - Device *et0*.

The frames have almost the same structure, although they differ in terms of how the higher layer protocol is identified. However, this difference makes them incompatible with each other. Systems using different frames are thus unable to communicate with one another.

Ethernet II frames (*en* interfaces) are typically used. When you select the interface, you will see a menu that allows you to configure it, as shown in Figure 12-2.

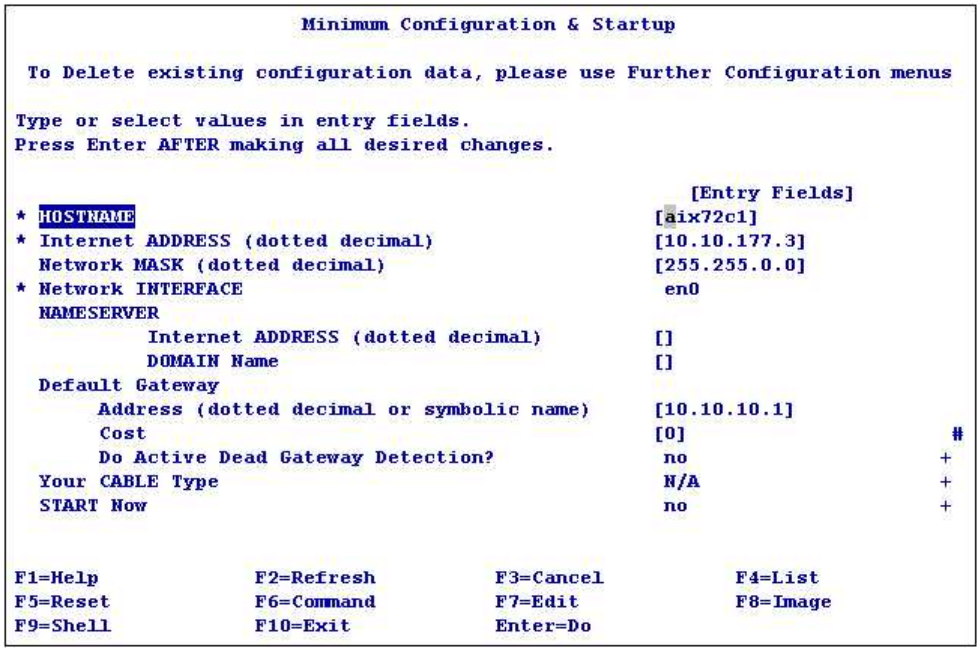

Figure 12-2: smit mktcpip - Basic network configuration.

You can specify configuration data such as the IP address and subnet mask that do not need to be further explained. In addition, you can specify:

- **NAMESERVER: Internet ADDRESS (dotted decimal)** - The name of the DNS server that will be queried in order to map the IP address to a name or a name to the IP address. The value entered here is stored in */etc/resolv.conf*.
- **NAMESERVER: DOMAIN Name** - The domain name of the system, for example, *test.newdomain.com*. The value entered is also stored in */etc/resolv.conf*.
- **Default Gateway: Address (dotted decimal or symbolic name)** - The default gateway address, that is, the IP address of the router responsible for sending packets to systems located outside the current subnet. By sending a packet to a server outside the current subnet, the server searches its routing table so as to find the interface and gateway it should use to send the packet. If there is no detailed entry in the routing table that tells the system where to send the packet, it sends the packet to the router that represents the default gateway. The routing configuration on the network determines which way the packet is routed to the destination.
- **Default Gateway: Cost** - Specifies the cost (priority) that the gateway has. Packets that have several routes to choose from are always routed at a lower cost route.
- **Default Gateway: Do Active Dead Gateway Detection?** - Enables the detection of non-functioning gateways. If it is detected as non-functioning, the cost of sending packets through a gateway increases. In this case, the backup gateway (if there is one) can take over the role of primary gateway. The cost of a backup gateway is usually defined as higher than that of the primary gateway, but if the latter is not working, it the cost decreased. This causes the backup gateway to take over the traffic that was thus far transmitted through the primary gateway.
- **Your CABLE Type** - The type of connector that the interface is connected to via the Ethernet network. This parameter usually remains at the default setting because the system correctly detects its type.

- **START Now** - Determines whether the configuration is to be updated immediately, or whether the changes are to be made in the ODM database and specific files and then activated only after the server is restarted.

After this configuration, the system can communicate with other systems and provide basic network services.

Name resolution

Remembering IP addresses can prove difficult. Therefore, to make life easier for network users, mechanisms have been created that allow them to map IP addresses to more user-friendly names. Instead of using addresses, you can use names, for example, replacing *ping 127.0.0.1* with *ping localhost*. There are several mechanisms for mapping names. The system can use them all in the order you specify (if it does not find a mapping using one mechanism, then it will try another, etc.). The default names mapping order is:

- DNS (*Domain Name System*)/BIND (*Berkeley Internet Name Domain*);
- NIS (*Network Information Service*); and
- Search the */etc/hosts* file.

The above order can be changed by adding the corresponding entry in */etc/netsvc.conf* (this is the global setting for all users) or by setting the *NSORDER* environment variable (this is a local setting for the session). If there is an entry in the */etc/netsvc.conf* file and the NSORDER variable is set, then the system uses the order set by the variable. An example of setting the NSORDER variable follows:

```
# export NSORDER=local,bind,nis
```

The following order is set in the above example:

- **local** - Search the **/etc/hosts** file. If the mapping is not found, then:
- **bind** - Query the DNS server (its address and configuration are in */etc/resolv.conf*). If the mapping is not found, then:
- **nis** - Search for the mapping via the Network Information Service.

The /etc/hosts file

The simplest way of mapping names to IP addresses is through the */etc/hosts* file. This file is found on every AIX server, and it has a local scope (it is only used by the server on which it is located). Using such a mapping in a large environment is very laborious. Any change to the IP address of any system in the network requires manual changes to the */etc/hosts* on all systems. The sample entries in this file look like this:

```
# tail -4 /etc/hosts   # Displays the last four lines of the file.
127.0.0.1               loopback localhost      # loopback (lo0) name/address
::1                     loopback localhost      # IPv6 loopback (lo0) name/address
10.10.177.3   aix72c1    # Associates the name aix72c1 with the address 10.10.177.3.
```

```
10.1.120.30   nas
```

By the way, look at the first line of the displayed file. This is an entry that specifies a loopback interface. When the system sends information through this interface, it actually sends it to itself. This entry should never be changed, since the loopback interface is often used to communicate among the different applications on the same system. You can also use it to validate the TCP/IP protocol stack. If you send a message to the loopback interface and you receive a response, then the protocol stack is working fine. An example of such a test is the *ping localhost* command.

DNS

Another popular and well-known name resolution mechanism is the DNS (Domain Name System). A classic example of using the DNS is the Internet. When you use the Internet, you almost always use a domain name instead of an IP address. The DNS name resolution process looks like this:

1. You refer to a host by its name, for example, *ping server4.organization.com*.
2. The system asks the DNS server which IP address corresponds to the given name *server4.organization.com*.
3. The DNS server checks for mapping. If it does not have the relevant information, it may want to query other DNS servers, depending on your network configuration.
4. The DNS server returns the IP address corresponding to the given name.
5. The operating system changes the given name to an IP address, for example, instead of *ping server4.organization.com*, it calls *ping 10.150.54.58*.

You can use */etc/resolv.conf* to configure your system to use a DNS server. If this file does not exist, the system does not use the DNS. In such a case, you can create a file and then configure it accordingly, so that the system will use the DNS. You can do this via the menu (*smit resolv.conf*) or with any text editor. Sample content of */etc/resolv.conf*:

```
# cat /etc/resolv.conf
nameserver      10.150.54.58
nameserver      10.144.7.21
domain  organization.com
```

Typical entries that may occur in the file:

- ***nameserver IP_address*** - The name server address. There may be several such entries. If that is the case, if one DNS server does not respond, then the next one is queried.
- ***domain domain_name*** - Specifies the default domain name that is added to the incomplete name. For example, if you refer to *server4*, then the system is referring to *server4.organization.com* (*server4.domain_name*). There may be only one entry of this type.
- ***search domain_name1 domain_name2 ..*** - Domain names that are searched when you give an incomplete name. This mechanism works in such a way that when referring to a server named, for example, *server4*, the *server4.domain_name1* will be looked for and, if it is not found, then *server4.domain_name2* will be looked for.

Domain and *search* entries cannot exist simultaneously. If there is a *search* entry, the first domain name from this entry is considered to be the default domain name.

The typical AIX system is a DNS client, although it can also act as a DNS server. The *named* daemon is responsible for this option, which is disabled by default.

NIS

The Network Information Service is a database that maintains specific network configuration files in one place. When you use the NIS, you replace locally maintained files (such as */etc/passwd*, */etc/group*, or */etc/hosts*) with files that are stored centrally on the NIS server.

It is easy to see that the role of the NIS is actually much wider than simply name resolution. Its main purpose is to facilitate the management of multiple servers. With the NIS, for example, you can log on to all machines on the network with one account and password stored in the NIS database. There is also an extended version of this service, which is known as NIS +. It has many additional features and is backwards compatible with the NIS.

This mechanism is currently very rarely used. Over time, it has been replaced by more modern mechanisms, such as the LDAP and DNS.

Activation of network interfaces and services

The activation of AIX's network interfaces and services is reduced to the processing of two scripts, namely */etc/rc.net* and */etc/rc.tcpip*.

/etc/rc.net - This script is executed during the second phase of operating system startup (see the chapter on operating system startup). The script configures the network interfaces on the system. It obtains the configuration data from the ODM database. After processing this script, the interfaces have their addresses and configuration. From now on, it is possible to communicate with other systems on the network.

AIX's network configuration method is not typical of UNIX systems. The *chdev* command is used to configure the network devices (the network devices are, for example, *inet0*, *eth0*, and *en0*). The classic network configuration method uses the *ifconfig* command (BSD style). If you prefer the classic network configuration style, you can activate it using *chdev -l inet0 -a bootup_option=yes* or the *smit setbootup_option* menu shown in Figure 12-3.

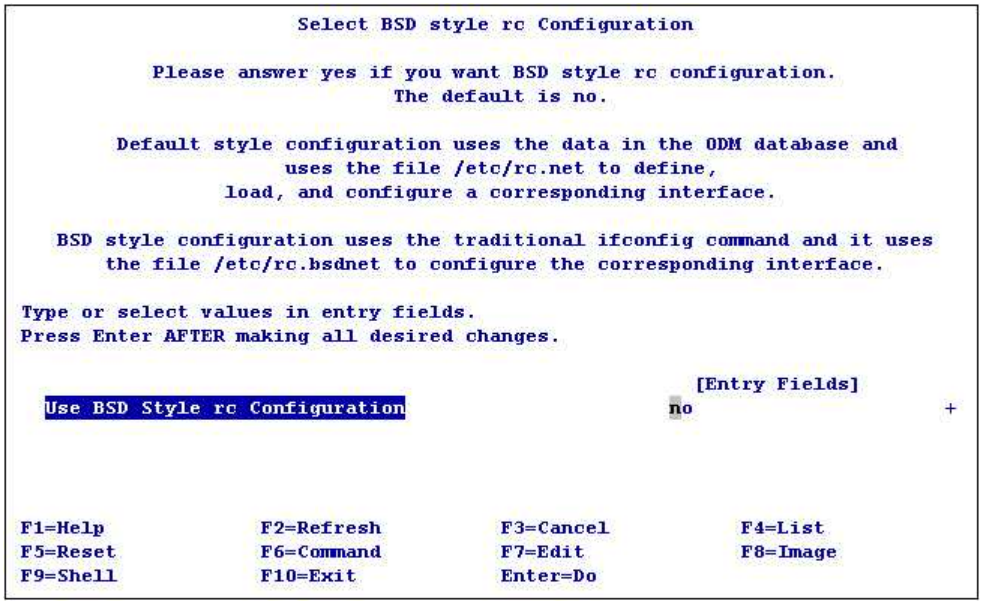

Figure 12-3: SMIT - The choice of network configuration style.

When you select the classic style, instead of */etc/rc.net*, the */etc/rc.bsdnet* file will be processed. The system will perform the entire configuration using the classic UNIX commands without using the ODM database.

/etc/rc.tcpip - This script is executed during the third phase of system startup, while processing the */etc/inittab* file. The following entry in the */etc/inittab* file causes the script to run:

```
# lsitab rctcpip
rctcpip:23456789:wait:/etc/rc.tcpip > /dev/console 2>&1 # Start TCP/IP daemons
```

The task of the */etc/rc.tcpip* script is to launch the selected network services through the SRC (System Resource Controller). By default, this script runs many services, as shown in the following output from AIX 7.2:

```
# grep "start /usr/" /etc/rc.tcpip
#start /usr/sbin/dhcpcd "$src_running"
#start /usr/sbin/autoconf6 ""
#start /usr/sbin/ndpd-host "$src_running"
#start /usr/sbin/ndpd-router "$src_running"
start /usr/sbin/syslogd "$src_running"
#start /usr/sbin/lpd "$src_running"
#start /usr/sbin/routed "$src_running" -q
#start /usr/sbin/gated "$src_running"
start /usr/lib/sendmail "$src_running" "-bd -q${qpi}"
[ -z "$portmap_pid" ] && start /usr/sbin/portmap "${src_running}"
start /usr/sbin/inetd "$src_running"
#start /usr/sbin/named "$src_running"
#start /usr/sbin/timed "$src_running"
#start /usr/sbin/xntpd "$src_running"
#start /usr/sbin/rwhod "$src_running"
start /usr/sbin/snmpd "$src_running"
#start /usr/sbin/dhcpsd "$src_running"
#start /usr/sbin/dhcprd "$src_running"
```

```
#start /usr/sbin/dpid2 "$src_running"
start /usr/sbin/hostmibd "$src_running"
start /usr/sbin/snmpmibd "$src_running"
start /usr/sbin/aixmibd "$src_running"
#start /usr/sbin/mrouted "$src_running"
#start /usr/sbin/pxed "$src_running"
#start /usr/sbin/binld "$src_running"
#start /usr/sbin/netcd "$src_running"
#start /usr/sbin/ptpd "$src_running"
```

Some services that run via */etc/rc.tcpip*:

- ***syslogd*** - Collects and/or forwards messages about specific events in the system.
- ***sendmail*** - Supports email.
- ***portmap*** - Converts RPC (Remote Procedure Call) numbers to IP port numbers. For the system to handle RPC calls, this daemon must be running. Services such as the NFS and NIS use RPC numbers. For their correct operation, it is necessary to run the *portmap* daemon.
- ***inetd*** - Supports many other network services defined in the */etc/inetd.conf* file, including *telnetd*, *ftpd*, *rlogind*, etc.
- ***snmpd, hostmibd, snmpmibd aixmibd, muxatmd*** - Simple Network Management Protocol. The services are responsible for monitoring the network and servers. The daemons are enabled by default, but to be useful, they need to be configured.

Useful network commands

Some basic commands for network configuration and monitoring include:

- *ifconfig*;
- *route*;
- *traceroute*;
- *ping*;
- *netstat*; and
- *entstat*.

ifconfig command

This is a command traditionally used on UNIX systems to configure network interfaces. In AIX, the interfaces are usually configured using its specific configuration command, namely *chdev*. However, it is possible to use the *ifconfig* command to configure the network interfaces. To use this means of configuration, you must set the network configuration style as BSD (*chdev -l inet0 -a bootup_option=yes*).

Although it is possible to make complete interface configurations using this command, it is rarely used for this purpose. The interfaces in AIX are usually configured using the default approach (*chdev* command). The BSD style is intended for those who are used to working with it when using other UNIX systems. The main use of the *ifconfig* command is to read the parameters of the interfaces that exist within the system:

```
# ifconfig -a
en0:
flags=1e084863,14c0<UP,BROADCAST,NOTRAILERS,RUNNING,SIMPLEX,MULTICAST,GROUPRT,64BIT,CHECKSUM_OFFLOAD
(ACTIVE),LARGESEND,CHAIN>
        inet 10.10.177.3 netmask 0xffff0000 broadcast 10.10.255.255
         tcp_sendspace 262144 tcp_recvspace 262144 rfc1323 1
lo0: flags=e08084b,c0<UP,BROADCAST,LOOPBACK,RUNNING,SIMPLEX,MULTICAST,GROUPRT,64BIT,LARGESEND,CHAIN>
        inet 127.0.0.1 netmask 0xff000000 broadcast 127.255.255.255
        inet6 ::1%1/0
         tcp_sendspace 131072 tcp_recvspace 131072 rfc1323 1
```

You can obtain similar information using the *lsattr* command:

```
# lsattr -El en0
alias4                  IPv4 Alias including Subnet Mask              True
alias6                  IPv6 Alias including Prefix Length            True
arp           on        Address Resolution Protocol (ARP)             True
authority               Authorized Users                              True
broadcast               Broadcast Address                             True
monitor       off       Enable/Disable monitor for virtual Ethernet   True
mtu           1500      Maximum IP Packet Size for This Device        True
mtu_bypass    on        Enable/Disable largesend for virtual Ethernet True
netaddr       10.10.177.3 Internet Address                            True
netaddr6                IPv6 Internet Address                         True
netmask       255.255.0.0 Subnet Mask                                 True
prefixlen               Prefix Length for IPv6 Internet Address       True
remmtu        576       Maximum IP Packet Size for REMOTE Networks    True
rfc1323                 Enable/Disable TCP RFC 1323 Window Scaling    True
security      none      Security Level                                True
state         up        Current Interface Status                      True
tcp_mssdflt             Set TCP Maximum Segment Size                  True
tcp_nodelay             Enable/Disable TCP_NODELAY Option             True
tcp_recvspace           Set Socket Buffer Space for Receiving         True
tcp_sendspace           Set Socket Buffer Space for Sending           True
thread        off       Enable/Disable thread attribute               True
```

You can observe something interesting here. By displaying the en0 interface parameters using *ifconfig* and *lsattr*, you see different values for certain parameters, such as *tcp_sendspace*, *tcp_recvspace*, and *tcp_mssdflt*. The difference lies in the fact that the *lsattr -El en0* command shows the parameters defined on this interface, while *ifconfig -a* shows the effective parameters for this interface. The effective parameters are those assigned to the interface (*lsattr*). However, if they are not explicitly set, the default values are assigned to the interface. The values depend on the interface type, and they are presented in Table 12-1.

Table 12-1: Interfaces parameters.

Interface	Speed	MTU	tcp_sendspace	tcp_recvspace	RFC1323
lo0 (loopback)	N/A	16,896	131,072	131,072	1
Ethernet	10 or 100 (mbit)				
Ethernet	1000 (gigabit)	1500	131,072	65,536	1
Ethernet	1000 (gigabit)	9000	262,144	131,072	1
Ethernet	10 GigE	1500	262,144	262,144	1
Ethernet	10 GigE	9000	262,144	262,144	1
EtherChannel	The configuration is based on the speed of all interfaces in the EtherChannel.				
Virtual Ethernet	N/A	any	262,144	262,144	1
InfiniBand	N/A	2044	131,072	131,072	1

You can change the parameters at the interface level by modifying them using the *chdev* or *ifconfig* commands:

```
# ifconfig en0 tcp_recvspace 65536 tcp_sendspace 65536 tcp_nodelay 1
# ifconfig – Changes the parameters temporarily until system restart. The change is not visible at
the lsattr command level.

# chdev -l en0 -a tcp_recvspace=65536 -a tcp_sendspace=65536 -a tcp_nodelay=1
# chdev - Changes the parameters permanently. The change is visible at the lsattr command level.
```

For the slowest interfaces (Ethernet 10–100 Mb), the parameters are not specified in the table. This means that they work by default with parameters defined on the interface itself (*chdev* or *ifconfig*). If the parameters are not defined at the interface level, their values are inherited from the global system settings (*no* command):

```
# no -L tcp_sendspace   # The CUR column represents the current value of the parameter.
--------------------------------------------------------------------------------
NAME                      CUR    DEF    BOOT   MIN    MAX    UNIT           TYPE
     DEPENDENCIES
--------------------------------------------------------------------------------
tcp_sendspace             16K    16K    16K    4K     8E-1   byte             C
     sb_max
--------------------------------------------------------------------------------
```

Route command

This is a traditional command that configures routing in UNIX systems. It manipulates a routing table that is stored in the memory. This means that changes made to the table using route command are lost when the system is restarted.

For the routing configuration, AIX uses the *chdev* command by default. Routing data are stored in the ODM database. The system reads the data from there during the network configuration. Therefore, in order to make permanent changes to the routing (changes that will be retained after the system reboot), you should directly use the *chdev* command or the SMIT menu (*smit route*). For temporary changes, or if you set up a BSD-style network configuration, use the route command. Examples follow:

Adding default routing:

If the system does not know which gateway to use to send a packet, it is sent through the default gateway:

```
# route add 0 10.150.57.1
```

Adding detailed entries to the routing table:

All packets sent to the subnet 10.144.100.0 go through the gateway (router) with the IP address 10.150.57.2:

```
# route add -net 10.140.100.0 10.150.57.2
10.150.57.2 net 10.140.100.0: gateway 10.150.57.2
```

All packets sent to the system 10.130.100.23 go through the gateway (router) with the IP address 10.150.57.3:

```
# route add -host 10.130.100.23 10.150.57.3
10.150.57.3 host 10.130.100.23: gateway 10.150.57.3
```

Executing the above commands creates the following entries in the routing table (*netstat -r* is used to display the routing table, which will be discussed later in this chapter):

```
# netstat -r | egrep "10.150.57|Destination"
Destination        Gateway           Flags    Refs      Use  If   Exp  Groups
10.130.100.23      10.150.57.3       UGH         0          0 en0      -      -
10.140.100/24      10.150.57.2       UG          0          0 en0      -      -
```

Deleting entries from the routing table:

The deletion process is the same as that for creation, only the word *add* is replaced with the word *delete*:

```
# route delete -net 10.140.100.0 10.150.57.2
10.150.57.2 net 10.140.100.0: gateway 10.150.57.2

# route delete –host 10.140.100.23 10.150.57.3
10.150.57.3 host 10.140.100.23: gateway 10.150.57.3
```

Parameters used:

- ***add*** - Add an entry to the routing table.
- ***delete*** - Delete an entry from the routing table.
- ***net*** - Add an entry for the entire subnet.
- ***host*** - Add an entry for a single machine.
- ***0*** - "Zero" indicates the default routing.

Traceroute command

You can use this command to check the route by which packets are sent to the host. Since there is no direct way to check packet routes, this is done using two components: the TTL property of the packet and the ICMP protocol.

- TTL (*Time To Live*) - A mechanism intended to prevent loops in the network. There is a number (TTL) in the IP packet header that specifies how many gateways a given package can go through. If the packet went through a specified number of gateways, it is deleted. At the same time, information about this fact is sent to the system that sent the packet.
- ICMP (*Internet Control Message Protocol*) - A protocol that is part of the TCP/IP protocol stack, which was created for the exchange of control information.

The process for checking the paths that packets travel is as follows. The traceroute command sends IP packets to the specified address, thereby increasing the number of TTLs in subsequent packets. By default, three packets are sent with the same TTL number, then the TTL number is incremented and

the next three packets are sent, and so on. When a packet goes through the gateway, its TTL value is reduced. When the value reaches zero, the packet is removed. This way, every router that is on the way to the destination will receive a packet that needs to be removed. When a router deletes a packet, it informs its source by sending it an information packet. This way, the traceroute command obtains information about all the routers along the route of the packet.

In practice, the route check is as follows:

- A packet with a TTL of 1 is sent. The first router through which it goes reduces its TTL to 0 and then removes it. The router sends an ICMP packet with control information to the system that sent the original packet. The command extracts the relevant information from the ICMP packet and presents it to the user.
- A packet with a TTL of 2 is sent. The first router through which it goes reduces its TTL to 1. The second router through which it goes reduces its TTL to 0 and then removes it. The router sends an ICMP packet with control information to the system that sent the original packet. The command extracts the relevant information from the ICMP packet and presents it to the user.
- And so on until a packet reach the addressee.

An example of the command usage:

```
# traceroute 10.160.34.32
trying to get source for 10.160.34.32
source should be 10.150.57.40
traceroute to 10.160.34.32 (10.160.34.32) from 10.150.57.40 (10.150.57.40), 30 hops max
outgoing MTU = 1500
 1  10.150.57.253 (10.150.57.253)  2 ms  1 ms  1 ms
 2  10.0.4.106 (10.0.4.106)  1 ms  1 ms  1 ms
 3  10.0.0.42 (10.0.0.42)  3 ms  3 ms  3 ms
 4  10.0.2.105 (10.0.2.105)  3 ms  3 ms  3 ms
 5  10.160.10.12 (10.160.10.12)  3 ms  3 ms  3 ms
 6  10.160.34.32 (10.160.34.32)  3 ms  3 ms  3 ms
```

The numbered lines represent the routers on the route the packet travels. The IP address given in each line indicates the address identifying the router. This is not necessarily the address at which you communicate with it (routers have at least several interfaces with different IP addresses, only one of which is its identifier). The three values in milliseconds at the end of each line show the response time of the router to each packet that reached the TTL limit and was thus erased by the router. Three packets with the given TTL are sent by default and hence there are three values.

The values presented by the *traceroute* command allow you to diagnose network problems. In the case of communication problems, they display the latency of the individual routers, and you can see where the route is broken.

Ping command

This is a basic command that examines a network connection with another host, which exists in every operating system. It is based on the ICMP protocol, which is part of the TCP/IP protocol stack.

You can use this command as the first test of network operation. The result indicates whether there is a

network connection, whether it is 100% correct, and how long it takes to get a response to the sent packet.

An example of command usage:

```
# ping 10.150.57.101
PING 10.150.57.101: (10.150.57.101): 56 data bytes
64 bytes from 10.150.57.101: icmp_seq=0 ttl=64 time=1 ms
64 bytes from 10.150.57.101: icmp_seq=1 ttl=64 time=0 ms
64 bytes from 10.150.57.101: icmp_seq=2 ttl=64 time=0 ms
^C
----10.150.57.101 PING Statistics----
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0/0/1 ms
```

By default, ping sends one packet per second, some 64 bytes in size. After receiving a response, the time that has elapsed from sending the packet to receiving the response is displayed. The command works until it is interrupted (*Ctrl + C*). Finally, the statistics of all communication are displayed.

You can modify the behavior of a command using various parameters, for example:

- ***-c number_of_packets*** - Determines the number of packets to send. After sending the specified number of packets, the command terminates its operation.
- ***-f*** - Sends another packet as soon as it receives a response to the previous one. This means that the network is flooded with huge numbers of packets, so root privileges are needed to use this option.
- ***-s packet_size*** - Specifies the size of each packet being sent.

Netstat command

This command displays statistics and certain network configuration parameters. It can display a lot of detailed and advanced network configuration information. Below you will see the most commonly used options.

Displaying a routing table

```
# netstat -r
Routing tables
Destination       Gateway           Flags   Refs     Use  If   Exp  Groups

Route Tree for Protocol Family 2 (Internet):   # IPv4 section.
default           10.10.10.1        UG        1         4 en0      -      -
10.10.0.0         aix72c1           UHSb      0         0 en0      -      -   =>
10.10/16          aix72c1           U         3      1690 en0      -      -
aix72c1           loopback          UGHS      0         1 lo0      -      -
10.10.255.255     aix72c1           UHSb      0         1 en0      -      -
10.130.100.23     10.150.57.3       UGH       0         0 en0      -      -
127/8             loopback          U         1       952 lo0      -      -

Route Tree for Protocol Family 24 (Internet v6):   # IPv6 section.
loopback          loopback          UH        1       106 lo0      -      -
```

The individual lines show the separate entries in the routing table. When the system attempts to communicate with a host to which it cannot find a direct path in the routing table, it tries to contact it via the *default* path.

Meaning of individual columns:

- ***Destination*** - The specific host or subnet to which the path goes.
- ***Gateway*** - The router through which the system sends packets to the destination (the address is in the *Destination* field).
- ***Flags*** - Parameters that describe the route. The most common parameters are:
  - ***U*** - "Up" - The path is active.
  - ***G*** - The path goes through the gateway (router).
  - ***H*** - The path goes to a single system rather than to the whole subnet.
  - ***S*** - The path was added manually.
  - ***b*** - The path represents the broadcast address.
- ***Refs*** - The number of active connections on a given path.
- ***Use*** - The number of packets sent using a given path.
- ***If*** - An interface through which packets are sent on a given path.
- ***Exp*** - The time in minutes after which the path will become inactive.
- ***Groups*** - The list of groups associated with a given path (applies to multicast addressing).

Consider this interpretation of an example line:

```
Destination      Gateway           Flags   Refs     Use  If   Exp Groups
10.140.100/24    10.150.57.2       UG        3       23  en0    -   -
```

All the packets sent to any system on subnet *10.140.100.0* are sent through gateway *10.150.57.2*. Further, */24* is an alternative way to write the subnet mask *255.255.255.0*. The path is active (*U*), and it goes through the gate (*G*). Some *23* packets have been sent since the machine was started or since the counter was reset. Communication on this path is performed through the *en0* interface.

Displaying interface statistics

```
# netstat -i
Name  Mtu   Network     Address            Ipkts Ierrs    Opkts Oerrs  Coll
en0   1500  link#2      0.2.55.76.a.b6      188326     0   164505     0     0
en0   1500  10.150.57   10.150.57.40        188326     0   164505     0     0
lo0   16896 link#1                             98     0      230     0     0
lo0   16896 127         loopback               98     0      230     0     0
lo0   16896 ::1                                98     0      230     0     0
```

This command displays the same interface several times. This is useful because interface can be considered in several cases, and communication through it can be performed using different protocols. As you can see above, the *en0* interface is shown as:

- A physical interface that communicates at the Ethernet level - It communicates on the second layer of the network in the ISO/OSI model. The host is identified by a MAC address (*0.2.55.76.a.b6*).

- A logical interface that communicates at the TCP/IP (IPv4) level - It communicates on the third layer of the network in the ISO/OSI model. Communication at this level is performed via the TCP/IP protocol, and the system is identified by an IP address (*10.150.57.40*).

In the case of *en0*, there would also be an interface that communicates at the TCP/IP (IPv6) level.

Individual columns with potentially unclear meanings:

- ***Mtu (Maximum Transfer Unit)*** - Maximum packet size. In traditional Ethernet networks, the typical packet size is 1500 B. In many networks, it is possible to use so-called jumbo frames, which can carry up to 9 KB of data. The limitation in this case is the fact that all the devices through which the packet goes have to handle this size.
- ***Ipkts*** - Number of received packets.
- ***Ierrs*** - Number of received packets with errors.
- ***Opkts*** - Number of packets sent.
- ***Oerrs*** - Number of errors when sending packets.
- ***Coll*** - Number of collisions. Displaying the number of collisions at the Ethernet level is not supported.

You can reset the interface statistics using the *netstat -Zi* command.

Displaying protocol statistics

Statistics are maintained for each network protocol that runs on the system. They are reset at the request of the user (via the appropriate command) as well as every time the system is restarted. These statistics are especially useful for people who are familiar with how protocols work, although even beginners should be able to interpret certain values. You can display the statistics for the following protocols, which are part of the TCP/IP stack:

- **TCP (Transmission Control Protocol)** - A connection-oriented protocol, which indicates that a connection is established and maintained until the application programs at each end have finished exchanging messages.
- **UDP (User Datagram Protocol)** - A connectionless TCP equivalent.
- **IP (Internet Protocol)** - A network layer protocol, which is also known as IPv4.
- **ICMP (Internet Control Message Protocol)** - A protocol created in order to send control messages across the network.
- **IGMP (Internet Group Management Protocol)** - A protocol used to handle address groups (for multicast addresses).
- **IPv6 (Internet Protocol version 6)** - Another version of the IP protocol. It was created to address the problem of too few addresses being available in the IPv4 pool.
- **ICMPv6** - Another version of the ICMP protocol.

The statistics available for individual protocols are quite extensive. The interpretation of all the parameters requires detailed knowledge of the construction and functioning of the protocols, which is beyond the scope of this book. Nevertheless, it is worth discussing some of them. An example of the IP protocol statistics follows:

```
# netstat -p ip
ip:
        328571 total packets received
        0 bad header checksums
        0 with size smaller than minimum
        0 with data size < data length
        0 with header length < data size
        0 with data length < header length
        0 with bad options
        0 with incorrect version number
        30 fragments received
        0 fragments dropped (dup or out of space)
        0 fragments dropped after timeout
        6 packets reassembled ok
        199652 packets for this host
        128895 packets for unknown/unsupported protocol
        0 packets forwarded
        2 packets not forwardable
        0 redirects sent
        309296 packets sent from this host
        279 packets sent with fabricated ip header
        0 output packets dropped due to no bufs, etc.
        0 output packets discarded due to no route
        150 output datagrams fragmented
        30 fragments created
        144 datagrams that can't be fragmented
        0 IP Multicast packets dropped due to no receiver
        0 successful path MTU discovery cycles
        0 path MTU rediscovery cycles attempted
        0 path MTU discovery no-response estimates
        0 path MTU discovery response timeouts
        0 path MTU discovery decreases detected
        0 path MTU discovery packets sent
        0 path MTU discovery memory allocation failures
        0 ipintrq overflows
        0 with illegal source
        0 packets processed by threads
        0 packets dropped by threads
        65 packets dropped due to the full socket receive buffer
        0 dead gateway detection packets sent
        0 dead gateway detection packet allocation failures
        0 dead gateway detection gateway allocation failures
        0 incoming packets dropped due to MLS filters
        0 packets not sent due to MLS filters
```

Interpretation of selected statistics:

- **total packets received** - Number of received packets of a given protocol.
- **packets sent from this host** - Number of packets sent using a given protocol.
- **output packets dropped due to no bufs, etc.** - Number of packets that were dropped due to the lack of free memory buffers.
- **output packets discarded due to no route** - Number of packets that were dropped due to the lack of a valid path to the destination.
- **output datagrams fragmented** - Number of packets that have been fragmented. Fragmentation occurs when the system tries to send a packet over a link that only supports packets with a maximum size smaller than that of the packet being sent. The MTU (Maximum Transfer Unit) determines the maximum size of the packets that can be transmitted over a given link. This parameter may be different for each entry in the routing table.

- **packets dropped due to the full socket receive buffer** - Number of packets from outside that were dropped due to the overflow of the receive buffer.

Network connections and services

Who connect to your server, from what address, and using what protocol? Which network services are running on the system and on which ports they are listening? You can answer these questions by running the *netstat -a* command. It displays the statistics for all the sockets in the server:

```
# netstat -a
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
tcp        0      0  *.*                    *.*                    CLOSED
tcp4       0      0  *.*                    *.*                    CLOSED
tcp4       0      0  *.daytime              *.*                    LISTEN
tcp        0      0  *.ftp                  *.*                    LISTEN
tcp6       0      0  *.ssh                  *.*                    LISTEN
tcp4       0      0  *.ssh                  *.*                    LISTEN
tcp        0      0  *.telnet               *.*                    LISTEN
tcp4       0      0  *.smtp                 *.*                    LISTEN
tcp4       0      0  *.time                 *.*                    LISTEN
tcp        0      0  *.sunrpc               *.*                    LISTEN
tcp        0      0  *.smux                 *.*                    LISTEN
tcp        0      0  *.exec                 *.*                    LISTEN
tcp        0      0  *.login                *.*                    LISTEN
tcp        0      0  *.shell                *.*                    LISTEN
tcp        0      0  *.klogin               *.*                    LISTEN
tcp4       0      0  *.kshell               *.*                    LISTEN
tcp        0      0  *.rmc                  *.*                    LISTEN
tcp4       0      0  *.writesrv             *.*                    LISTEN
tcp        0      0  *.caa_cfg              *.*                    LISTEN
tcp        0      0  *.filenet-             *.*                    LISTEN
tcp        0      0  *.filenet-             *.*                    LISTEN
tcp        0      0  *.filenet-             *.*                    LISTEN
tcp4       0      0  aix72c1.ssh            10.10.254.241.51623    ESTABLISHED
tcp        0      0  *.clcomd_c             *.*                    LISTEN
udp4       0      0  *.*                    *.*
udp4       0      0  *.daytime              *.*
udp4       0      0  *.time                 *.*
udp        0      0  *.sunrpc               *.*
udp        0      0  *.snmp                 *.*
udp        0      0  *.syslog               *.*
udp4       0      0  *.ntalk                *.*
udp        0      0  *.rmc                  *.*
udp        0      0  *.xmquery              *.*
udp        0      0  *.filenet-             *.*
udp        0      0  *.filenet-             *.*
udp        0      0  *.32775                *.*
udp        0      0  *.33234                *.*
udp        0      0  *.33235                *.*
udp        0      0  *.33236                *.*
# All unnecessary information has been omitted.
```

The individual lines inform both the service that is listening on specific ports (*LISTEN*) and the external connections to the service (*ESTABLISHED*).

Meaning of individual columns:

- ***Proto*** - The ISO/OSI fourth-layer protocol on which the service operates or through which a connection has been established. In the case of TCP/IP networks, there may be only TCP or UDP protocols.
- ***Recv-Q*** - Number of external packets waiting in line for service. The high values that appear in this column may indicate a performance problem with the service.
- ***Send-Q*** - Number of packets waiting in the queue for sending. The high values that appear in this column may indicate a transport problem (e.g., network performance is too low) or a performance problem on the data recipient side.
- ***Local Address*** - Takes different values depending on the meaning of the line:
  - ****.port_number*** - The line that describes the listening service. An asterisk indicates that the service is listening on all network addresses assigned to the server. After the dot is the port number on which the service listens. If the port is described in the */etc/services* file, the name that represents it is displayed instead of the number (e.g., *ssh* instead of *22*, according to the *ssh 22/tcp* line found in the */etc/services* file).
  - ***host_name.port_number*** - The line that describes the established connection. The *host_name* specifies the interface through which the connection was established (the name is taken from */etc/hosts*). This is useful information if the server has several interfaces. After the dot is the port number with which the connection was established.
- ***Foreign Address*** - Takes different values depending on the meaning of the line:
  - ****. **** - The typical record for services that are running on the server and waiting for connections.
  - ***IP_address*** - The line that describes the established connection. The IP address indicates the host from which the connection was established, and after the dot is the source port number. If the port is described in the */etc/services* file, the name that represents it is displayed instead of the number.
- ***(state)*** - The status of the connection.

To achieve a better understanding, let's interpret a few lines:

```
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
tcp4       0      0  *.ssh                  *.*                    LISTEN
tcp4       0   1027  aix72c1.ssh            10.10.254.241.51623    ESTABLISHED
```

The first line indicates that the *ssh* service is waiting for outside connections (*LISTEN* state) to be made using the TCP protocol. The service waits for connections on the port named *ssh*. The port numbers hidden under the names can be read from */etc/services*. In our case, it is port *22* (the line in */etc/services*: *ssh 22/tcp*).

The second line indicates that the system with the IP address *10.10.254.241* has established a connection from port *51623* to our port named *ssh*. The communication is in progress, since there are *1027* items waiting to be sent.

Entstat command

This command allows you to test the second layer of the protocol stack. The classic solution for layer two is Ethernet. The *entstat* command displays information about the interfaces in this layer. It displays a different set of information depending on the type of interface. Other information is displayed for a virtual adapter, etherchannel, shared Ethernet adapter, or physical adapter. The following example demonstrates the use of a command on a physical adapter:

```
# entstat -d ent0
------------------------------------------------------------
ETHERNET STATISTICS (ent0) :
Device Type: 2-Port 10/100/1000 Base-TX PCI-X Adapter (14108902)
Hardware Address: 00:09:6b:6e:60:10
Elapsed Time: 8 days 3 hours 50 minutes 10 seconds

Transmit Statistics:                          Receive Statistics:
--------------------                          -------------------
Packets: 20234483                             Packets: 107076692
Bytes: 11395821897                            Bytes: 12818882120
Interrupts: 0                                 Interrupts: 46231447
Transmit Errors: 0                            Receive Errors: 0
Packets Dropped: 0                            Packets Dropped: 0
                                              Bad Packets: 0
Max Packets on S/W Transmit Queue: 2
S/W Transmit Queue Overflow: 0
Current S/W+H/W Transmit Queue Length: 0

Broadcast Packets: 3189                       Broadcast Packets: 52649139
Multicast Packets: 3996                       Multicast Packets: 31894983
No Carrier Sense: 0                           CRC Errors: 0
DMA Underrun: 0                               DMA Overrun: 0
Lost CTS Errors: 0                            Alignment Errors: 0
Max Collision Errors: 0                       No Resource Errors: 0
Late Collision Errors: 0                      Receive Collision Errors: 0
Deferred: 0                                   Packet Too Short Errors: 0
SQE Test: 0                                   Packet Too Long Errors: 0
Timeout Errors: 0                             Packets Discarded by Adapter: 0
Single Collision Count: 0                     Receiver Start Count: 0
Multiple Collision Count: 0
Current HW Transmit Queue Length: 0

General Statistics:
-------------------
No mbuf Errors: 0
Adapter Reset Count: 0
Adapter Data Rate: 2000
Driver Flags: Up Broadcast Running
        Simplex 64BitSupport ChecksumOffload
        LargeSend DataRateSet

2-Port 10/100/1000 Base-TX PCI-X Adapter (14108902) Specific Statistics:
--------------------------------------------------------------------------
Link Status : Up
Media Speed Selected: Auto negotiation
Media Speed Running: 1000 Mbps Full Duplex
PCI Mode: PCI-X (100-133)
PCI Bus Width: 64-bit
Latency Timer: 144
Cache Line Size: 128
Jumbo Frames: Disabled
```

```
TCP Segmentation Offload: Enabled
TCP Segmentation Offload Packets Transmitted: 2164989
TCP Segmentation Offload Packet Errors: 0
Transmit and Receive Flow Control Status: Enabled
XON Flow Control Packets Transmitted: 0
XON Flow Control Packets Received: 0
XOFF Flow Control Packets Transmitted: 0
XOFF Flow Control Packets Received: 0
Transmit and Receive Flow Control Threshold (High): 49152
Transmit and Receive Flow Control Threshold (Low): 24576
Transmit and Receive Storage Allocation (TX/RX): 8/56
```

The result of the above command is divided into several sections, which describe different aspects of the interface operation. The selected information obtained using this command includes:

- ***Hardware Address*** - The hardware address of the interface. Otherwise known as the MAC Address (Medium Access Control). This is the 48-bit address used in the Ethernet layer.
- ***Receive Statistics: Packets*** - The number of packets received.
- ***Receive Statistics: Bytes*** - The number of bytes received. This value allows you to specify the amount of information that flows into a given interface and, as a result, allows you to evaluate its load.
- ***Receive Statistics: Receive Errors*** - The number of errors during data reception. The constant growth of these values may suggest physical damage to some component within the network.
- ***Receive Statistics: Packets Dropped*** - The number of dropped packets (for example, due to the checksum errors).
- ***Transmit Statistics: Max Packets on S/W Transmit Queue*** - The maximum number of packets waiting in the software queue to be sent. The software queue is located in the operating memory. From this queue, data are sent to the hardware queue, which is the memory buffer in the network interface.
- ***Transmit Statistics: Current S/W+H/W Transmit Queue Length*** - The number of packets in the software and hardware queues.
- ***Transmit Statistics: Current HW Transmit Queue Length*** - The number of packets in the hardware queue waiting to be sent.
- ***Transmit Statistics: No Carrier Sense*** - The number of physical failures of a connection with a switch. Such cases are commonly referred to as "link failures," and they often indicate the failure of the network cable or its disconnection.
- ***Receive Statistics: CRC Errors*** - A frame (packet) checksum error. The received packet is checked by calculating its checksum and then comparing its value with the checksum included in the packet. A checksum error usually occurs when the link quality is poor.
- ***Media Speed Selected: Auto negotiation*** - The setting that, during the activation of the interface, causes it to negotiate the speed and operating mode with the device at the other end of the link. In the case of Ethernet, 10, 100, and 1000 Mbps speeds are negotiated, while the potential operating modes are:
    - ***half duplex*** - Only one device can transmit at a time.
    - ***full duplex*** - Both devices can transmit at any time - Theoretically, this is twice as efficient as the half duplex.
- ***Media Speed Running*** - The speed at which the interface is now running.

Sample information you can obtain from the virtual interface:

```
# entstat -d ent0 | grep Switch   # Virtual switch with which the adapter works.
Switch ID: ETHERNET0   # You can create multiple virtual switches on the server.

# entstat -d ent0 | grep VLAN   # VLANs supported by this adapter.
Invalid VLAN ID Packets: 0
Port VLAN ID:     1   # The virtual adapter is in VLAN 1.
VLAN Tag IDs:  None   # A trunk adapter can support multiple VLANs.
```

Basic network services - inetd

A typical UNIX system provides many network services. If the system continued to run a separate process for each of these services, then there would be enough processes running to have a negative impact on system performance. Therefore, there is a mechanism that assigns a large portion of these processes to the management of one daemon, namely *inetd*.

The role of *inetd* is to manage other network services (processes). Thanks to it, some network services working on the system do not need to be run permanently. The *inetd* daemon listens to the ports of each service that it supports. When one of these services is called, the inetd daemon launches it and provides a service handler. This service handles the call and ends its work. Due to this, the memory and processor are not loaded with many very small processes. They are only run if they have a job to do.

At startup, the *inetd* process reads its configuration from */etc/inetd.conf*. It can also read its configuration from the file specified by the parameter when it is started. The configuration file describes the services that are to be serviced by *inetd*. A fragment of the */etc/inetd.conf* file follows:

```
## service  socket  protocol  wait/  user    server    server program
##  name     type             nowait         program     arguments
##
ftp     stream  tcp6    nowait  root    /usr/sbin/ftpd         ftpd
telnet  stream  tcp6    nowait  root    /usr/sbin/telnetd      telnetd -a
shell   stream  tcp6    nowait  root    /usr/sbin/rshd         rshd
kshell  stream  tcp     nowait  root    /usr/sbin/krshd        krshd
login   stream  tcp6    nowait  root    /usr/sbin/rlogind      rlogind
klogin  stream  tcp     nowait  root    /usr/sbin/krlogind     krlogind
exec    stream  tcp6    nowait  root    /usr/sbin/rexecd       rexecd
#comsat dgram   udp     wait    root    /usr/sbin/comsat       comsat
#uucp   stream  tcp     nowait  root    /usr/sbin/uucpd        uucpd
#bootps dgram   udp     wait    root    /usr/sbin/bootpd       bootpd /etc/bootptab
#finger stream  tcp     nowait  nobody  /usr/sbin/fingerd     fingerd
#systat stream  tcp     nowait  nobody  /usr/bin/ps           ps -ef
#netstat stream tcp     nowait  nobody  /usr/bin/netstat      netstat -f inet
#tftp    dgram  udp6    SRC     nobody  /usr/sbin/tftpd         tftpd -n
```

Meaning of individual columns:

- ***service name*** - The port number represented by the name in the */etc/services* file. Each request that arrives at the port number listed here will be handled in the manner described in the following part of the line.
- ***socket type*** - The type of socket used by the service. The available types are *stream*, *dgram*, *sunrpc_tcp*, and *sunrpc_udp*.
- ***protocol*** - The type of protocol used by the service (TCP, UDP).
- ***wait/new/SRC*** - Specifies whether inetd should wait for port release before it continues

listening on this port. The *wait*, *new*, or *SRC* setting is for stream sockets, while *nowait* is used for the *dgram* socket. *SRC* works in the same way as *wait*, using the two subsystems (*startsrc*, *stopstrc*).

- ***user*** - Specifies the user who owns the service (process) after it is started. This specifies the system privileges that the service will have.
- ***server program*** - The full path to the program (service) that handles the calls on that port.
- ***server program arguments*** - The name of the program to be run, along with its flags.

As an example, consider the *inetd* daemon in relation to the line that specifies the *ftp* (File Transfer Protocol) service:

```
ftp     stream  tcp6    nowait  root    /usr/sbin/ftpd          ftpd
```

1. The *ftp* client tries to connect to the service on the system. It tries to do so by communicating on port *21* (the default port where *ftp* operates).
2. The system receives the request and directs it to the *inetd* daemon.
3. *Inetd* checks which program supports this port. This is an *ftp* service (see the configuration file */etc/inetd.conf*).
4. */usr/sbin/ftpd* is started and takes care of the request.
5. After processing the request (disconnecting the customer), the *ftpd* process terminates.

When you make any changes to the */etc/inetd.conf* file, you must instruct *inetd* to update its configuration. You can do so in two ways:

1. The traditional way, available on all UNIX systems:

```
# kill -HUP process_number
```

2. The AIX-specific way:

```
# refresh -s inetd
```

Services controlled by inetd

When you look at */etc/inetd.conf*, you can see that the inetd daemon can control dozens of different services. A large part of these services are the historical services of UNIX systems that are often blocked on servers due to the high security risks posed in relation to their use. The high level of risk is due to the open, unsecured communication that they use. It can be easily eavesdropped on and, in some cases, allows a hacker to easily authenticate his system as a legitimate host on the network. In this simple way, anyone can gain unauthorized access to the server.

Table 12-2: Typical services supported by inetd.

Service	Program	Function
ftp	*/usr/sbin/ftpd*	File Transfer Protocol. It transfers files over the network. The service is considered unsafe because the user's name, password, and data are sent in an unencrypted form.
telnet	*/usr/sbin/telnetd*	Allows remote operations on the system. The service is considered unsafe because the user's name, password, and data are sent in an unencrypted form.
shell	*/usr/sbin/rshd*	Remote Shell. When you set up this service, you can run a command that will be executed on another system. For example, *rsh server1 df* runs the *df* command on *server1*. The service is considered unsafe.
login	*/usr/sbin/rlogind*	Remote Login. When you set up this service, you can switch your terminal to another operating system. In other words, you can log on to another system from the command line of the current system. An example of such an action: *rlogin server1*. The service is considered unsafe.
exec	*/usr/sbin/rexecd*	Remote Execution. If there are corresponding entries in the *$HOME/.netrc* file, the service allows you to remotely execute the program without entering your login and password. The service is considered unsafe.
comsat	*/usr/sbin/comsat*	Notifies users of incoming e-mail.
uucp	*/usr/sbin/uucpd*	Copies files between different systems (User-to-User Copy).
bootps	*/usr/sbin/bootpd*	A service for configuring and starting a system over a network. It is necessary for the Network Installation Management (NIM) and the remote booting of systems.
finger	*/usr/sbin/fingerd*	Provides system and user information.
systat	*/usr/bin/ps*	Allows you to check the status of running processes from another system (*ps -ef* command).
netstat	*/usr/bin/netstat*	Allows another system to check the active connections of this system (*netstat -f inet* command).
tftp	*/usr/sbin/tftpd*	Trivial File Transfer Protocol. This is equivalent to the FTP, although there are no user and login concepts in this service. If your system has access to the files provided by *tftp*, then any user defined on that system has access to it. The service is used to download files by diskless clients as well as for installation via the NIM (Network Installation Management).
talk	*/usr/sbin/talkd*	Allows an interactive conversation between two users on the network using the talk command.
rstatd	*/usr/sbin/rpc.rstatd*	Provides other systems with performance statistics derived from the kernel.
rwalld	*/usr/lib/netsvc/rwall /rpc.rwalld*	Allows the sending (and receiving) of messages to all users on the network (global version of the *wall* command).
sprayd	*/usr/lib/netsvc/spray /rpc.sprayd*	A service that allows you to test the performance of a network connection with the system on which it is running. For performance testing, use the *spray* command.
echo	Served internally by	Service for testing. Returns packets to the sender.

	inetd	
discard	Served internally by *inetd*	Service for testing. Removes the received packets. The network equivalent of the */dev/null* device.
chargen	Served internally by *inetd*	Service for testing (character generator). Responds to the sender by sending a random string of characters.
daytime	Served internally by *inetd*	Service for testing. Responds by sending a packet that contains the current date and time.
caa_cfg	*/usr/sbin/clusterconf*	The service enables the creation of clusters in AIX. Part of the Cluster Aware AIX subsystem.

Telnet

The *Telnet* service allows you to log on to the system from another computer on your network. You can work with the system in the same way as you would if logged in locally. When you log in, you gain access to the command line. For all UNIX systems, this interface allows you to perform all your administrative tasks. You can grant or revoke permissions to log in using Telnet via the *smit chuser* menu and the "*User can LOGIN REMOTELY(rsh,tn,rlogin)?*" option.

The Telnet service is supported by *inetd*. This means that the appropriate process is only run for the time needed to handle the request. The process that is started is */usr/sbin/telnetd*.

It is important to note that connecting to this service poses a security risk. When making a connection, the username and password are sent as plain text. All subsequent transmissions between the machines are also not secure. Due to this lack of security, Telnet is rarely used, and one of the first things the administrator must do after installing the system is to disable the Telnet service. Instead of using Telnet, use SSH. It provides fully encrypted transmission, so it has effectively replaced the Telnet protocol.

FTP - File Transfer Protocol

The FTP service (which is based on the FTP protocol) allows other systems to connect to the server and get or put files. The service requires logging in to the system. You do so in the same way as with Telnet.

The service is considered unsafe because both the username and password are sent as plain text. The transferred files are also not secured in any way. Therefore, it is usually replaced by the *Secure File Transfer Protocol (SFTP)* or *Secure Copy (SCP)*. Both of these protocols are based on *SSH (Secure Shell)*. The entire logon process and transmission using these protocols are encrypted.

The FTP service is supported by *inetd*. This means that the appropriate process is only run for the time needed to handle the request. The utilized process is *ftpd* (*/usr/sbin/ftpd*).

All users who have an account in the system can use the FTP service by default. To revoke this option for a particular user, enter his name in the */etc/ftpusers* file. This file contains a list of those users who are unable to log in to the system via FTP. Example file:

```
# cat ftpusers
root
stud01
```

More parameters for this service can be defined in */etc/ftpaccess.ctl*. This file allows you to specify:

- The IP addresses from which you can connect via FTP;
- The directories that will be accessible after logging in; and
- A few other details.

The lines in this file are structured as "keyword: value, value, ..." The following keywords are available:

- ***allow: server1, server2*** - If there is a line starting with *allow*, only the systems listed in that line have access to the server via FTP.
- ***deny: server1, server2*** - If there is a line beginning with *deny* and there is no line starting with allow, then all machines on the network except those listed in the *deny* line have access to the server via FTP.
- ***readonly: /etc, /var, /usr*** - Restricts access to the listed directories, allows only read.
- ***writeonly: /files*** - Restricts access to the listed directories, allows only write.
- ***readwrite: /tmp*** - Grants full access to the listed directories. If there is such a line, then a user who logs on via FTP only has access to the directories listed in the lines: *readwrite*, *readonly*, and *writeonly*. Otherwise, the user has full access (of course, restricted by user rights) to all the directories not mentioned in the *readonly* and *readwrite* lines.
- ***useronly: anonym, guest*** - Defines the anonymous users. A user defined this way, after logging in, only has access to his home directory.
- ***grouponly: anonyms*** - Defines groups of anonymous users.
- ***herald: /etc /info*** - Contains the full path to the message that is displayed when a user attempts to log in.
- ***motd: on*** - The Message of the Day can take an *on* or *off* value. This means that when you log in, you will see a message from the *$HOME/motd* file. An anonymous user will see a message from the */etc/motd* file.

SSH - Secure Shell

The SSH protocol has the same capabilities as Telnet, except that all the transmissions made using it are encrypted. The same level of security applies to the login process and the associated username and password. It has extensions that allow you to transfer files. These extensions are the SFTP (Secure File Transfer Protocol) and SCP (Secure Copy). Due to these functionalities, the SSH protocol has been able to replace the unsecured Telnet and FTP protocols.

The principle behind this protocol is based on RSA cryptographic technology (this name is an acronym formed from the first letters of the names of its creators) and is as follows:

Each system uses two keys:

- **private** - Available only to the local host. Data encrypted with this key can only be decrypted using a public key.
- **public** - Accessible to all users and systems on the network. The data encrypted with this key can only be decrypted using a private key.

It is important to note that even though the public key is well known, users are not able to decrypt the encrypted messages (even if they have encrypted those messages themselves).

Establishing a connection through SSH proceeds are follows:

1. The client connects to the server and receives its public key. If the client connected to this server earlier, it already has the server's public key. In this case, it compares the received key with the one stored in the internal database. If a key mismatch is detected, a special warning is displayed to terminate the connection.
2. The client transfers its public key to the server.
3. The client generates a random value (Session Key), encrypts it, and passes it to the server. This key is used to encrypt further transmissions. Once the connection is established, the transmissions are encrypted with a symmetric key, which provides better performance.

Installation

In early versions of AIX, the SSH software was not included with the operating system. It required a separate installation after downloading from an external source. Patching was an additional problem in this case, since the administrator had to separately patch the operating system and SSH software. In modern versions of AIX, SSH is integrated with the operating system and the above issues are outdated.

When installing the operating system, it is worth remembering that you should select the OpenSSH software installation in the *Install Options*. In AIX 7.2, this option is not checked by default, which usually means that you will have to install SSH later. Some of the *Install Options* are shown in Figure 12-4.

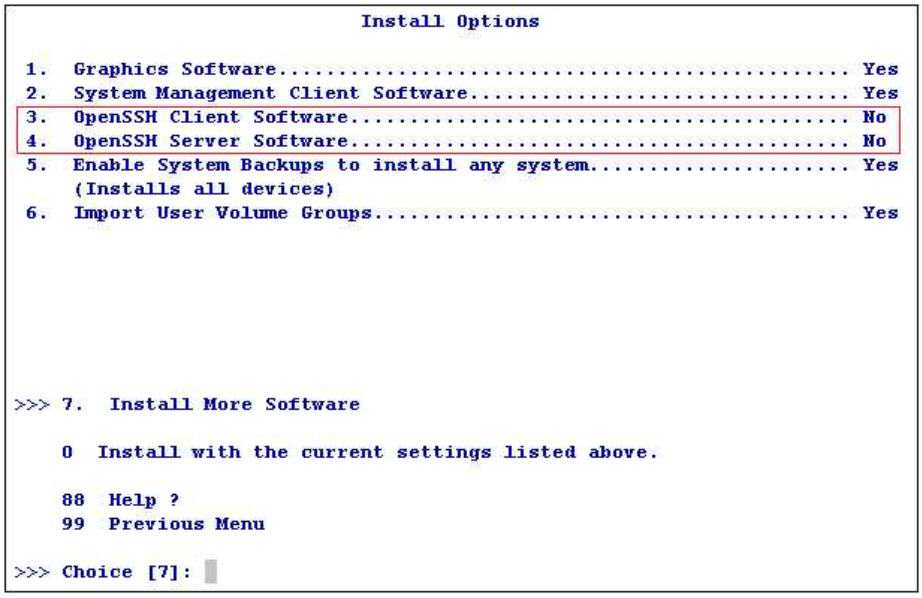

Figure 12-4: SSH - Installation.

The installed SSH service consists of two filesets, namely *openssh.base.server* and *openssh.base.client*, which contain server and client software, respectively:

```
# lslpp -f openssh.base.server   # Displays the SSH server components.
  Fileset               File
  ----------------------------------------------------------------------------
Path: /usr/lib/objrepos
  openssh.base.server 6.0.0.6201
                        /usr/sbin/ssh-pkcs11-helper
                        /usr/sbin/sshd   # SSH server program.
                        /usr/sbin/sftp-server

Path: /etc/objrepos
  openssh.base.server 6.0.0.6201
                        /etc/ssh/ssh_host_ecdsa_key
                        /etc/ssh/sshd_config   # SSH configuration file.
                        /etc/ssh/ssh_host_key
                        /etc/rc.d/rc2.d/Sshd   # Service start script.
                        /etc/ssh/ssh_host_dsa_key.pub
                        /etc/ssh/ssh_host_ecdsa_key.pub
                        /etc/ssh/ssh_host_rsa_key
                        /var/empty
                        /etc/rc.d/rc2.d/Ksshd   # Service stop script.
                        /etc/ssh/ssh_host_key.pub
                        /etc/ssh/ssh_host_rsa_key.pub
                        /etc/ssh/ssh_host_dsa_key
                        /etc/ssh

# lslpp -f openssh.base.client   # Displays the SSH client components.
  Fileset               File
  ----------------------------------------------------------------------------
Path: /usr/lib/objrepos
  openssh.base.client 6.0.0.6201
                        /usr/bin/ssh-keyscan
                        /usr/bin/scp   # SCP command - Copy files.
                        /usr/bin/sftp   # SFTP command - Secure FTP.
```

```
                        /usr/bin/ssh    # SSH command - Connection with other hosts.
                        /usr/openssh/README
                        /usr/sbin/ssh-keysign
                        /usr/bin/ssh-add
                        /usr/openssh/license_ssh.txt
                        /usr/bin/ssh-keygen    # SSH key generator.
                        /usr/swlag/en_US/sshnotices.txt
                        /usr/bin/ssh-agent
                        /usr/openssh

Path: /etc/objrepos
  openssh.base.client 6.0.0.6201
                        /etc/ssh/moduli
                        /etc/ssh/ssh_config
                        /etc/ssh
```

When installing the *openssh* software, the system performs a few additional actions.

- System creates scripts that launch the *sshd* subsystem during system startup (*/etc/rc.d/rc2.d/Ssshd*) and close it during system shutdown or restart (*/etc/rc.d/rc2.d/Ksshd*).

- System creates a user named *sshd* and a group named *sshd*.

  ```
  # grep sshd /etc/passwd /etc/group
  /etc/passwd:sshd:*:203:202::/var/empty:/usr/bin/ksh    # User.
  /etc/group:staff:!:1:ipsec,sshd
  /etc/group:sshd:!:202:sshd    # Group.
  ```

- System creates a subsystem named *sshd* that facilitates the management of the SSH server. You can check the status of this subsystem using the *lssrc* command:

  ```
  # lssrc -s sshd
  Subsystem         Group            PID          Status
  sshd              ssh              5046606      active
  ```

Thanks to the existence of a subsystem for the SSH service, you can more easily enable or disable it (*startsrc -s sshd*, *stopsrc -s sshd*). However, refreshing the configuration of the *sshd* subsystem using the *refresh* command does not work. To refresh (load) the configuration, you must instead use signal communication (*kill* command).

```
# refresh -s sshd
0513-005 The Subsystem, sshd, only supports signal communication.

# ps -ef | grep /usr/sbin/sshd
    root  5046606  3866948   0 04:12:56      -  0:00 /usr/sbin/sshd

# kill -HUP 5046606
```

After installation, every time the server is started, the *sshd* service is automatically started with the default parameters.

Configuration

Once installed, the SSH service should work and allow connections. However, for security reasons, you can generate new encryption keys on the server, with the length mandated by your organization's standards. It is best to generate new RSA and DSA keys immediately. To generate these keys, use the */usr/bin/ssh-keygen* command:

Generating an RSA key pair:

```
# ssh-keygen -t rsa -b 2048
Generating public/private rsa key pair.
Enter file in which to save the key (//.ssh/id_rsa):
Created directory '//.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in //.ssh/id_rsa.
Your public key has been saved in //.ssh/id_rsa.pub.
The key fingerprint is:
c9:4b:06:27:63:78:b8:f0:5f:45:d5:cc:db:75:4c:9e root@aix72c1
The key's randomart image is:
+--[ RSA 2048]----+
|           ...+ o.|
|     o   .    +.=|
|  . o * . .    E+|
|   o + * o    . .|
|    o   S        |
|     . + .       |
|      . .        |
|                 |
|                 |
+-----------------+
```

Generating a DSA key pair:

```
# ssh-keygen -t dsa -b 1024
Generating public/private dsa key pair.
Enter file in which to save the key (//.ssh/id_dsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in //.ssh/id_dsa.
Your public key has been saved in //.ssh/id_dsa.pub.
The key fingerprint is:
a8:81:e1:14:32:bd:76:d5:c7:a6:53:98:d6:67:15:49 root@aix72c1
The key's randomart image is:
+--[ DSA 1024]----+
|o..    . =   oEo |
| o..  . = * o .  |
|  o. . . = o     |
| ooo.  .o        |
| .o.. . S.       |
|     o           |
|    .            |
|                 |
|                 |
+-----------------+
```

In each case, a pair of keys is generated: private and public. The parameters used in the examples above are:

- ***-t*** - Specifies the type of key (*rsa*, *dsa*); and
- ***-b*** - Specifies how long the key should be. By default, a 1024-bit key is created. When determining the length of a key, keep in mind that the longer the key, the greater the security, although the more computing power is wasted on encryption.

The SSH service reads the parameters it runs on from the */etc/ssh/sshd_config* file. Immediately after installation, all the parameters are disabled (preceded by a hash #), which means that the service overrides them with default values. Modifying these parameters involves removing the # sign and entering the desired value.

Below is a description of the meaning of certain parameters and the accompanying values:

- ***Port 22*** - The port on which the SSH service is listening.
- ***ListenAddress*** - The IP address, and hence the interface on which the *sshd* daemon is to listen. By default, it listens on all interfaces, although it can be limited to one or some interfaces.
- ***HostKey*** - Paths to the files that contain the keys used for encryption.
- ***KeyRegenerationInterval 1h*** - The period after which the key to encrypt the session is recreated. This is true for the SSH version 1 protocol.
- ***ServerKeyBits 1024*** - Determines the key strength. The more bits you use, the harder the key is to break. Unfortunately, the same dependency applies to the server load when encrypting and decrypting, that is, the larger the key, the more computing power the server uses.
- ***SyslogFacility AUTH*** - Specifies the *syslogd* message facility to which messages will be reported by *sshd*. AUTH indicates that errors will be reported to the facility related to security and authorization. For more information on the *syslogd* daemon, see the problem determination chapter.
- ***LogLevel INFO*** - Specifies the priority with which the above-mentioned errors will be reported. INFO means informational.
- ***LoginGraceTime 2m*** - The time that the user has to log in. It is counted from the time of establishing the connection (i.e., from the time the "login" window appears). If the user does not log in within 2 minutes, he will be disconnected.
- ***PermitRootLogin yes*** - Specifies whether the *root* user has the right to log in directly using this service. Normally, this parameter is set to *no*. To log in as *root*, you must first log in as a regular user, then issue the *su* command to log in to the *root*.
- ***StrictModes yes*** - Specifies whether *sshd* should check permissions for a user's home directory before logging in. The user's home directory cannot be writable to other users.
- ***RSAAuthentication yes*** - Determines whether RSA authentication is allowed. Applies to SSH version 1.
- ***PubkeyAuthentication yes*** - Determines whether public key authentication is allowed. Applies to only SSH version 2.
- ***AuthorizedKeysFile .ssh/authorized_keys*** - Indicates the file containing the public keys for user authentication. The parameter can be given as an absolute path (starting with /) or a relative path (starting with .)
- ***RhostsRSAuthentication no*** - Specifies whether authentication using *rhosts* or */etc/hosts.equiv* is to be used simultaneously with RSA authentication. This option only applies to the version 1 protocol.

- ***HostbasedAuthentication no*** - Specifies whether to authenticate using *rhosts* or */etc/hosts.equiv* at the same time as using public key authentication. This applies to the version 2 protocol.
- ***IgnoreUserKnownHosts no*** - Specifies whether *sshd* should ignore the *$HOME/.ssh/known_hosts* file during *RhostsRSAAuthentication* or *HostbasedAuthentication*.
- ***IgnoreRhosts yes*** - Determines whether the *.rhosts* and *.shosts* files are to be used during *RhostsRSAAuthentication* or *HostbasedAuthentication*.
- ***PermitEmptyPasswords no*** - Allows (or forbids) users with "blank" passwords to log in.
- ***PrintMotd yes*** - Specifies whether the system should display the contents of the */etc/motd* (Message of the Day) when the user logs in.
- ***PrintLastLog yes*** - Specifies whether, at the time a user logs on, *sshd* should print the date and time of the user's last login to the system on the screen.
- ***TCPKeepAlive yes*** - Determines whether the system should check (sending a packet from time to time) whether the established connection still exists. Such checking prevents the existence of open sessions that have been broken that *sshd* is not aware of.
- ***UsePrivilegeSeparation yes*** - Setting this parameter means that a process with the same privileges as the logged in user will be created to support the user's session. This increases the security of the system.
- ***PermitUserEnvironment no*** - Determines whether the *~/.ssh/environment* file and the *environment=* from the *~/.ssh/authorized_keys* option are to be processed by *sshd*. Enabling this parameter carries some security risks, which is why it is disabled by default.
- ***Compression yes*** - Determines whether compression is allowed (this is decided by the client application).
- ***ClientAliveInterval 0*** - The time after which the connection with the client is checked. The check is performed by sending a message through an encrypted channel. The default setting (*0*) disables this mechanism. This option applies to the version 2 protocol.
- ***ClientAliveCountMax 3*** - Upon reaching the number of sent messages that has been specified here, the connection to the client is dropped.
- ***UseDNS yes*** - Determines whether *sshd* is to use the DNS server, thereby translating IP addresses into names and names into IP addresses.
- ***PidFile /var/run/sshd.pid*** - Specifies the file in which the *sshd* process ID is to be stored.
- ***MaxStartups 10*** - The maximum number of simultaneous unauthorized connections to the SSH service. After obtaining this number, each subsequent connection is dropped. The connection is unauthorized from the moment the system asks you to enter your username until you provide the correct password. After this, the connection is authorized.
- ***Banner /some/path*** - Indicates the message that the system displays during login. The message is displayed when you enter your username.
- ***Subsystem sftp /usr/sbin/sftp-server*** - Defines the external SSH subsystem. In this case, it is an *sftp* subsystem for secure file transfer (a secure *ftp* solution).

NFS (Network File System)

The Network File System is a UNIX standard for sharing files over the network. It was created in the eighties by Sun Microsystems. Initially, it represented a way for a diskless workstation to access the files located on the server.

The NFS allows you to work with files located on another machine in the same way as with local files. From the system administrator perspective, NFS activity is concerned with sharing certain server directories and their contents. This process is commonly referred to as export. The client and the NFS server do not need to be compatible at either the hardware or operating system level.

There is software installed by default in AIX that allows the server to work as a client and as an NFS server. AIX supports NFS versions 2, 3, and 4. The following sections will discuss how NFS versions 2 and 3 works, and then the improvements that NFS version 4 has introduced.

Daemons

The NFS is started via the */etc/rc.nfs script*, which initiates the processes needed for its operation. This script, in turn, is started while the system boots, by an entry in the */etc/inittab* file:

```
# lsitab rcnfs
rcnfs:23456789:wait:/etc/rc.nfs > /dev/console 2>&1 # Start NFS Daemons
```

The operation of the NFS on AIX requires the cooperation of several daemons. One group is needed to operate as the NFS server, while the other is needed to operate as the client. This situation is presented in Table 12-3.

Table 12-3: Daemons of the NFS server and NFS client.

Daemon	Server	Client
portmap	YES	YES
nfsd	YES	NO
rpc.mountd	YES	NO
rpc.statd	YES	YES
rpc.lockd	YES	YES
biod	NO	YES

The roles of individual daemons:

- ***portmap*** - Converts the RPC (Remote Procedure Call) identification numbers to the corresponding port numbers (TCP or UDP). The portmap must be running for RPC-based services to work (the NFS is such a service).

  RPC-compliant programs have their own unique identification numbers. Upon startup, they must register with the *portmap* daemon, since no RPC service can operate without registration. Therefore, it is important that the *portmap* launches prior to the RPC services, and that

restarting the *portmap* daemon requires restarting all the services that use it. You can check the registered services using the *rpcinfo* command:

```
# rpcinfo -p
   program vers proto   port  service
    100000    4   udp    111  portmapper
    100000    3   udp    111  portmapper
    100000    2   udp    111  portmapper
    100000    4   tcp    111  portmapper
    100000    3   tcp    111  portmapper
    100000    2   tcp    111  portmapper
    100021    1   udp  32775  nlockmgr
    100021    2   udp  32775  nlockmgr
    100021    3   udp  32775  nlockmgr
    100021    4   udp  32775  nlockmgr
    100021    1   tcp  32769  nlockmgr
    100021    2   tcp  32769  nlockmgr
    100021    3   tcp  32769  nlockmgr
    100021    4   tcp  32769  nlockmgr
    100024    1   tcp  32770  status
    100024    1   udp  32773  status
    100133    1   tcp  32770
    100133    1   udp  32773
    200001    1   tcp  32770
    200001    1   udp  32773
    200001    2   tcp  32770
    200001    2   udp  32773
    100003    2   udp   2049  nfs
    100003    3   udp   2049  nfs
    100003    2   tcp   2049  nfs
    100003    3   tcp   2049  nfs
    100003    4   tcp   2049  nfs
    200006    1   udp   2049
    200006    4   udp   2049
    200006    1   tcp   2049
    200006    4   tcp   2049
    100005    1   tcp  32789  mountd
    100005    2   tcp  32789  mountd
    100005    3   tcp  32789  mountd
    100005    1   udp  44549  mountd
    100005    2   udp  44549  mountd
    100005    3   udp  44549  mountd
    400005    1   udp  44550
```

A client who wants to run an RPC call for a program must first contact the server-side *portmap* daemon. He obtains the port number to which he should connect.

- ***nfsd*** - A server-side daemon that supports client requests. It has a multi-threaded design. One *nfsd* thread can handle only one read or write job at a time. By default, the maximum number of threads for this daemon is eight. This means that eight jobs can be processed simultaneously. If the number of simultaneous jobs exceeds eight, then some of them will have to wait in the queue for service.
- ***rpc.mountd*** - A daemon that handles the mounting process of a shared resource and provides information about the exported (shared) directories. It reads the information of the shared directory from the */etc/xtab* file. You can use the *showmount -e ServerIPaddress* command to find out which directories have been exported on a given server.
- ***rpc.statd*** - Works with *rpc.lockd* in the area of connection recovery and NFS service recovery after a failure. It should always be run before *lockd*. This daemon stores information about established connections and their status. You can find this information in the */var/statmon/sm*

directory as well as in the */var/statmon/sm.bak* and */var/statmon/state* files. After restart, the daemon checks the above files and tries to restore the connections that were active while daemon was stopped. If you do not want to recover the connections, delete the above files before running the NFS service.

- ***rpc.lockd*** - Files shared by the NFS service can also be used locally on the server that exports them. Simultaneous access to these files using two different methods could cause problems in terms of consistency. Hence, *rpc.lockd* and *rpc.statd* are responsible for preserving consistency in such situations.
- ***Lockd*** - Supports locking requests for files in the following way:
    1. *Lockd* on the client side sends a request to a server-side daemon to lock the file on which the client will operate. It does this using the RPC packet.
    2. *Lockd* communicates with the *statd* daemon to determine whether the lock can be created.
    3. If both daemons confirm the possibility of creating the lock, the request is sent to the kernel and the file is locked.
- ***biod*** - Block I/O Daemon - Runs on each client. All operations on the remote (mounted via the NFS) directory are supported by the BIOD. By default, the system starts six BIOD threads. A single BIOD can handle multiple remote directories.

Server

The first step toward sharing files by means of the NFS is to enable the service. You can do this using the *mknfs* command:

- ***mknfs -N*** - Launches the */etc/rc.nfs* script to enable the service. The service, when started in this way, will work until the next server restart.
- ***mknfs -I*** – Adds the */etc/rc.nfs* entry to the */etc/inittab* file. This runs *nfs* services every time the server is restarted.
- ***mknfs -B*** - A combination of the two *mknfs* calls above. The service runs immediately and every time the server is restarted.

You can do the same thing using the SMIT menu: *smit mknfs*. You can also activate and deactivate all the NFS daemons via the SRC commands, for example, *startsrc -g nfs* and *stopsrc -g nfs*. When starting the service, please note that the server interfaces must be described in the */etc/hosts* file. Otherwise, *rpc.statd* will not start, and it will display the following message:

```
statd -a can't get ip configuration
```

No matter which way you start the service, it works in the same way.

The next step toward making the NFS service useful is to export the directories that you want to share and determine which systems will be able to mount those directories. You can do this by editing the */etc/exports* file or using the *smit mknfsexp* menu. The first approach looks like this:

In the */etc/exports* file, you add the corresponding entries that indicate the shared directories. They may look like this:

```
# cat /etc/exports
/netdir -root=aix72c2,access=aix72c2:aix72
/all -ro

# grep -E "aix72c2|aix72" /etc/hosts   # Hosts used in the /etc/exports file.
10.10.177.2     aix72
10.10.177.4     aix72c2
```

- */netdir* - This row indicates that the resource can be mounted by aix72c2 and aix72 hosts, with aix72c2 having full access to the exported resource (the root rights).
- */all* - All the hosts will have access to the directory in read-only mode.

For the NFS to use these entries, they must be located in the */etc/xtab* file. To copy them from */etc/exports* to */etc/xtab*, use the *exportfs* command. This command also checks the correctness of the entries:

```
# exportfs -a
```

After executing this command, the resources specified in the */etc/exports* file can be mounted via the NFS. You can see if they are visible to external systems by running the *showmount* command on a local host or on another host in the network:

```
# showmount -e   # Checking from the perspective of the host that exports the resources.
export list for aix72c1:
/netdir aix72c2,aix72   # A resource shared with the aix72c2 and aix72 hosts.
/all    (everyone)   # A resource shared with all the hosts in read-only mode (the command output
does not contain access mode information).

# showmount -e aix72c1   # Checking from the perspective of the host that can use the exported
resources.
export list for aix72c1:
/netdir aix72c2,aix72
/all    (everyone)
```

When exporting directories via the *smit mknfsexp* menu, the changes are immediately saved to the */etc/exports* and */etc/xtab* files. All you must do in this case is enter the relevant data in the menu.

Client

On AIX, the NFS client daemons (*biod*, *rpc.lockd*, *rpc.statd*) are started by default after system installation. The only action you need to take to use the resources provided by the server is to mount them. The mounting process proceeds as follows:

1. The client queries the *portmap* daemon on the server side for the port number on which *rpc.mountd* is running.
2. The *portmap* provides the port number.
3. The client make a request to the *rpc.mountd* daemon to mount the resource.
4. *Rpc.mountd* checks whether the client has the right to mount the resource and whether the resource exists. If both conditions are met, it provides the resource to the client.
5. The client mounts the resource. From now on, it can work on the resource as on a regular file system.

The mounting process is illustrated in Figure 12-5.

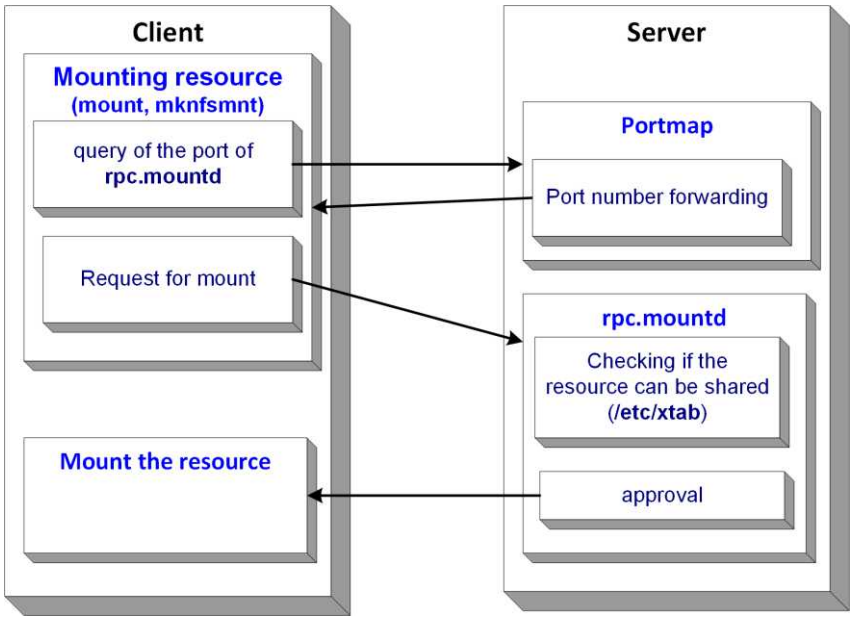

Figure 12-5: Mounting the resource shared by the NFS.

You can mount resources using several different tools:

- The *mount* command - As with ordinary file systems, for example:

```
# mount -n aix72c1 /netdir /mnt/nfs   # The first way.
# mount aix72c1:/netdir /mnt/nfs   # The second way.
```

- The *mknfsmnt* command - This is specific to the AIX operating system, for example:

```
# mknfsmnt -f /mnt/nfs -d /netdir -h aix72c1
```

- The *mknfsmnt* SMIT menu, which runs on the *mknfsmnt* command. The menu is shown in Figure 12-6.

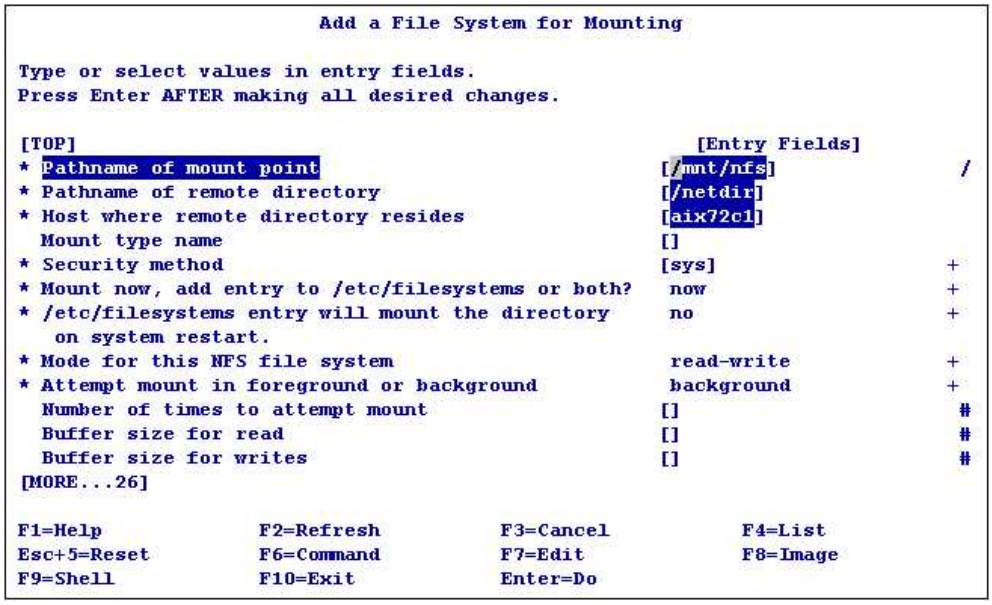

Figure 12-6: Smit mknfsmnt - Mounting shared directories via the NFS

Using the latter two methods, you can add the appropriate entry to the */etc/filesystems* file. This simplifies the subsequent mounting of resources or causes them to automatically mount during system startup:

```
/mnt/nfs:
        dev             = "/netdir"
        vfs             = nfs
        nodename        = aix72c1
        mount           = false
        options         = intr
        account         = false
```

If there is a corresponding entry in the mentioned file, then simply use the *mount* command to gain access to the described resource. In the above case: *mount /mnt/nfs*.

During the mounting process, you can decide on many parameters that affect the process itself, or the way the mounted resource works. The most important parameters that you can decide are:

Mount method:

- ***soft*** - The client tries to mount the resource a certain number of times (1000 by default) and, if it fails, the client resigns from further attempts.
- ***hard*** - The client tries to mount the resource indefinitely.
- ***foreground*** - The default setting. The host tries to mount the resource and does not return control (does not release the command line) until it does so.
- ***background*** - If the first attempt fails, subsequent attempts are made in the background. This is recommended when mounting file systems automatically at startup, since the foreground setting may cause the system to hang if there is no connection to the server.

Protocol used:

- ***TCP*** - Supports NFS connections using the TCP. The TCP is a connection-oriented protocol. Sent packets are acknowledged by the recipient so that the system knows whether the packet has reached the recipient or not. Lost packets are quickly retransmitted, which does not cause significant delays. However, maintaining a permanent connection and confirming the receipt of packets generate additional network traffic. This protocol is used by default.
- ***UDP*** - Supports NFS connections using the UDP. The UDP is a connectionless protocol. Sent packets are not acknowledged. The fact that some packets have been lost is only determined after the time out. Losing a packet results in significantly longer delays than in the case of the TCP. However, the UDP does not send any additional information on the network.

For NFS version 4, you can use only the TCP.

Mount Type:

- ***rw*** - Read/write.
- ***ro*** - Read only.

You can work on a resource mounted on the NFS as on a resource in a local file system. The resource is visible in the system in a similar way to a local file system:

```
# df -k
Filesystem    1024-blocks      Free %Used    Iused %Iused Mounted on
/dev/hd4           344064    167536   52%    11418    23% /
/dev/hd2          5226496    989460   82%    36714    15% /usr
/dev/hd9var        180224    146296   19%      598     2% /var
/dev/hd3           131072    100560   24%       57     1% /tmp
/dev/hd1            16384     16012    3%        7     1% /home
/dev/hd11admin      131072    130692    1%        5     1% /admin
/proc                  -         -    -         -      - /proc
/dev/hd10opt        32768     15964   52%      225     6% /opt
/dev/livedump      262144    261776    1%        4     1% /var/adm/ras/livedump
aix72c1:/netdir       344064    167368   52%    11418    23% /mnt/nfs
```

Performance

One of the key components of NFS server performance is the number of threads involved in the *nfsd* process. By default, eight threads are defined, and one is started. The rest are only started when needed. One thread can handle only one job at a time. This means that for a very large number of jobs, the *nfsd* threads may not be able to handle them all simultaneously, which in turn slows down the entire NFS subsystem. However, the need to improve NFS server performance by increasing the number of *nfsd* threads is rare. Changing their number does not cause any problems; you can do so using the command *chnfs*, for example:

```
# chnfs -n 12   # Increases the number of nfsd threads to 12.
0513-077 Subsystem has been changed.
Setting nfs_max_threads to 12
Warning: a restricted tunable has been modified
```

With the same command, you can also configure the NFS service on the client side. You can modify

the number of active threads of the *biod* daemon that support reads and writes to an NFS resource. By default, there are six threads of this daemon. This number can be increased using the *chnfs -b* command, for example:

```
# chnfs -b 12   # Increasing the number of biod daemon threads to 12.
0513-044 The biod Subsystem was requested to stop.
0513-077 Subsystem has been changed.
0513-059 The biod Subsystem has been started. Subsystem PID is 4718998.
```

Nfso is an important NFS command. It provides you with access to many NFS server parameters. With it, you can both check and change those parameters:

```
# nfso -Fa   # Displays the NFS server parameters.
           client_delegation = 1
           nfs_max_read_size = 65536
          nfs_max_write_size = 65536
                nfs_rfc1323 = 1
   nfs_securenfs_authtimeout = 0
    nfs_server_base_priority = 0
           nfs_server_clread = 1
      nfs_use_reserved_ports = 0
   nfs_v3_server_readdirplus = 1
    nfs_v4_fail_over_timeout = 0
                  portcheck = 0
           server_delegation = 1
             utf8_validation = 1
      nfs_server_close_delay = 0
               nfs_hang_log = 6
##Restricted tunables
           lockd_debug_level = 0
       nfs_allow_all_signals = 0
        nfs_auto_rbr_trigger = 0
         nfs_dynamic_retrans = 1
        nfs_gather_threshold = 4096
            nfs_iopace_pages = 0
             nfs_max_threads = 12
         nfs_repeat_messages = 0
              nfs_socketsize = 600000
nfs_tcp_duplicate_cache_size = 5000
          nfs_tcp_socketsize = 600000
nfs_udp_duplicate_cache_size = 5000
                nfs_v2_pdts = 1
                nfs_v3_pdts = 1
                nfs_v4_pdts = 1
             nfs_v2_vm_bufs = 10000
             nfs_v3_vm_bufs = 10000
             nfs_v4_vm_bufs = 10000
           statd_debug_level = 0
           statd_max_threads = 50
                 udpchecksum = 1
                  gss_window =
```

Description of some key parameters:

- ***nfs_socketsize*** - Defines the size of the queue for storing the requests of clients that have been sent in the form of UDP packets. If the system handles many such client requests, you should increase the queue. However, it should never be reduced. This parameter is dynamic; you can change it with the active NFS service.
- ***nfs_tcp_socketsize*** - Specifies the size of the queue for storing the requests of clients that

have been sent in the form of TCP packets. This parameter works in the same way as *nfs_socketsize*. The only difference is the protocol that it concerns.

- ***nfs_max_read_size***, ***nfs_max_write_size*** - The maximum size of packages that clients can use when communicating with the NFS. You can change this parameter at any time.
- ***nfs_rfc1323*** - A parameter that allows you to use a TCP window that is larger than 65,535 bytes. You can change this parameter at any time.

You can change the NFS server parameters using the *-o* flag of the *nfso* command, for example:

```
# nfso -o nfs_rfc1323=0   # Changes the current value of the parameter (until you restart the
server).
Setting nfs_rfc1323 to 0
```

Additional useful flags:

- ***-p*** - Changes the current value of the parameter and sets this value every time the system is restarted (because of the entry in */etc/tunables/nextboot*). Example: *nfso -po nfs_rfc1323 = 1*.
- ***-r*** - Changes the parameter permanently, starting from the next restart. Example: *nfso -ro nfs_rfc1323 = 1*.
- ***-d*** - Changes the parameter to the default value. Example: *nfso -rd nfs_rfc1323*.

Automatic mount

AIX allows you to mount file systems on demand. You can configure the system so that the selected file systems are only mounted when the user wants to perform an operation on them. Unmounting also takes place automatically after the period of inactivity specified in the configuration.

You can configure the automount of all file systems known to AIX: JFS, JFS2, NFS, and CDRFS. The automount process manages this mechanism. It is inactive by default. Two files are used for the automount configuration, namely */etc/auto_master* and the map file.

/etc/auto_master - Contains entries indicating the main mount point and the map file, which specifies in detail the resources that are to be mounted at this mount point. If the */etc/auto_master* file does not exist, you should create it. Sample content:

```
# cat /etc/auto_master
/auto   /etc/automount.map
```

Meaning of entries:

- ***/auto*** - Specifies the main mount point. The resources specified in the map file will be mounted here. The directory does not have to already exist, since it is created automatically by the *automount* daemon.
- ***/etc/automount.map*** - A map file that points to the resources you want to mount automatically in the */auto* directory, if needed.
- ***Map file*** - A separate map file is indicated for each main mountpoint defined in */etc/auto_master*. The individual lines of the map file provide information about three elements:
  1. The subdirectory of the main mount point where the resource should be mounted.

2. The mount options.
3. The location of the resource to be mounted.

Sample of the map file contents:

```
# cat /etc/automount.map
netdir  -rw     aix72c1:/netdir

all     -ro     aix72c1:/all
```

The first line tells you that in the */auto/netdir* directory you want to mount the */netdir* resource, which is exported by the *aix72c1* server using the NFS. The resource is to be mounted in full access mode (*rw*). Similarly, the second line tells you that you want to mount the */all* resource from the *aix72c1* server in the */auto/all* directory. This resource is to be mounted in read-only mode (*ro*).

In order to run the described mechanism, run the */usr/sbin/automount* daemon after configuring the configuration files. If it is already running, you must restart it, which will force it to read the changed configuration files.

After doing so, the automatic mount service works. From now on, any attempts to reference the defined directories will mount the resources specified in the configuration files. After five minutes of inactivity, they will be unmounted. The following example shows this process:

```
# df -k   # System state immediately after launching the automount service.
Filesystem    1024-blocks      Free %Used    Iused %Iused Mounted on
/dev/hd4           344064    167524   52%    11422    23% /
/dev/hd2          5226496    989460   82%    36714    15% /usr
/dev/hd9var        180224    146192   19%      598     2% /var
/dev/hd3           131072    100560   24%       57     1% /tmp
/dev/hd1            16384     16012    3%        7     1% /home
/dev/hd11admin      131072    130692    1%        5     1% /admin
/proc                  -         -    -         -      - /proc
/dev/hd10opt        32768     15964   52%      225     6% /opt
/dev/livedump      262144    261776    1%        4     1% /var/adm/ras/livedump

# cd /auto/netdir   # After "touching" with any command from the defined catalog, a specific
resource is mounted.

# df -k
Filesystem    1024-blocks      Free %Used    Iused %Iused Mounted on
/dev/hd4           344064    167524   52%    11422    23% /
/dev/hd2          5226496    989460   82%    36714    15% /usr
/dev/hd9var        180224    146192   19%      598     2% /var
/dev/hd3           131072    100560   24%       57     1% /tmp
/dev/hd1            16384     16012    3%        7     1% /home
/dev/hd11admin      131072    130692    1%        5     1% /admin
/proc                  -         -    -         -      - /proc
/dev/hd10opt        32768     15964   52%      225     6% /opt
/dev/livedump      262144    261776    1%        4     1% /var/adm/ras/livedump
aix72c1:/netdir       344064    167360   52%    11422    23% /auto/netdir

# ls /auto/all   # "Touching" the second directory also mounts the specified resource.

# df -k
Filesystem    1024-blocks      Free %Used    Iused %Iused Mounted on
/dev/hd4           344064    167524   52%    11422    23% /
/dev/hd2          5226496    989460   82%    36714    15% /usr
/dev/hd9var        180224    146192   19%      598     2% /var
```

```
/dev/hd3          131072    100560   24%       57     1% /tmp
/dev/hd1           16384     16012    3%        7     1% /home
/dev/hd11admin    131072    130692    1%        5     1% /admin
/proc                  -         -    -        -     - /proc
/dev/hd10opt       32768     15964   52%      225     6% /opt
/dev/livedump     262144    261776    1%        4     1% /var/adm/ras/livedump
aix72c1:/netdir      344064    167360   52%    11422    23% /auto/netdir
aix72c1:/all      344064    167360   52%    11422    23% /auto/all
```

NFS version 4

Starting with AIX 5.3, the latest version of the NFS introduced, namely NFS version 4. It introduced several important changes, including:

- **RPCSEC-GSS RPC Authentication** - Provides enhanced NFS security. Using it is not compulsory. Authentication from previous versions of the protocol can still be used.
- **Changes the way files are locked** - In earlier versions, the *Network Lock Manager* (NLM) and *Status Monitor Protocol* were used to establish and control file locking. These protocols were supported by the *rpc.lockd* and *rpc.statd* daemons, respectively. In version 4, this task has been moved to the main NFS protocol, so the above-mentioned daemons are no longer used, although they continue to function in the case of NFS versions 2 and 3.
- **Support for NFS v4 ACL** - NFS version 4 defines an access control list (ACL) that allows for the more detailed definition of file permissions. AIX provides two types of access control lists. Only the list defined with the NFS protocol can be used by it. The access control lists known to AIX are:
  - **AIX Classic (AIXC)** - Known from earlier versions of the system. It is AIX-specific, so it does not apply to NFS connections between different operating systems. This type of list is only supported by the NFS when both the client and the server are AIX-based.
  - **NFS4** - Defined within the NFS v4 protocol and implemented with it. It is independent of the operating system. This list should work regardless of the operating system type on the client and on the server.

To use the NFS4 access control list, shared file systems must be created using the Extended Attribute Format parameter of *Version 2*. If they are not, you can change this parameter "on the fly," without unmounting the file system. It is worth mentioning that when both the NFS client and the server are based on the AIX system, the NFS4 access control lists can also work with the NFS v3 protocol.

To provide the additional functionality of NFS v4 described in this chapter, additional daemons have been introduced: *nfsrgid* and *gssd*. They only work if you configure the NFS to use the version 4 protocol; otherwise, they will not run:

```
# lssrc -g nfs
Subsystem         Group            PID          Status
 rpc.statd        nfs              5702092      active
 rpc.lockd        nfs              4981160      active
 nfsd             nfs              11338098     active
 rpc.mountd       nfs              13238594     active
 biod             nfs              4718998      active
 nfsrgyd          nfs                           inoperative
 gssd             nfs                           inoperative
```

Along with NFS v4, additional tools have also been introduced, such as *chnfsdom*, *nfs4cl*, *nfshostkey*, *chnfsim*, and others.

Network options

AIX allows you to specify many network parameters. One of the main commands for this purpose is the *no* (Network Options) command. With it, you can both check and set values. You can make temporary changes to the currently running system or permanent changes that persist after system restart. When you change the parameters permanently, new values for each parameter are written to the */etc/tunables/nextboot* file, which is read at startup. By default, changes are only made to the currently running system.

To obtain information about the current values of individual parameters, you can use the *no -a* command. However, much more information is provided by the *-L* parameter. The following output shows how the *no -L* command displays information (in the interests of brevity, some information has been omitted):

```
# no -L
General Network Parameters
-------------------------------------------------------------------------------
NAME                      CUR    DEF    BOOT   MIN    MAX    UNIT           TYPE
     DEPENDENCIES
-------------------------------------------------------------------------------
bsd_loglevel              3      3      3      0      7      numeric          D
-------------------------------------------------------------------------------
fasttimo                  200    200    200    50     200    millisecond      D
-------------------------------------------------------------------------------
# All unnecessary information has been omitted.

TCP Network Tunable Parameters
-------------------------------------------------------------------------------
NAME                      CUR    DEF    BOOT   MIN    MAX    UNIT           TYPE
     DEPENDENCIES
-------------------------------------------------------------------------------
clean_partial_conns       0      0      0      0      1      boolean          D
-------------------------------------------------------------------------------
delayack                  0      0      0      0      3      boolean          D
-------------------------------------------------------------------------------
# All unnecessary information has been omitted.

UDP Network Tunable Parameters
-------------------------------------------------------------------------------
NAME                      CUR    DEF    BOOT   MIN    MAX    UNIT           TYPE
     DEPENDENCIES
-------------------------------------------------------------------------------
udp_bad_port_limit        0      0      0      0      8E-1   numeric          D
-------------------------------------------------------------------------------
udp_sendspace             9K     9K     9K     4K     8E-1   byte             C
     sb_max
-------------------------------------------------------------------------------
# All unnecessary information has been omitted.

IP Network Tunable Parameters
-------------------------------------------------------------------------------
NAME                      CUR    DEF    BOOT   MIN    MAX    UNIT           TYPE
     DEPENDENCIES
```

```
--------------------------------------------------------------------------------
directed_broadcast        0      0      0      0      1      boolean           D
--------------------------------------------------------------------------------
ie5_old_multicast_mapping 0      0      0      0      1      boolean           D
--------------------------------------------------------------------------------
# All unnecessary information has been omitted.

ARP/NDP Network Tunable Parameters
--------------------------------------------------------------------------------
NAME                      CUR    DEF    BOOT   MIN    MAX    UNIT          TYPE
     DEPENDENCIES
--------------------------------------------------------------------------------
arpqsize                  1K     1K     1K     1      32K-1  numeric           D
     tcp_pmtu_discover
     udp_pmtu_discover
--------------------------------------------------------------------------------
arpt_killc                20     20     20     0      255    minute            D
--------------------------------------------------------------------------------
# All unnecessary information has been omitted.

Stream Header Tunable Parameters
--------------------------------------------------------------------------------
NAME                      CUR    DEF    BOOT   MIN    MAX    UNIT          TYPE
     DEPENDENCIES
--------------------------------------------------------------------------------
lowthresh                 90     90     90     0      100    %_of_thewall      D
--------------------------------------------------------------------------------
medthresh                 95     95     95     0      100    %_of_thewall      D
--------------------------------------------------------------------------------
# All unnecessary information has been omitted.

Other Network Tunable Parameters
--------------------------------------------------------------------------------
NAME                      CUR    DEF    BOOT   MIN    MAX    UNIT          TYPE
     DEPENDENCIES
--------------------------------------------------------------------------------
bcastping                 0      0      0      0      1      boolean           D
--------------------------------------------------------------------------------
dgd_flush_cached_route    0      0      0      0      1      boolean           D
--------------------------------------------------------------------------------
# All unnecessary information has been omitted.
n/a means parameter not supported by the current platform or kernel

Parameter types:
    S = Static: cannot be changed
    D = Dynamic: can be freely changed
    B = Bosboot: can only be changed using bosboot and reboot
    R = Reboot: can only be changed during reboot
    C = Connect: changes are only effective for future socket connections
    M = Mount: changes are only effective for future mountings
    I = Incremental: can only be incremented

Value conventions:
    K = Kilo: 2^10        G = Giga: 2^30        P = Peta: 2^50
    M = Mega: 2^20        T = Tera: 2^40        E = Exa: 2^60
```

As you can see in the above output, the individual parameters are divided into sections that group parameters of a particular type. All the sections consist of the same columns, and they present the following information:

- ***NAME*** - Parameter name.
- ***DEPENDENCIES*** - Tells you what other parameters depend on the one described. For

example, the *udp_recvspace* buffer is dependent on the *sb_max* (it cannot exceed the *sb_max* because it is part of it).

- ***CUR (current value)*** - Current value of the parameter.
- ***DEF (default value)*** - Default value of the parameter. This value is set immediately after the system installation. You can change the value of a parameter to the default at any time.
- ***BOOT (reboot value)*** - The value that the parameter will take after system reboot.
- ***MIN (minimal value)*** - The minimum value that a parameter can accept.
- ***MAX (maximum value)*** - The maximum value that a parameter can accept.
- ***UNIT (tunable unit of measure)*** - Units in which a given parameter is determined. The units can be of various types, such as boolean (only two values are allowed: 1 - enabled, 0 - disabled), seconds, minutes, kilobytes, numerical values, etc.
- ***TYPE*** - Parameter type:
  - ***D (Dynamic)*** - The parameter can be changed at any time.
  - ***S (Static)*** - The parameter can never be changed.
  - ***R (Reboot)*** - The parameter can only be changed by rebooting the system.
  - ***B (Bosboot)*** - The parameter only becomes active after running the *bosboot* command and rebooting the system. When changing this type of parameter, the command does not ask whether to run the *bosboot* command.
  - ***M (Mount)*** - The parameter change will only take effect after re-mounting the file systems.
  - ***I (Incremental)*** - The parameter can only be increased.
  - ***C (Connect)*** - The parameter change will only apply to newly established connections (it does not apply to existing connections). When this parameter is changed, the system automatically restarts the *inetd* daemon.

Other examples of using the no command:

- **Change the *tcp_recvspace* parameter for the running system:**

```
# no -o tcp_recvspace=32768
Setting tcp_recvspace to 32768
Change to tunable tcp_recvspace, will only be effective for future connections
```

- **Change the parameter that will only be activated after the system reboots:**

```
# no -ro tcp_recvspace=32768
Setting tcp_recvspace to 32768 in nextboot file
Warning: changes will take effect only at next reboot
```

- **Change the parameter for the currently running system. The parameter remains changed after each system restart:**

```
# no -po tcp_recvspace=32768
Setting tcp_recvspace to 32768
Setting tcp_recvspace to 32768 in nextboot file
Change to tunable tcp_recvspace, will only be effective for future connections
```

- **Permanently change the parameter to the default value in the currently running system:**

```
# no -pd tcp_recvspace
Setting tcp_recvspace to 16384
Setting tcp_recvspace to 16384 in nextboot file
```

- **Display information about the selected parameter:**

```
# no -h tcp_recvspace
Help for tunable tcp_recvspace:
Purpose:
Specifies the system default socket buffer size for receiving data. This affects the window size
used by TCP.

Values:
        Default: 16384
        Range: 4096 - 9223372036854775807
        Type: Connect
        Unit: byte
Tuning:
The optimum buffer size is the product of the media bandwidth and the average round-trip time of a
packet. The tcp_recvspace network option can also be set on a per interface basis (reference
documentation on Interface Specific Network Options (ISNO) ). Most interfaces now have this tunable
set in the ISNO defaults. The tcp_recvspace attribute must specify a socket buffer size less than or
equal to the setting of the sb_max attribute.
```

The meaning of the individual parameters is described in the system manual (*man no*).

Chapter 13. Scheduling

There are many products that enable the starting, controlling, and synchronizing of tasks on different systems. In large organizations, where data processing requires running tasks in synchronization with other systems, such products are frequently used. In smaller organizations such complex schedulers are usually not needed. For typical tasks, the possibilities provided by the operating system are enough.

An important task in any system is jobs scheduling. Operating systems live their own lives by carrying out tasks that you often do not know straight away, as well as jobs that you can verify by browsing relevant configuration files. We will discuss these files a bit later in the chapter.

The *cron* daemon is responsible for running scheduled jobs. It is run by */etc/inittab* with the respawn option, which means that when you kill it, the daemon will automatically resume.

```
# lsitab cron
cron:23456789:respawn:/usr/sbin/cron
```

You can change the way cron handles jobs by modifying the */var/adm/cron/queuedefs* configuration file. You can change parameters such as the maximum number of simultaneous jobs, the priority with which those jobs will be run, and how often the *crontab* file will be checked in the context of the jobs to be run. However, there is usually no need to modify this file.

You can pass the tasks to be performed by the daemon in two ways:

- By configuring the crontab file, wherein you can define one-time or recurring jobs.
- By using the at or batch command to execute the command once.

Crontab files

Unless the administrator has configured it differently, each user has the option of defining the jobs to be run. Thus, each user has his own crontab file. The files are located in the */var/spool/cron/crontabs* directory, and they have names corresponding to the username.

The administrator decides which users have the access necessary to schedule jobs. There are two files to configure permissions:

- ***/var/adm/cron/cron.deny*** - If the *cron.allow* file does not exist, all users can schedule jobs, except for those listed in the cron.deny file.

- ***/var/adm/cron/cron.allow*** - If this file exists, only users who are listed in this file can schedule jobs. If both files exist, only *cron.allow* is used. If there is no file, only the root user can schedule jobs.

You can define one-time or recurring jobs in the *crontab* file. You can define the time of task execution by entries in the following format:

Minute(0–59) Hour(0–23) DayOfMonth(1–31) MonthOfYear(1–12) DayOfWeek(0–6, 0 means Sunday**) command**

An example of the root crontab file:

```
# crontab -l   # "-l" - Display the contents of the crontab file.
# All unnecessary information has been omitted.
#0 3 * * * /usr/sbin/skulker
#45 2 * * 0 /usr/lib/spell/compress
#45 23 * * * ulimit 5000; /usr/lib/smdemon.cleanu > /dev/null
0 11 * * * /usr/bin/errclear -d S,O 30
0 12 * * * /usr/bin/errclear -d H 90
0,5,10,15,20,25,30,35,40,45,50,55 * * * * /usr/sbin/dumpctrl -k >/dev/null 2>/dev/null
0 15 * * *  /usr/lib/ras/dumpcheck >/dev/null 2>&1
55 23 * * * /var/perf/pm/bin/pmcfg  >/dev/null 2>&1     #Enable PM Data Collection
0 23 1-15 * 1-5 /usr/bin/test
```

The date and time format can be seen on the example of the above output:

`0 11 * * * /usr/bin/errclear -d S,O 30`
The */usr/bin/errclear -d S, O 30* command will run every day at *11:00*.

`0,5,10,15,20,25,30,35,40,45,50,55 * * * * /usr/sbin/dumpctrl -k >/dev/null 2>/dev/null`
The */usr/sbin/dumpctrl* command will start every *five* minutes.

`0 23 1-15 * 1-5 /usr/bin/test`
The */usr/bin/test* command will be run every day of the month from *1* to *15* (*1–15*) at *23:00*, but only on *weekdays* from *Monday* to *Friday* (*1–5*).

You can use special characters to define entries, including:

- *"*"* - Every number in a given field.
- *","* - You can specify several values in a given field. For example, 1,2,3 in the "day of the month" field means January, February, March.
- *"-"* - The range. For example, 1–5 in the "day" field means Monday to Friday.

You can use the *crontab* command to work with *crontab* files:

```
# crontab - l  # Display the contents of the crontab file.

# crontab -e   # Edit the crontab file in the editor described by the variable $ EDITOR.
```

After editing the file and saving its contents, cron begins to support the added or modified jobs.

At and batch commands

The *at* command adds a job to cron, which will be executed once at the time specified in the parameters. The output of the command will be sent to the owner by email. The *batch* command works the same as the *at* command, except that it should run the command during low CPU usage. As with the crontab file, the administrator can decide who can schedule jobs. There are two files for this

purpose:

- */var/adm/cron/at.deny* - If the *at.allow* file does not exist, all users can schedule jobs, except for those listed in the *at.deny* file.
- */var/adm/cron/cron.allow* - If this file exists, only users who are listed in this file can schedule jobs.

The time and date for starting a specific job can be written using keywords such as noon, midnight, am, pm, today, tomorrow, etc. The time can be saved as absolute, for example, *10 am Monday*, or relative, for example, *now + 5 minutes*.

The following commands are used to schedule and control the jobs:

```
# at [date and time]     # Create a task to run at a given time.

# batch         # Create a task to run during low CPU usage.

# at -l         # Display the created jobs.

# atq           # Equivalent to the at -l command.

# at -r job     # Cancel a job.

# atrm          # Equivalent to the at -r command.
```

An example of using the *at* command follows:

```
# at now + 100 minutes   # Run tasks after 100 minutes.
echo "finished"     > /tmp/at.test # Command to be run.
<ctrl+d>       # We press ctrl + d to finish entering the command.
Job root.1515439370.a will be run at Mon Jan  8 13:22:50 CST 2018.

# at -l         # We display the scheduled jobs.
root.1515439370.a Mon Jan 8 13:22:50 CST 2018

# at -vl root.1515439370.a    # We display details of the scheduled job.
========================
root.1515439370.a Mon Jan 8 13:22:50 CST 2018
========================
echo "finished"> /tmp/at.test
```

Chapter 14. Security

AIX has many features related to the security of the system. These features start from the basic configuration options that most UNIX/LINUX systems have, and they end with advanced mechanisms based on the cooperation of the operating system with the server and hypervisor. There are a number of features that you should pay particular attention to:

- **ACL (Access Control Lists)** - Extended access to files that goes beyond the standard *rwx* permissions for the owner, group, and other users. In this case, two variants of the access lists can be used: AIXC or NFS4 ACL.

- **RBAC (Role Based Access Control)** - A response to a typical security problem in UNIX/LINUX systems. The problem is the wide use of the root user. With this mechanism, it is possible to create specific roles for specific tasks performed by users. Thus, it eliminates the need to work with too high permissions within a system.

- **Auditing** - Tracking and logging of system work and the user's work, as well as the detection of undesired activities in the system. With this mechanism, it is possible to trace whether the given events took place in the system, or to trace who performed specified file operations.

- **AIXpert (AIX Security Expert)** - A mechanism that allows the management of multiple system security parameters. It contains several hundred settings in defined security levels. These levels can be implemented in the system using one command, making the system compatible with a predefined security pattern or with your own custom pattern.

- **IPSec** - A mechanism that allows the full protection of communication between systems.

- **TE (Trusted Execution)** - A mechanism that allows the testing of the integrity of the system. It can act after the event by providing information about violations of the integrity of the system. With proper configuration, it can run in the background, thereby increasing the level of security. The *Trusted Execution* is the successor of the *Trusted Computing Base (TCB)* mechanism known from many UNIX systems. The TCB can still be used on the AIX system if this option is selected during installation.

- **EFS (Encrypted File system)** - A file encryption method in the JFS2 file system.

- **PowerSC** - A collection of advanced AIX security features that are available as a separate product in the IBM offer. This approach is explained by the fact that the elements contained here operate on the interface between the server and the virtualization layer, and they are often based on additional software components. The main PowerSC components:

    - **PowerSC: Trusted Logging** - The response to a problem related to an intelligent hacker. He can change the system to suit his own needs, and then can erase any traces of his presence from the system logs. In the case of Trusted Logging, the log storage location can be defined as unmodifiable. A copy of the logs can be located on the Virtual I/O Server. An additional advantage of this approach is the means of writing

these logs, which takes place through a dedicated device and hypervisor. It does not pass through the network, which makes it impossible to modify or stop this information when it is in the transmission path.

- **PowerSC: Trusted Boot** - An advanced mechanism that can confirm that the system being started is "trusted," that is, it has not been compromised or modified in any way and is hence compatible with the stored pattern. Special registers are used to operate this mechanism. Communication between the system and the hypervisor in the context of a Trusted Boot is carried out using the *Virtual Trusted Platform Module (VTPM)*. This device is activated from the HMC, and it exists in the operating system as */dev/vtmp0*.

- **PowerSC: (TNC/PM) Trusted Network Connect and Patch Management** - A feature for increasing network security by controlling the level of AIX patching. It also allows you to patch these systems manually or automatically. The mechanism is based on a combination of several functionalities:
  - **TNC server (Trusted Network Connect)**, which is the core element of the system.
  - **NIM server (Network Installation Manager)**, which is used for patching the systems.
  - **SUMA service (Service Update Management Assistant)**, which allows the automatic downloading of patches from the IBM repository.

- **PowerSC: Compliance and Security** - A collection of predefined settings based on the AIXpert functionality. It allows the quick adaptation of the system to certain security standards, such as the Department of Defense Security Technical Implementation Guide, PCI DSS, SOX/COBIT, HIPAA, and NERC.

- **PowerSC: Trusted Firewall** - A mechanism that allows the filtering of traffic to and from virtual machines at the VIOS level. At the same time, it enables the optimization of traffic in such a way that traffic between the LPARs that are on the same server but in different VLANs does not go outside the server. This type of packet routing is served by the VIOS.

Some of the security mechanisms can be enabled during system installation:

- **Secure by Default** - Installs the system in a specific, more secure way. Using this option will cause several changes in the installed system:
  - A limited number of packages will be installed, approximately half those installed during a normal installation without this option.
  - *bos.net.tcp* filesets will be installed in a lighter version that does not include services considered dangerous, such as RSH, RCP, Telnet, and FTP.
  - The first time the system is started, the security level described in the */etc/security/aixpert/core/SbD.xml* file is implemented using the AIXpert functionality.

  The setting of the *Secure by Default* option may be rolled back at a later time.

- **Trusted AIX** - Installs the operating system with the mechanism of access to objects, which is based on labels. This system uses the RBAC and TE mechanism by default. Thanks to the

classification of objects as well as the use of *labels*, you are able to model the structure of the organization in the operating system, thereby granting access consistent with belonging to a given department and the level of the granted access.

- **LSPP/EAL4+, CAPP/EAL4+** - Installs the system in accordance with the security profiles of the LSPP (*Labeled Security Protection Profile*) or CAPP (*Controlled Access Protection Profile*) at the level of EAL 4+ (*Evaluation Assurance Level*). More information about security profiles can be found at https://www.commoncriteriaportal.org.

In the remainder of this chapter, some security-related mechanisms will be described in more detail.

RBAC (Role-Based Access Control)

In traditional UNIX systems, there are simple authorization management mechanisms. You can assign permissions to the owner, group, and other users. In cases where such an allocation of rights is insufficient, you can set the SUID or SGID bits. These bits ensure that the effective owner of the running program (process) is not the user/group who started it, but rather the user/group who owns the file.

Such a limited permission mechanism causes a lot of unsecure behaviors, such as:

- **Frequent work as a root user**. This way of working has many disadvantages. When working from the superuser's position, you have much higher permissions than are actually required to perform a given task. This leads to the risk of intentional or unintentional abuse of authorization, such as the launch of malicious software. Such work is also difficult to audit. If there are several administrators who simultaneously work on the root user account, it is difficult to identify who made the change, since in each case the root user will be listed in the logs.

- **Frequent use of SUID and SGID bits**. This way of working causes many gates to be kept in the system that could be used by hackers. Files that have the SUID and SGID bits set are considered dangerous, and their number should be limited.

The above-mentioned traditional approach means that many trivial activities must also be performed by the system administrator. There is no easy and safe way to separate a group of tasks for other system users to perform.

The solution to the above problems lies in the implementation of RBAC (*Role-Based Access Control*) in AIX. With this solution, every user in the system can have permissions in line with their requirements, without needing to use the SUID/SGID bits. This solution existed from the early versions of AIX, but from AIX 6.1 it existed in a significantly improved form. Starting from that version, so-called Enhanced RBAC was available.

The most important files related to RBAC:

- ***/etc/security/roles*** - Role definitions.
- ***/etc/security/authorizations*** - User-defined authorizations. The file does not contain

system authorizations.

- */etc/security/privcmds* - Command authorizations.
- */etc/security/privfiles* - File authorizations.
- */etc/security/privdevs* - Device authorizations.

After installing the operating system, the solution is automatically enabled, and you can use it to assign and use predefined roles or to create new roles as needed. With RBAC, we distinguish several elements that allow you to configure permissions in the system. These are the roles, authorizations, and privileges associated with the process. The relations between these factors are shown in Figure 14-1.

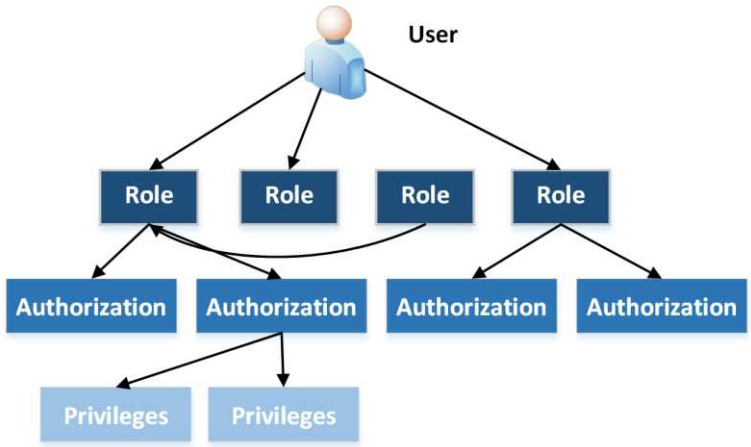

Figure 14-1: RBAC - Roles, authorizations, privileges.

Roles

The role is defined by the set of rights (authorizations) in the system. It is assigned to users, each of whom can have one or more roles and switch between them in order to perform different tasks. In the system, there are several types of roles available by default:

- **Legacy roles** - Simple roles consisting of specific authorizations that do not inherit permissions from other roles (no *rolelist* attribute in the role definition).

- **Compound roles** - Complex roles that have their own authorizations and inherit authorizations from other roles listed in the *rolelist* attribute. Typical and commonly used roles of this type include:
  - *Information System Security Officer (isso)*;
  - *System Administrator (sa)*; and
  - *System Operator (so)*.

The definitions of the roles can be found in the */etc/security/roles* file. An example of the definition of the *so* (*System Operator*) role follows:

```
# grep -p "so:" /etc/security/roles
so:
```

```
        id = 3
        dfltmsg = System Operator
        authorizations = aix.proc.kill,aix.ras,aix.system.config.init,aix.system.config.wlm
        rolelist = BackupRestore,SysBoot
        msgcat = role_desc.cat
        msgset = 1
        msgnum = 3
```

The most important information about the role definition in the example concerning the *System Operator* role:

- **authorizations** - Detailed permissions and authorizations assigned to the role. They will be discussed in more detail later in this chapter

- **rolelist** - A list of additional roles assigned to a given role. This means that the role has its own authorizations, and authorizations that are assigned to the roles listed in *rolelist* parameter. So, the *so* role will have its own authorizations as well as authorizations listed in the *BackupRestore* and *SysBootroles* roles, that is:

```
# grep -p "BackupRestore:" /etc/security/roles | grep authorizations
        authorizations = aix.fs.manage.backup,aix.fs.manage.restore

# grep -p "SysBoot:" /etc/security/roles | grep authorizations
        authorizations =
aix.system.boot.create,aix.system.boot.halt,aix.system.boot.info,aix.system.boot.reboot,aix.system.b
oot.sutdown
```

- **dfltmsg** - A description of the role, which is used when the information in the *msgcat* is unavailable.

- **msgcat, msgset, msgnum** - Indicates the role description, which in this case is in the */usr/lib/nls/msg/en_US/role_desc.cat* file.

Commands related to roles

A user may have multiple roles, but only one of them remains active at any time. If a user wants to use a certain role, he must switch to it using the *swrole* command, for example:

```
# mkuser testuser   # User creation.

# chuser roles=BackupRestore,DomainAdmin,FSAdmin testuser  # Assignment of roles to the user.

# su - testuser

$ rolelist   # Listing the roles a user has.
BackupRestore   Backup and Restore Administration
DomainAdmin     Remote Domain Administration
FSAdmin         File System Administration

$ rolelist -ea   # Listing the active roles (-e) and authorizations (-a).
rolelist: 1420-062 There is no active role set.

$ swrole BackupRestore   # Switch to the BackupRestore role.
testuser's Password:
```

```
$ rolelist -ea   # The role is active and has the following authorizations.
BackupRestore   aix.fs.manage.backup
                aix.fs.manage.restore
```

The above example shows the creation of a user, the assignment of three roles to him, and then the activation of the role the user wants to use. When switching the role, you must enter the user password again. If you want to avoid switching between roles, you can create a separate *compound* role that would use all three roles assigned to the user (the *rolelist* parameter). In addition, you can make this role the default role for the user. In this case, it will be active immediately after the user logs in to the system. An example of setting the default role follows:

 # chuser roles=NewRole default_roles=NewRole testuser

Other important commands:

```
# swrole  # Switching between roles.
# rolelist    # Listing user roles.
# mkrole  # Creation of a role.
# lsrole  # Listing information about roles.
# chrole  # Changing the role attributes.
# rmrole  # Removal of a role.
```

Authorizations

As mentioned earlier, roles are assigned specific authorizations. The authorization names are built hierarchically, and they consist of four parts starting with the word "aix", for example, *aix.system.boot.reboot* indicates the right to restart the system. Examples of hierarchical records are presented in Table 14-1.

Table 14-1: Authorizations - Hierarchical records.

Component1	Component2	Component3 for Component2="system"	Component4 for Componant3="boot"
aix	security	stat	create
	ras	boot	info
	proc	config	reboot
	system	install	shutdown
	network		halt
	fs		
	…		

The hierarchical record allows you to clearly identify the authorizations granted, and it allows for the more efficient definition of roles. For example, instead of assigning multiple individual permissions such as *aix.system.boot.reboot*, *aix.system.boot.shutdown*, and so on, you can assign a more general permission: *aix.system.boot*. This will contain all the permissions that exist in element 4 for the record *aix.system.boot*.

It is best to describe the way of defining authorization based on the configuration files. The most important configuration files are:

/etc/security/privcmds

This file stores the descriptions of authorizations and permissions for individual operating system commands. By default, nearly 1000 different commands are described in the file. Each command is described in a separate section that begins with the command path. Below are some examples for the */usr/sbin/mkdev* and */usr/bin/mksysb* commands:

```
/usr/sbin/mkdev:
        accessauths = aix.device.manage.create
        innateprivs = PV_AU_ADD,PV_AU_PROC,PV_DAC_O,PV_DAC_R,PV_DAC_X,PV_FS_MKNOD,
                      PV_KER_CLUSTER
        inheritprivs = PV_DAC_O,PV_DAC_R,PV_DAC_W,PV_DAC_X,PV_DEV_CONFIG,PV_DEV_LOAD,
                       PV_DEV_QUERY,PV_FS_CHOWN,PV_FS_MKNOD,PV_KER_ACCT,PV_KER_CONF,
                       PV_KER_EXTCONF,PV_KER_LVM,PV_KER_RAS,PV_KER_VARS,PV_NET_CNTL,
                           PV_NET_CONFIG,PV_PROC_PRIV,PV_PROC_SIG,PV_SU_UID,PV_KER_CLUSTER
        secflags = FSF_EPS
        euid = 0
        egid = 0

/usr/bin/mksysb:
        accessauths = aix.system.boot.create
        innateprivs = PV_AZ_ROOT,PV_DAC_O,PV_DAC_R,PV_DAC_W,PV_DAC_X
        inheritprivs = PV_AZ_ROOT,PV_DAC_GID,PV_DAC_O,PV_DAC_R,PV_DAC_UID,PV_DAC_W,
                       PV_DAC_X,PV_FS_CHOWN,PV_KER_ACCT,PV_PROC_PRIV,PV_PROC_SIG,
                           PV_SU_UID,PV_TCB,PV_AZ_CHECK
        euid = 0
        egid = 0
        secflags = FSF_EPS
```

The most important parameters:

- ***euid, egid*** - Effective permissions. User and group permissions with which the command is run.

- ***accessauths*** - A list of authorization names that are used when defining a role. In the examples above, there is one name per file, although there may be more. There can be up to 16 authorization names assigned to a command. Having any of these authorizations will allow you to run the command. This field may also contain special values, such as *ALLOW_OWNER*, *ALLOW_GROUP*, and *ALLOW_ALL*. These values allow the owner, group, or all users (according to the classic file access permissions) to run the command without checking the authorization.

- ***innateprivs*** - A list of privileges assigned to the process when the command is run.

- ***inheritprivs*** - A list of privileges passed to child processes.

Privileges of the process

As detailed above, process privileges are one of the main parameters defined for commands. Privileges may seem unnecessary at first glance, but they represent a very important element of security. Thanks to privileges, a given process can only do what you let it do. If a hacker manages to get into your system and somehow replaces the executable file with his own version, then the substituted program will only have permission to perform those actions that the original program could have performed. It is likely that the hacker's process will require different privileges than the original program and, hence, it will not achieve the intended goals.

The names of privileges consist of several parts. They start with the letters PV, and the next part of the name should indicate their role. You can check what the privileges allow using the *lspriv -v* command:

```
# lspriv -v
PV_ROOT Allows a process to pass any non-SU privilege check.
PV_AU_ Equivalent to all Auditing privileges (PV_AU_*) combined.
PV_AU_ADD Allows a process to record/add an audit record.
PV_AU_ADMIN Allows a process to configure and query the audit system.
PV_AU_READ Allows a process to read a file marked as an audit file.
# All unnecessary information has been omitted.
PV_PROBEVUE_TRC_KERNEL Allows users to dynamically trace the entire system
PV_PROBEVUE_MANAGE Allows users to update probevue parameters and query all the probevue sessions
PV_FS_ Equivalent to all File System privileges (PV_FS_*) combined.
PV_FS_MOUNT Allows a process to mount and unmount a file system.
# All unnecessary information has been omitted.
```

With the above command, you can check the meaning of the privileges assigned to individual commands.

Another case of using privileges for processes concerns the desire to expand the commands covered by RBAC by adding our own programs. In this case, it is necessary to verify which permissions our program needs in order to run. The solution to this issue is the */usr/sbin/tracepriv* command. This command tracks the execution of a given program and verifies the permissions it uses:

```
# tracepriv -ef passwd testuser
Changing password for "testuser"
testuser's New password:
Enter the new password again:

10027432: Used privileges for /usr/bin/passwd:
  PV_AU_ADD                         PV_AU_PROC
  PV_DEV_CONFIG                     PV_DEV_QUERY
  PV_NET_CNTL                       PV_NET_PORT
```

As you can see in the example above, the *passwd* command requires six privileges. The meanings of those privileges can be verified using the *lspriv* command:

```
# lspriv -v | egrep "PV_AU_ADD|PV_DEV_CONFIG|PV_NET_CNTL|PV_AU_PROC|PV_DEV_QUERY|\
PV_NET_PORT"
PV_AU_ADD Allows a process to record/add an audit record.
PV_AU_PROC Allows a process to get or set an audit state of a process.
PV_DEV_CONFIG Allows a process to configure kernel extensions and devices in the system.
PV_DEV_QUERY Allows a process to query kernel modules.
PV_NET_CNTL Allows a process to modify network tables.
PV_NET_PORT Allows a process to bind to privileged ports.
```

It should be noted that such a simple verification may sometimes prove insufficient. To obtain full information about the required privileges, you would have to verify the invocation of this command under different circumstances and with different parameters. For example, the same *passwd* command with the *-s* flag requires a different set of privileges:

```
# tracepriv -ef passwd -s testuser

10027454: Used privileges for /usr/bin/passwd:
  PV_AU_PROC

 Current available shells:
                /bin/sh
                /bin/bsh
                /bin/csh
                /bin/ksh
                /bin/tsh
                /bin/ksh93
                /usr/bin/sh
                /usr/bin/bsh
                /usr/bin/csh
                /usr/bin/ksh
                /usr/bin/tsh
                /usr/bin/ksh93
                /usr/bin/rksh
                /usr/bin/rksh93
                /usr/sbin/uucp/uucico
                /usr/sbin/sliplogin
                /usr/sbin/snappd
 testuser's current login shell:
                /usr/bin/ksh
 Change (yes) or (no)? > yes
 To?>/bin/ksh

13500858: Used privileges for /usr/bin/chuser:
  PV_AU_ADD                             PV_AU_PROC
  PV_DAC_R                              PV_DAC_W
  PV_FS_CHOWN                           PV_DEV_CONFIG
  PV_DEV_QUERY
```

/etc/security/privfiles and /etc/security/privdevs

The files listed in the title store the descriptions of the authorizations and permissions for the most important files (*privfiles*) and devices (*privdevs*) in the operating system. By default, the *privfiles* file stores the authorizations for approximately 90 files, while the *privdevs* file is empty. Each file is described in a separate section and indicates which authorization will allow for it to be read or written to. This is demonstrated in the following fragment of the */etc/security/privfiles file*:

```
/etc/security/.profile:
        writeauths = aix.security.user

/etc/security/audit/config:
        readauths = aix.security.audit.list
        writeauths = aix.security.audit.config
```

The two parameters that can be defined with each file are:

- ***readauths*** - Indicates the authorization name that allows the reading of a file.
- ***writeauths*** - Indicates the authorization name that allows writing to a file.

Commands related to authorizations

The most important commands related to authorizations are:

```
# lsauth  # Displays authorization information:

# lsauth -f aix.network.config.tcpip
aix.network.config.tcpip:
        id=4050
        dfltmsg=Configure TCPIP Network Interface Parameters
        msgcat=sysauths.cat
        msgset=5
        msgnum=8

# mkauth       # Creates user authorization.

# chauth  # Changes the attributes of a user authorization (system authorizations cannot be
changed).

# rmauth  # Deletes of a given user authorization (system authorizations cannot be changed).
```

Any changes you make in the context of authorizations and roles will not become active until the information from the configuration files is loaded into the KST (*Kernel Security Tables*). The *setkst* command:

```
# setkst
Successfully updated the Kernel Authorization Table.
Successfully updated the Kernel Role Table.
Successfully updated the Kernel Command Table.
Successfully updated the Kernel Device Table.
Successfully updated the Kernel Object Domain Table.
Successfully updated the Kernel Domains Table.
```

To read information about roles and authorizations from the KST, use the *lskst* command:

```
# lskst -t [ auth | role | cmd |dev | dom | domobj ] …
```

- ***auth*** - Authorizations table.
- ***role*** - Roles table, which is equivalent to the contents of the */etc/security/roles* file.
- ***cmd*** - Commands table, which is equivalent to the contents of the */etc/security/privcmds* file.
- ***dev*** - Devices table, which is equivalent to the contents of the */etc/security/privdevs* file.
- ***dom, domobj*** - Domains table (Domain RBAC).

The */etc/security/privfiles* file is not part of the KST; hence, changes to this file do not require re-loading to the *Kernel Security Tables* (*setkst*). To edit files with authorization, use the special version of the *vi* editor, that is, *pvi* (*/usr/bin/pvi* - privileged *vi*).

Examples of using the *lskst* command follow:

```
# lskst -t auth aix.system.boot.create   # Information about the authorization
aix.system.boot.create.
aix.system.boot.create id=7025 dfltmsg=Create Boot Image msgcat=sysauths.cat msgset=9 msgnum=3

# lskst -t cmd -f /usr/bin/mksysb   # Information about the mksysb command.
/usr/bin/mksysb:
        accessauths=aix.system.boot.create
        innateprivs=PV_AZ_ROOT,PV_DAC_R,PV_DAC_W,PV_DAC_X,PV_DAC_O
        inheritprivs=PV_AZ_ROOT,PV_AZ_CHECK,PV_DAC_R,PV_DAC_W,PV_DAC_X,PV_DAC_O,
            PV_DAC_UID,PV_DAC_GID,PV_FS_CHOWN,PV_PROC_SIG,PV_PROC_PRIV,PV_TCB,
            PV_KER_ACCT,PV_SU_UID
        euid=0
        egid=0
        secflags=FSF_EPS

# lskst -t role so   # Informacje dotyczące roli System Operator.
so
authorizations=aix.fs.manage.backup,aix.fs.manage.restore,aix.proc.kill,aix.ras,aix.system.boot.crea
te,aix.system.boot.halt,aix.system.boot.info,aix.system.boot.reboot,aix.system.boot.shutdown,aix.sys
tem.config.init,aix.system.config.wlm rolelist=BackupRestore,SysBoot groups= visibility=1 screens=*
dfltmsg=System Operator msgcat=role_desc.cat msgnum=3 msgset=1 auth_mode=INVOKER id=3
```

Domain RBAC

The use of *Role-Based Access Control* raises the operating system to a higher level of security. You gain the possibility of greater control over permissions as well as the ability to divide tasks within the system into individual roles without abusing the root user. You can limit the use of programs that use the SUID and GUID bits, and you are able to control the specific permissions assigned to the processes being run.

It may happen that the above-mentioned improvements are not sufficient and there remains a need to assign permissions even more accurately. In the case of the functionality discussed thus far, a user who can run the given command is able to take full advantage of its capabilities. He could modify all the structures on which the command works. Domain RBAC allows you to limit these activities to specific structures.

If the operating system supports many applications and you have installed appropriate database and application structures on it, you may not want the same person to be able to operate simultaneously on a database and an application. To stop this from occurring, it is necessary to use RBAC domains.

RBAC domains are enabled by default. Their operation is based on assigning the appropriate domains (tags) to the subject of actions (subject - user, process) and the object of actions (object - a given object in the system, such as a volume group, disk, network interface, or file).

An object can have the following flags set, which define how to verify the access permissions for it:

- ***FSF_DOM_ALL*** - The subject can use the object if it has all the domains that the object has. For example:

User1 has domains *dom1* and *dom2*.
User2 has domain *dom2*.
The object has domains *dom1* and *dom2*.

In the example above, *User1* can use the object because it has all the domains that the object has. However, *User2* cannot use the object because it does not have the *dom1* domain.

- ***FSF_DOM_ANY*** - The subject can use the object if it has any of the domains that the object has.

User1 has domains *dom1* and *dom2*.
User2 has domain *dom2*.
User3 has domain *dom3*.
The object has domains *dom1* and *dom2*.

In the above example, *User1* and *User2* can use the object. However, *User3* cannot because he does not have any of the domains that the object has.

- ***Conflictset*** - Additionally, the object can have conflicting domains defined. This means that if the subject has a domain that the object defines as a conflict domain, then that subject will not have access to the object.

A simple example of using domains to access a file is given below in the RBAC scenarios.

Additional features and tools for working with RBAC

There are several tools available to help you work with RBAC. These are, among others, reporting tools, such as:

- ***rolerpt*** - Displays the configuration from the perspective of a specific role (command permissions, permissions for the files of users who have a given role);

- ***authrpt*** - Displays information from the authorization perspective (command permissions, file permissions, roles that use the given authorization); and

- ***usrrpt*** - Displays the user's RBAC permissions (authorizations, command permissions, file permissions).

RBAC management is also available from the SMIT menu (Figure 14-2), from which you can perform most of the actions available from the command line *(#smit rbac)*:

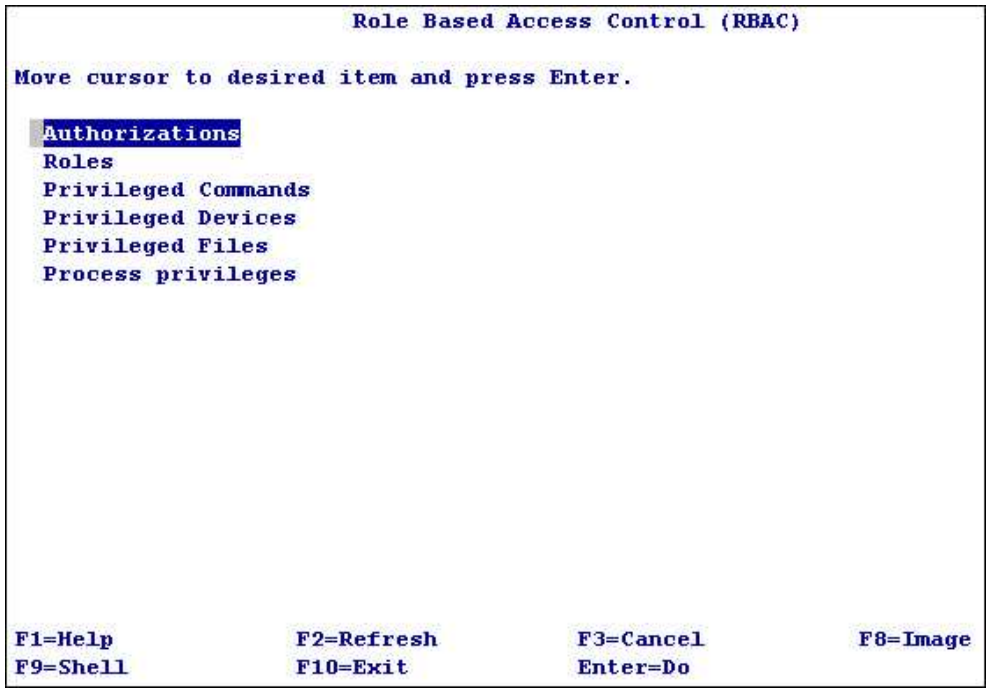

Figure 14-2: SMIT - RBAC.

RBAC allows the implementation of additional security in the form of prohibiting the execution of a specific command in the system by a single user. This mechanism is known as multi-user authentication. With it, you can configure a given command in such a way that before it is run, it will have to be approved by another user (a maximum of 16 approvals may be required).

The above feature is mainly used in systems that require exceptional levels of supervision, since it significantly reduces the comfort of working with the system. However, it allows you to avoid unauthorized or not fully thought out actions.

The following command is used to enable this mechanism:

```
# setsecattr -c authroles=[roles confirming the execution] path_to_command
```

To run the command covered by the multi-user authentication mechanism, the *authexec* command is used (attempting to run without *authexec* will fail):

```
# authexec path_to_command
```

RBAC scenarios

Using the SA (System Administrator) role:

We verify the authorizations held by the *sa* role:

```
# lsrole sa
sa
authorizations=aix.system.config.acct,aix.system.config.cron,aix.system.config.src,aix.system.instal
l rolelist=FSAdmin,AccountAdmin groups= visibility=1 screens=* dfltmsg=System Administrator
msgcat=role_desc.cat msgnum=2 msgset=1 auth_mode=INVOKER id=2
```

User1 attempts to create a file system. This is a regular user with no roles. The attempt fails due to the lack of relevant permissions:

```
# su - User1
$ crfs  -v jfs  -g  rootvg  -m /test -a size=32768
sh: /usr/sbin/mklv: 0403-006 Execute permission denied.
crfs: 0506-973 Cannot create logical volume for log device.
$ exit
```

We assign the *System Administrator* role to *User1*:

```
# chuser roles=sa User1
```

We load the changes to the KST (*Kernel Security Tables*). Without this step, the user would not be able to use the new permissions:

```
# setkst
Successfully updated the Kernel Authorization Table.
Successfully updated the Kernel Role Table.
Successfully updated the Kernel Command Table.
Successfully updated the Kernel Device Table.
Successfully updated the Kernel Object Domain Table.
Successfully updated the Kernel Domains Table.
```

We switch to *User1* and check the active roles (no active roles).

```
# su - User1
$ rolelist -ea
rolelist: 1420-062 There is no active role set.
```

The user then activates the *System Administrator* role. From now on, he can use the permissions assigned to that role:

```
$ swrole sa
User1's Password:   # Role activation requires a user password.
$ rolelist -ea   # Verification of the active roles (-e) and authorization (-a).
sa              aix.fs.manage.change
                aix.fs.manage.create
                aix.fs.manage.debug
                aix.fs.manage.defrag
                aix.fs.manage.dump
                aix.fs.manage.list
                aix.fs.manage.mount
                aix.fs.manage.quota
                aix.fs.manage.recover
                aix.fs.manage.remove
                aix.fs.manage.snapshot
                aix.fs.manage.unmount
                aix.fs.object
                aix.lvm
                aix.security.group
                aix.security.user
                aix.system.config.acct
```

```
                aix.system.config.cron
                aix.system.config.src
                aix.system.install
```

The user again tries to create a file system. This time the attempt is successful. The user has used the System Administrator role privileges:

```
$ crfs  -v jfs  -g  rootvg  -m /test -a size=32768
Based on the parameters chosen, the new /test JFS file system
is limited to a maximum size of 134217728 (512 byte blocks)

New File System size is 32768
```

Create a role that allows access in write mode to the /etc/syslog.conf file:

We start with the creation of the *SyslogConfUpd* authorization. The corresponding record appears in the */etc/security/authorizations* file. There are no "system" authorizations in this file, although it does contain all the authorizations created by the user.

```
# mkauth dfltmsg="Update Syslog conf" SyslogConfUpd
# lsauth SyslogConfUpd
SyslogConfUpd id=10024 dfltmsg=Update Syslog conf
```

We create the *SyslogRole* role and assign the *SyslogConfUpd* authorization to it. The corresponding record appears in the */etc/security/roles* file:

```
# mkrole dfltmsg="Role Update Syslog conf" authorizations=SyslogConfUpd SyslogRole
# lsrole SyslogRole
SyslogRole authorizations=SyslogConfUpd rolelist= groups= visibility=1 screens=* dfltmsg=Role Update
Syslog conf msgcat= auth_mode=INVOKER id=21
```

We create the appropriate entry in the */etc/security/privfiles* configuration file. This entry gives write permission to the */etc/syslog.conf* file for the *SyslogConfUpd* authorization.

```
# setsecattr -f writeauths=SyslogConfUpd /etc/syslog.conf
# grep -p /etc/syslog.conf /etc/security/privfiles
/etc/syslog.conf:
        writeauths = SyslogConfUpd
```

We assign the *SyslogRole* role to *User2*.

```
# chuser roles=SyslogRole User2
# setkst   # Loading changes to the KST.
Successfully updated the Kernel Authorization Table.
Successfully updated the Kernel Role Table.
Successfully updated the Kernel Command Table.
Successfully updated the Kernel Device Table.
Successfully updated the Kernel Object Domain Table.
Successfully updated the Kernel Domains Table.
```

We log in as *User2* and activate the *SyslogRole* role:

```
# su - User2
$ swrole SyslogRole
User2's Password:
```

User2 has no direct permissions to modify the file, as shown in the following output. However, with the given authorization, he can edit this file. The user can only edit the file using the *pvi* command (*vi* editor adapted to work with RBAC). Other commands will not be able to use this write permission.

```
$ ls -l /etc/syslog.conf
-rw-r--r--    1 root     system         4798 Sep 13 04:41 /etc/syslog.conf
$ pvi /etc/syslog.conf
```

The use of RBAC domains to restrict access to a file:

As a first step, we create a file on which we will test the Domain RBAC and create the *dom1* domain:

```
# echo "Domain file" >/tmp/dfile
# chmod 777 /tmp/dfile
# mkdom dom1
# lsdom dom1
dom1 id=1
```

We assign the *dom1* domain to the */tmp/dfile* file with the *FSF_DOM_ALL* flag (this means that the user who wants to access the file must have all the domains that have the file):

```
# setsecattr -o domains=dom1 secflags=FSF_DOM_ALL objtype=file /tmp/dfile
# lssecattr -o /tmp/dfile
/tmp/dfile domains=dom1 objtype=file secflags=FSF_DOM_ALL
```

We load the domain information into the KST:

```
# setkst
Successfully updated the Kernel Authorization Table.
Successfully updated the Kernel Role Table.
Successfully updated the Kernel Command Table.
Successfully updated the Kernel Device Table.
Successfully updated the Kernel Object Domain Table.
Successfully updated
```

User1 tries to read the file. Although he has full rights (rwx) to the file, he is not able to read it because he does not have the *dom1* domain:

```
# su - User1
$ cat /tmp/dfile
cat: 0652-050 Cannot open /tmp/dfile.
$ exit
```

We assign the *dom1* domain to the user.

```
# chuser domains=dom1 User1
```

In the next attempt, the user reads the file. The user now has the *dom1* domain, which is the same as the file, and meets the conditions imposed by the *FSF_DOM_ALL* flag:

```
# su - User1
$ cat /tmp/dfile
Domain file
```

Auditing

The AIX audit subsystem allows you to monitor the broadly defined activity within the operating system. It allows the logging of approximately 500 different classes of system events that relate to all aspects of the system operation (users, authentication, processes, files, data transfer, devices, and many others). In addition, for each file that is important from the security perspective, you can define events to log its read, write, or execution.

Due to the large number of potentially audited events, it is important to choose their number in a reasonable fashion. Choosing too many events can lead to a heavy load on the system, which will result it in spending too much time logging them. At the same time, this will lead to a rapid increase in the number of logs. In the best-case scenario, despite the large amount of information gathered, it may prove useless, since the excess of information will be very difficult to verify. In the worst-case scenario, it will overload the file system with audit information and hence adversely affect the performance of daily operations.

Starting the auditing process is not a complicated task and requires only a few steps:

1. Determine which events will be monitored.
2. Determine whether the audit subsystem will work in binary mode, collecting data for later analysis, or in streaming mode, by processing these data on an ongoing basis and performing defined actions such as forwarding to further analytical tools.
3. Reflect this information in the configuration files.
4. Start the audit subsystem.

The details of the above steps will become clearer when you read further information in this chapter.

Configuration files

The audit subsystem is based on several files located in the */etc/security/audit* directory. These files are *config*, *events*, and *objects*.

/etc/security/audit/events - Contains about 500 defined events that can be monitored. The events are described in the *event_name = logged_information_format* format. A fragment of the file contents:

```
auditpr[3]:
*       shmget()
        SHM_Create = printf "key: %d size: %ld flags: %o shmid: %d"
*       adjtime()
        PROC_Adjtime = printf "old time: %T, delta: %d:%d"
*       settimer()
        PROC_Settimer = printf "old time: %T, new time: %T"
*       mknod()
        FILE_Mknod = printf "mode: %o dev: %D filename %s"
*       mknodat()
        FILE_Mknodat = printf "mode: %o dev: %D filename %s dirfd %d dirpath %s"
*       mkdev
```

```
        DEV_Create = printf "mode: %o dev: %D filename %s"
        DEV_Start = printf " %s "

auditpr:
* kernel proc events
*       fork()
        PROC_Create = printf "forked child process %d"
*       exit()
        PROC_Delete = printf "exited child process %d, rc: %d, filename: %s"
        EXIT_Used_Privs = printf "pid: %d, ppid: %d, used:
# All unnecessary information has been omitted.
*       /etc/security/login.cfg
        S_LOGIN_WRITE = printf "%s"

*       /etc/security/passwd
        S_PASSWD_READ = printf "%s passwd read"
# All unnecessary information has been omitted.
```

Let's consider the operation of the above entries using the example below:

```
PROC_Create = printf "forked child process %d"
```

This entry applies to the creation of a process. It writes general information to the audit log in the form of the event time, the user responsible for the event, and the execution status, as well as information in the form of text "*forked child process %d*." Further, "*%d*" means that the value will be inserted here, in this case the number of the created process, formatted as a 32-bit decimal value. Examples of the possible formats are shown in Table 14-2.

Table 14-2: Auditing - Formatting values in the /etc/security/audit/events file.

Format	Description
%s	Text
%d	32-bit decimal value
%D	Major and minor number of the device
%T	Date and time in the format: DD Mmm YYYY HH: MM: SS: mmmuuu
…	…
More formats can be found in the system documentation: *man events*	

/etc/security/audit/objects - Contains descriptions of the objects (files) to be covered by auditing. The objects are described in the *access_mode = event_name* format. The access mode can be either *r* - read or *w* - write. *Event_name* indicates that the event is described in the */etc/security/audit/events* file.

Examples of default entries in the objects file after a fresh operating system installation:

```
/etc/security/environ:
        w = "S_ENVIRON_WRITE"
/etc/security/group:
        w = "S_GROUP_WRITE"
/etc/security/limits:
        w = "S_LIMITS_WRITE"
/etc/security/login.cfg:
        w = "S_LOGIN_WRITE"
/etc/security/passwd:
        r = "S_PASSWD_READ"
        w = "S_PASSWD_WRITE"
/etc/security/user:
```

```
        w = "S_USER_WRITE"
/etc/security/audit/config:
        w = "AUD_CONFIG_WR"
```

In the example entry given above regarding the */etc/security/passwd* file, you can see that the file is being monitored. In the case of reading, an event named *S_PASSWD_READ* (*r = "S_PASSWD_READ"*) is generated, while in the case of writing to a file, an event named *S_PASSWD_WRITE* (*w = "S_PASSWD_WRITE"*) is generated. These events are reflected in the */etc/security/audit/events* file, where their formatting is defined:

```
*       /etc/security/passwd
        S_PASSWD_READ = printf "%s"
*       /etc/security/passwd
        S_PASSWD_WRITE = printf "%s"
```

In the above fragment, *%s* means the path to the file, for example, the */etc/security/passwd* path, will appear in the log. You can audit any files, that is, both system files and application files, creating an event and adding relevant entries in the */etc/security/audit/objects* and */etc/security/audit/events* files.

/etc/security/audit/config is the main configuration file that specifies how auditing works. Its contents are divided into several sections. Here's the relevant part of the file:

```
start:
        binmode = on
        streammode = off

bin:
        trail = /audit/trail
        bin1 = /audit/bin1
        bin2 = /audit/bin2
        binsize = 10240
        cmds = /etc/security/audit/bincmds
        freespace = 65536
        backuppath = /audit
        backupsize = 0
        bincompact = off
stream:
        cmds = /etc/security/audit/streamcmds
        streamcompact = off

classes:
        general = USER_SU,PASSWORD_Change,FILE_Unlink,FILE_Link,FILE_Rename,FS_Chdir,
FS_Chroot,PORT_Locked,PORT_Change,FS_Mkdir,FS_Rmdir
        objects = S_ENVIRON_WRITE,S_GROUP_WRITE,S_LIMITS_WRITE,S_LOGIN_WRITE,
S_PASSWD_READ,S_PASSWD_WRITE,S_USER_WRITE,AUD_CONFIG_WR
        SRC = SRC_Start,SRC_Stop,SRC_Addssys,SRC_Chssys,SRC_Delssys,SRC_Addserver,
SRC_Chserver,SRC_Delserver
        kernel = PROC_Create,PROC_Delete,PROC_Execute,PROC_RealUID,PROC_AuditID,
PROC_RealGID,PROC_Environ,PROC_Limits,PROC_SetPri,PROC_Setpri,PROC_Privilege,PROC_Settimer
        files = FILE_Open,FILE_Read,FILE_Write,FILE_Close,FILE_Link,FILE_Unlink,
FILE_Rename,FILE_Owner,FILE_Mode,FILE_Acl,FILE_Privilege,DEV_Create,File_copy
# All unnecessary information has been omitted.
users:
        root = general
role:
```

The *start* section - Describes how auditing is supposed to work. There are two modes: binary (*bin*) and streaming (*stream*). They can be turned on simultaneously. The binary mode stores events in a more concise form, which results in less disk space usage. It is used in the case of the verification of logs at a

later date. The streaming mode can be used to take immediate action in the event of a threat or anomaly being detected. It processes the stream of audit data on an ongoing basis.

The *bin* section - Describes the way of auditing in binary mode. Audit data are saved alternately in files defined as *bin1* and *bin2*. If the currently used bin file reaches the size defined as the *binsize* or the audit subsystem is restarted, then auditing starts using the second *bin* file. The contents of the first file is copied to the *trail* file and then deleted. The command defined in the *cmds* parameter is used to perform the above operations.

The freespace option specifies the amount of free space in the file system in which the trail file is located. If the free space drops below this point, an alarm will be generated (ie, *Syslog* will send a message).The *backuppath* option specifies the place where the *trail* file should be copied if it reaches the size of the *backupsize* (the value *0* means no copying).

The *stream* section - Indicates the command to process events that appear. By default, this is the entry *cmds = /etc/security/audit/streamcmds*, with the following content:

```
# cat /etc/security/audit/streamcmds
/usr/sbin/auditstream | auditpr > /audit/stream.out &
```

The above configuration causes the immediate recording of the event to the */audit/stream.out* file in a textual form. However, the mode of operation can be easily modified in the context of existing needs. You can, for example, send information to a syslog log server, send an e-mail, send a text message (redirecting information to applications that support a given functionality), notify another monitoring system that operates 24 hours a day, and even take preventive actions, such as blocking a specific user. The method of operation depends on the creativity of the script developer or the program that handles the events.

The *classes* section - Defines the event classes, thereby making it easier to use them. With the classes, you do not have to assign many individual events to a given user, but you can assign a defined class of events. This approach is clearer from the user's point of view. The pattern is in the form *class_name = events_from_events_file*. By default, several classes are defined. You can modify them and create new ones.

The *users* section - Defines which users should be audited and which audit classes should be used in the case of a given user. If you assume that each newly created user should be audited, you can add the appropriate entry to the */usr/lib/security/mkuser.default* file in the user section. For example, *auditclasses = general*.

The *roles* section - Works in the same way as the *users* section. It defines which roles will be audited and what class of events will be used. You can find more information about the various roles in the section describing RBAC.

By default, there is no WPAR section in the */etc/security/audit/config* file. However, by adding such a section, you can also audit WPARs. To add this section:

```
WPARS:
<wpar_name> = <auditclass>, ... <auditclass>
```

Commands

The audit subsystem has several commands that you can use. The main command is the *audit*. This works with several flags:

```
audit {on|off|query|start|shutdown}
```

- ***audit start/shutdown*** - Start or shutdown auditing. During startup, the configuration file is loaded. Therefore, to load a new configuration, you should shutdown the audit, and then restart it. After the audit shutdown is completed, the subsystem rewrites the audit information from the *bin* file to the *trail* file. Once the audit is started again, the *bin* files are empty.

- ***audit on/off*** - Suspend and resume auditing. In this case, you should remember that the changes you made to the configuration file will not be loaded into the audit subsystem after it resumes its work. You must perform the *shutdown/start* operation to achieve this goal.

- ***audit query*** - Displays the current status and configuration of the *audit* subsystem.

Another important command is *auditpr*. This reads the audit data from files, whether in binary (*bin* or *trail*) or streaming mode, and displays them on a standard output (usually a screen). An example of reading a *trail* file (a column pointing to the WPAR has been cut from the command result in order to make the output more readable) follows:

```
#auditpr -v < /audit/trail

event           login    status      time                     command
--------------- -------- ----------- ------------------------ ------------------------
S_PASSWD_READ   root     OK          Thu Sep 08 04:05:00 2016 cron
        audit object read event detected /etc/security/passwd passwd read
CRON_Start      root     OK          Thu Sep 08 04:05:00 2016 cron
        event = start cron job cmd = /usr/sbin/dumpctrl -k >/dev/null 2>/dev/null time = Thu Sep  8
04:05:00 2016
FS_Chdir        root     OK          Thu Sep 08 04:05:00 2016 cron
change current directory to: /
FILE_Unlink     root     OK          Thu Sep 08 04:05:00 2016 compress
        filename /audit/tempfile.08388872
FS_Chdir        root     OK          Thu Sep 08 04:15:00 2016 cron
PASSWORD_Change root     OK          Wed Sep 07 08:19:41 2016 passwd
```

The data presented in the above output include:

- ***Event*** - Name of the event defined in the */etc/security/audit/events* file;
- ***Login*** - The user who generated the event;
- ***Status*** - The status of the event, that is, whether it was successful;
- ***Time*** - The date and time of the action; and
- ***Command*** - The program that triggered the event.

In addition, you can see the specific entries under the commands. The entries result from the format

lines in the */etc/security/audit/events* file.

```
/etc/security/audit/events:
*       /etc/security/passwd
        S_PASSWD_READ = printf "%s passwd read"

auditpr -v < /audit/trail:
event           login    status      time                     command
--------------- -------- ----------- ------------------------ ----------------------
S_PASSWD_READ   root     OK          Thu Sep 08 04:05:00 2016 cron
        audit object read event detected /etc/security/passwd passwd read
```

AIXpert (AIX Security Expert)

AIX Security Expert is a mechanism that allows the automation of operating system hardening by means of implementing parameters in accordance with one of the standards (patterns) prepared by IBM. You can adjust the settings patterns to suit your needs. The same patterns can be used on many systems, thereby unifying the security configuration. You can report inconsistencies between the patterns and the actual system settings at any time. In addition to the typical security settings, the use of IBM-supplied patterns can trigger additional mechanisms:

- Auditing - Improves the accountability of activities within the system.
- Packet filtering - A firewall that prevents, among others, port scanning threats.
- Dictionary of typical passwords - Implements methods for the verification of user-created passwords based on a dictionary. It prevents the use of passwords that are easy to guess or break via a dictionary attack.
- Limiting the number of programs with SUID and SGID - Limits the number of programs with the SUID and SGID bits using the *fpm* command.

With the installation of the operating system, you receive several security patterns. In some cases, you can implement one of them, although they are not usually used in exactly the form IBM provides. Most companies have their own security standards, which change periodically. Therefore, they use patterns modified to suit their own needs, which are continuously adapted to changes that occur in the company's security standards. Nevertheless, a number of security standards (levels) patterns are delivered with the system:

- ***Default*** - The default security level. It should be implemented after the installation of the system. It is the starting point for further changes. Before proceeding to the first implementation of the target security level, it is recommended that you first set the system to the *default* level in order to verify the correctness of the mechanisms. When new changes are made from the default level, there is a slightest risk that some changes will not be made correctly.
- ***Low*** - A low level of security. The only advantage is that its implementation does not require a system reboot, and it should not affect the current work of users.
- ***Medium*** - A medium level of security. It goes a step further than the low level, although it still

allows the use of unsafe Telnet or FTP services that transmit user passwords in an unencrypted form. It also enables the basic function of packet filtering (firewall).

- ***High*** - A high level of security. Disables potentially unsafe services, sets strong criteria for user passwords, and sets strong filters by blocking multiple ports, although it does not block active application ports. When setting this level, ensure that the applications used are running. Otherwise, the ports normally used by the applications can be blocked.
- ***Secure by Default*** - A set of settings implemented during installation, so long as the *Secure by Default* option is selected. You can also implement these settings after installation, but in that case, you will not be able to fully utilize this mode, since the *Secure by Default* mode selected during installation limits the number of packages installed.
- ***SOX-COBIT*** - Implements additional settings related to compliance with the Sarbanes-Oxley Act, which was promulgated in 2002 by the United States Congress.

Most important files related to AIXpert

The files related to AIXpert are located in the */etc/security/aixpert* directory. The most important files and catalogs:

- ***/etc/security/aixpert/core*** - There are *.xml* files in this directory that describe the configuration for each security level described above.
- ***/etc/security/aixpert/core/aixpertall.xml*** - Contains configuration information for the security levels *default*, *low*, *high*, and *SOX-COBIT*. The description of the configuration parameters is based on information that can be found in the dedicated paragraphs of the *.xml* file. The most important paragraphs in relation to understanding the mechanism are:
  - ***AIXPertEntry*** - The beginning of a section that features its name and function.
  - ***AIXPertRuleType*** - The rule type. It has several tags that correspond to the defined security levels:
    - *HLS (High Level Security)* - High security level.
    - *MLS (Medium Level Security)* - Medium security level.
    - *LLS (Low Level Security)* - Low security level.
    - *DLS (Default Level Security)* - Default security level.
    - *SCBPS (SOX-COBIT)* - Compliance with SOX-COBIT.
    - *Prereq* - Rules that verify the applicability of given changes.
  - ***AIXPertDesctiption*** - Description of the given rule.
  - ***AIXPertPrereqList*** - List of prerequisites - Requirements that the system must meet in order to apply the rule. Usually, there is a list of filesets required to implement the rule. A script that verifies the ability to make changes may also be used.
  - ***AIXPertCommand*** - A script run to implement the rule. The scripts used by AIXpert are stored in the */etc/security/aixpert/bin* directory.
  - ***AIXPertArgs*** - Arguments passed to the above script.
  - ***AIXPertGroup*** - The group to which the given settings belong. It is a descriptive parameter for the better orientation in the rules.

To illustrate the meanings of the individual sections, let's discuss the rule concerning the *minlen* parameter, which specifies the minimum length of the password. Part of the output from the *aixpertall.xml* file regarding the *minlen* setting follows:

```
</AIXPertEntry>
<AIXPertEntry name="hls_minlen" function="minlen">
    <AIXPertRuleType type="HLS"/>
    <AIXPertDescription catalog="aixpert.cat" setNum="101" msgNum="24">Minimum length
     for password: Specifies the minimum length of a password to 8.
    </AIXPertDescription>
    <AIXPertPrereqList>bos.rte.date,bos.rte.commands,bos.rte.security,     I
    bos.rte.shell,bos.rte.ILS</AIXPertPrereqList>
    <AIXPertCommand>/etc/security/aixpert/bin/chusrattr</AIXPertCommand>
    <AIXPertArgs>minlen=8 ALL hls_minlen</AIXPertArgs>
    <AIXPertGroup>Password policy rules</AIXPertGroup>
 </AIXPertEntry>
```

The above rule applies to the high security level (*<AIXPertRuleType type = "HLS" />*), so it will be used when you set that level:

```
# aixpert -l high
```

The rule is processed as follows:

1. AIXpert checks whether the rule can be implemented. It can be implemented if the requirements listed in the *<AIXPertPrereqList>* section are met, that is, the operating system must have the following filesets installed: *bos.rte.date*, *bos.rte.commands*, *bos.rte.security*, *bos.rte.shell*, and *bos.rte.ILS*.

2. The script that sets this rule is run (section *<AIXPertCommand>*). The data from the *<AIXPertArgs>* section are the parameters of the script. Thus, the following command is run:

```
# /etc/security/aixpert/bin/chusrattr minlen=8 ALL hls_minlen
```

3. The change that has been made is recorded to the */etc/security/aixpert/core/appliedaixpert.xml* file, while the value that the parameter had prior to the change is added to the */etc/security/aixpert/core/undo.xml* file. This allows you to undo your changes.

The described file features several versions of the rule for the *minlen* parameter. To illustrate this fact, below is a rule for the default security level that sets the *minlen* parameter to *0* (i.e., no password length requirements).

```
</AIXPertEntry>
  <AIXPertEntry name="dls_minlen" function="minlen">
    <AIXPertRuleType type="DLS"/>
    <AIXPertDescription catalog="aixpert.cat" setNum="101" msgNum="25">Minimum length
     for password: Removes the minimum length constraint on
     password.</AIXPertDescription>
    <AIXPertPrereqList>bos.rte.date,bos.rte.commands,bos.rte.security,bos.rte.shell,
     bos.rte.ILS</AIXPertPrereqList>
    <AIXPertCommand>/etc/security/aixpert/bin/chusrattr</AIXPertCommand>
    <AIXPertArgs>minlen=0 ALL dls_minlen</AIXPertArgs>
    <AIXPertGroup>Password policy rules</AIXPertGroup>
  </AIXPertEntry>
```

Most settings occur at every security level (*default*, *high*, *medium*, and *low*). The same is true with the *minlen* setting. A separate element is the *prereq* setting, which verifies whether the changes can be made. An example of a *prereq* rule:

```
<AIXPertEntry name="prereqrl" function="prereqrl">
  <AIXPertRuleType type="Prereq"/>
  <AIXPertDescription catalog="aixpert.cat" setNum="101" msgNum="8">Prereq rule for
   root login: Checks whether any non-root user exists that has login
   privileges.</AIXPertDescription>
  <AIXPertPrereqList/>
  <AIXPertCommand>/etc/security/aixpert/bin/prereqrl</AIXPertCommand>
  <AIXPertArgs/>
  <AIXPertGroup/>
```

The above rule verifies whether there is any user (except the *root* user) who has the possibility of logging in to the system. It verifies this by running the */etc/security/aixpert/bin/prereqrl* script, which should return *0* in the case of a positive verification. The above rule is included in the rule below, which blocks the ability to log in directly as a *root* user. The checking rule is placed in the *<AIXPertPrereqList>* section (*prereqrl*):

```
<AIXPertEntry name="mls_rootrlogin" function="rootrlogin">
  <AIXPertRuleType type="MLS"/>
  <AIXPertDescription catalog="aixpert.cat" setNum="101" msgNum="64">Remote root
   login: Disables remote root login.</AIXPertDescription>
   <AIXPertPrereqList>bos.rte.date,bos.rte.commands,bos.rte.security,bos.rte.shell,
   bos.rte.ILS,prereqrrl</AIXPertPrereqList>
  <AIXPertCommand>/etc/security/aixpert/bin/chuserstanza</AIXPertCommand>
  <AIXPertArgs>/etc/security/user rlogin=false root mls_rootrlogin</AIXPertArgs>
  <AIXPertGroup>Login policy recommendations</AIXPertGroup>
```

Further parts of the files that are key for AIXpert:

- ***/etc/security/aixpert/core/SbD.xml*** - Contains the parameter configuration for the *Secure by Default* security level.
- ***/etc/security/aixpert/core/appliedaixpert.xml*** - Contains the last configuration changes that have been made using AIXpert.
- ***/etc/security/aixpert/core/undo.xml*** - The opposite of *appliedaixpert.xml*. Contains the old parameter values that were used prior to the last change. It can be used to roll back changes.
- ***/etc/security/aixpert/undo*** - Directory, contains scripts described in *undo.xml* and used to roll back changes made using AIXpert.
- ***/etc/security/aixpert/bin*** - Contains scripts used in rules intended to modify the system parameters.
- ***/etc/security/aixpert/log/FAILEDRULES.log*** - Information about the rules that the implementation failed.
- ***/etc/security/aixpert/log/PASSESRULES.log*** - Information about the rules that the implementation was successful.

AIXpert can use the LDAP as the source of configuration data. In this case, the LDAP server must have a schema deployed, which is stored in the */etc/security/ldap/sec.ldif* file. *Aixpertldap* is a special command for implementing and verifying rules in such a case.

Tools for managing AIXpert

The main tool available for managing the system security levels is the *aixpert* command:

```
# aixpert
Usage:  aixpert -l {high|medium|low|default|sox-cobit} [-p] [{-n|-a} -o File]
        aixpert -l {h|m|l|d|s} [-p] [{-n|-a} -o File]
        aixpert -f File1 [-p] [-a -o File2]
        aixpert -c [-p] [-r|-R] [-P File] [-l Level]
        aixpert -u [-p]
        aixpert -d
        aixpert -t
```

Examples:

```
# Setting a high level of security:
# aixpert -l high

# Setting custom security parameters:
# aixpert -f /path_to_xml_file

# Checking that the applied parameters have not changed (based on appliedaixpert.xml):
# aixpert -c
```

Some of the operations related to AIX Security Expert can be made from the SMIT menu (Figure 14-3):

```
#smit aixpert
```

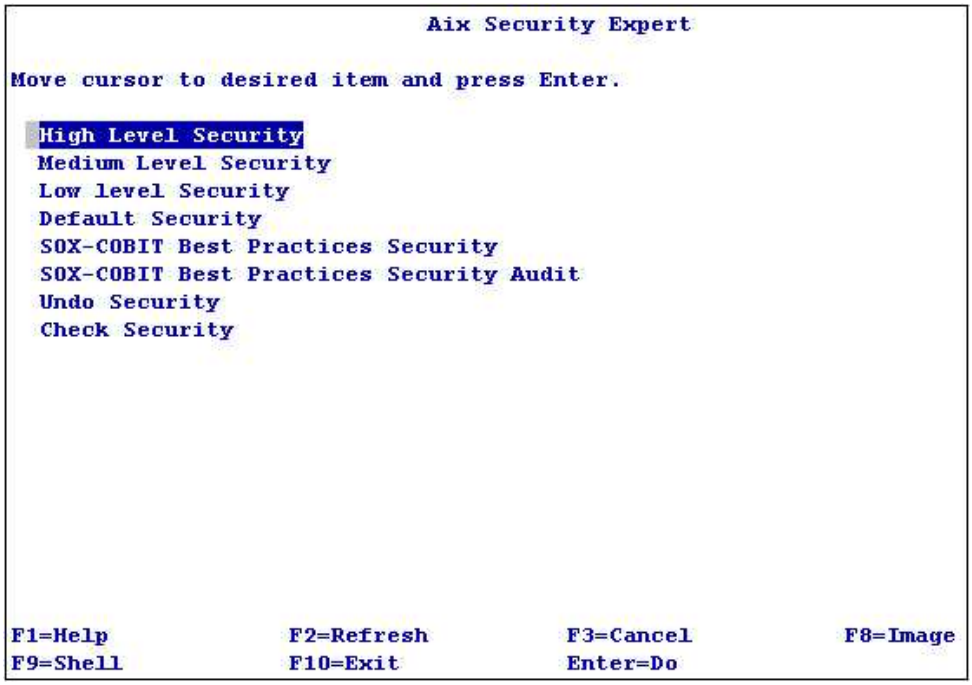

Figure 14-3: SMIT - AIXpert.

The FPM (*File Permission Manager*) is a tool associated with AIXpert. It is used to modify the SUID and SGID bits of specific command sets. This tool is called by AIXpert when setting defined security levels. This is demonstrated in the following fragment of the *aixpertall.xml* file. It shows the call to the */etc/security/aixpert/bin/filepermgr* script with parameters that indicate a low security level. This script in turn calls the *fpm -l low* command and sets the SUID and GUID bits according to the low security definition from the FPM perspective.

```
<AIXPertEntry name="lls_filepermgr" function="filepermgr">
   <AIXPertRuleType type="LLS"/>
   <AIXPertDescription catalog="aixpert.cat" setNum="101" msgNum="155">File
    Permissions Manager: Runs fpm command with low option to remove setuid from
    privileged commands.</AIXPertDescription>
    <AIXPertPrereqList>prereqnontcb,bos.rte.date,bos.rte.commands,bos.rte.security,
     bos.rte.shell,bos.rte.ILS</AIXPertPrereqList>
    <AIXPertCommand>/etc/security/aixpert/bin/filepermgr</AIXPertCommand>
    <AIXPertArgs>l lls_filepermgr</AIXPertArgs>
    <AIXPertGroup>Disable SUID of commands</AIXPertGroup>
  </AIXPertEntry>
```

The *File Permission Manager* can also be used independently of AIXpert. It is started using the *fpm* command. It has, similar to AIX Security Expert, defined security levels on which it operates. These levels indicate the files from which to remove the SUID and SGID bits. The FPM has four security levels:

- ***Low*** - Removes the SETUID and SETGID bits from the files listed in */etc/security/fpm/data/med_fpm_list* as well as those listed in the files located in the

/usr/lib/security/fpm/custom/med directory. This is the same as the *medium* level.

- ***Medium*** - Removes the SETUID and SETGID bits from the files listed in */etc/security/fpm/data/med_fpm_list* as well as those listed in the files located in the */usr/lib/security/fpm/custom/med* directory. This is the same as the *low* level.

- ***High*** - Removes the SETUID and SETGID bits from the files listed in */etc/security/fpm/data/high_fpm_list* as well as those listed in the files located in the */usr/lib/security/fpm/custom/high* directory.

- ***Default*** - Returns to the default settings.

All the changes made by the FPM are logged to the */var/security/fpm/log/date_time* file. In addition to the levels defined by the operating system provider, you can use the *-f file_name* flag and specify files from a list of files to change their SUID and SGID bits according to your requirements.

Examples of using the *fpm*:

```
# fpm -l high  # High level setting.
# fpm -l default    # Return to the default level.
# fpm -l high -p    # Verification of the changes that will be made by FPM without implementing
them.
```

AIXpert scenarios

Setting the medium security level:

```
# aixpert -l medium
do_action(): rule(mls_tcbupdate): warning.
do_action(): Warning: Prereq failed for prereqtcb
do_action(): rule(mls_rootrlogin): warning.
do_action(): Warning: Prereq failed for prereqrrl
do_action(): rule(mls_xhost): warning.
do_action(): Warning: Prereq failed for X11.Dt.ToolTalk
do_action(): rule(mls_ISSServerSensorFull): warning.
do_action(): Warning: Prereq failed for prereqRSSSFull
do_action(): rule(mls_ISSServerSensorLite): warning.
do_action(): Warning: Prereq failed for prereqRSSSLite
Processedrules=83       Passedrules=78  PrereqFailedrules=5     Failedrules=0   Level=MLS
        Input file=/etc/security/aixpert/core/aixpertall.xml
```

As you can see above, some 78 out of 83 rules have been implemented. Five rules did not pass the verification process. After checking the causes of the failures, it appears that the filesets listed in the *<AIXPertPrereqList>* paragraph of those rules are missing. If the implementation of any rules were to fail, then information concerning those rules would be available in the */etc/security/aixpert/log/FAILEDRULES.log* file.

After implementing the medium security level, the following files appeared:

- ***/etc/security/aixpert/core/appliedaixpert.xml*** - A file that reflects the changes introduced by AIXpert. From now on, when verifying changes to the security settings, the

aixpert -c command will check the rules described in this file.

- ***/etc/security/aixpert/core/undo.xml*** - A file that reflects the values that particular parameters had prior to setting the medium security level. The file can be used to roll back changes.

Verification of the security level that has been implemented:

```
# aixpert -t
Applied Profiles:MLS
```

Checking the system's compliance with the implemented security level:

A medium level of security (MLS) is implemented. We can verify whether the implemented security parameters have been changed. The verification is performed using the */etc/security/aixpert/core/appliedaixpert.xml* file:

```
# aixpert -c -r
Processedrules=78       Passedrules=78  Failedrules=0   Level=MLS
        Input file=/etc/security/aixpert/core/appliedaixpert.xml
```

As you can see, the rules have not been changed. To illustrate how the system will behave in the case of discrepancies, we change the *minlen* parameter in the */etc/security/users* file from the value *8* to the value *6*. Then, we perform a second check:

```
# aixpert -c -r
Processedrules=78       Passedrules=77  Failedrules=1   Level=MLS
        Input file=/etc/security/aixpert/core/appliedaixpert.xml
```

One rule has a failed status. We check what this rule is:

```
# cat /etc/security/aixpert/log/FAILEDRULES.log
do_action(): rule(mls_minlen_0D31FE29) : failed.
```

It is the *mls_minlen_0D31FE29* rule. After its verification in the */etc/security/aixpert/core/appliedaixpert.xml* file, you can see that it applies to the *minlen* parameter, which we have changed:

```
<AIXPertEntry name="mls_minlen_0D31FE29" function="minlen">
        <AIXPertRuleType type="MLS"/>
        <AIXPertDescription catalog="aixpert.cat" setNum="101" msgNum="24">
         Minimum length for password: Specifies the minimum length of a passwo
         rd to 8.</AIXPertDescription>
        <AIXPertPrereqList>bos.rte.date</AIXPertPrereqList>
        <AIXPertCommand>/etc/security/aixpert/bin/chusrattr</AIXPertCommand>
        <AIXPertArgs>minlen=8 ALL mls_minlen</AIXPertArgs>
        <AIXPertGroup>Password policy rules</AIXPertGroup>
  </AIXPertEntry>
```

You can manually fix the discrepancy or re-implement the given security level.

Rolling back the changes:

A medium level of security is applied. We want to roll back this implementation:

```
# aixpert -t   # A medium level of security is implemented.
Applied Profiles: MLS

# aixpert -u   # Undo the changes based on the undo.xml file.
process_undo_entry(): entry name=mls_usrck_B43BD197 not found
process_undo_entry(): entry name=mls_pwdck_B43BD197 not found
process_undo_entry(): entry name=mls_grpck_B43BD197 not found
process_undo_entry(): entry name=mls_dislpd_B43BD197 not found
process_undo_entry(): entry name=mls_discde_B43BD197 not found
process_undo_entry(): entry name=mls_rmrhostsnetrc_B43BD197 not found
process_undo_entry(): entry name=mls_rmdotfrmpathroot_B43BD197 not found
process_undo_entry(): entry name=mls_removeguest_B43BD197 not found
```

You should not worry about any "*not found*" information. Usually, this means that during the implementation, the given parameter was in accordance with the target settings, so it was not changed. Therefore, its definition appears in the *appliedaixpert.xml* file, which is needed to be able to verify the compliance of the current system configuration with a given level. It does not appear in the *undo.xml* file, since no change has been made.

```
# aixpert -t   # The appliedaixpert.xml file is missing - No security level is implemented.
Error in file "/etc/security/aixpert/core/appliedaixpert.xml", line 3, column 28
Empty content not valid for content model: '(AIXPertEntry)+'
An error occurred during parsing
```

Implementation of your own set of security rules:

Generally, none of the available sets of rules fully match a particular company's security policies. Therefore, the most common rules are prepared based on existing ones. When setting the pre-defined security levels, there may be errors in the rules due to individual system customization. This represents a good reason to prepare your own set of rules, eliminating those that report errors. In the example below, we have prepared our own set of rules based on the *medium* level of security:

We unload the rules related to the *medium* level of security to the file:

```
# aixpert -l medium -n -o /etc/security/aixpert/custom/myMedium.xml
# ls -l /etc/security/aixpert/custom/myMedium.xml
-rw-r--r--    1 root     system        58226 Sep 15 05:54 /etc/security/aixpert/custom/myMedium.xml
```

There were several warnings in the first scenario that implemented the *medium* level:

```
do_action(): rule(mls_tcbupdate): warning.
do_action(): Warning: Prereq failed for prereqtcb
do_action(): rule(mls_rootrlogin): warning.
do_action(): Warning: Prereq failed for prereqrrl
do_action(): rule(mls_xhost): warning.
do_action(): Warning: Prereq failed for X11.Dt.ToolTalk
do_action(): rule(mls_ISSServerSensorFull): warning.
do_action(): Warning: Prereq failed for prereqRSSSFull
do_action(): rule(mls_ISSServerSensorLite): warning.
do_action(): Warning: Prereq failed for prereqRSSSLite
```

In order to eliminate the warnings that appear during implementation, we remove the rules that cause them from the created */etc/security/aixpert/custom/myMedium.xml* file. You can use any text editor for this purpose. Next, we implement the rules from the prepared file:

```
# aixpert -f /etc/security/aixpert/custom/myMedium.xml
Processedrules=78       Passedrules=78  PrereqFailedrules=0     Failedrules=0   Level=MLS
        Input file=/etc/security/aix
```

The rule file cleared in this way does not display any warnings. The command is "clean" and does not raise any doubts.

Trusted Execution (TE)

Trusted Execution is a feature that allows you to verify the integrity of the system. Maintaining integrity and detecting unauthorized changes are key to preventing threats such as Trojans or rootkits.

The trusted execution engine is installed by default with the operating system. It can work in two modes:

- **Offline mode** - Similar to the TCB (*Trusted Computing Base*) mechanism, which is popular in the UNIX world. It allows the periodic verification of integrity by the system administrator. Integrity verification involves comparing the current parameters of individual files in the system with the parameters defined as appropriate in the signature database.
- **Online mode** - A background operation mode that can automatically block certain potentially harmful activities.

Files related to TE

The key files related to *Trusted Execution* are:

- ***/etc/security/tsd/tsd.dat*** - *Trusted Signature Database (TSD)* - A database of file signatures that specify the "proper" file parameters. By default, the database contains information about the most important files in the system. It is possible to define additional files, whether related to the operating system or the application. A fragment of the file contents:

```
# grep -p -e "/usr/sbin/cfgmgr:" -e "/etc/security/passwd" /etc/security/tsd/tsd.dat
/usr/bin/mv:
        owner = bin
        group = bin
        mode = 555
        type = FILE
        hardlinks =
        symlinks =
        size = 22332
        cert_tag = 49424d4149583a31324331342d33314332303a324b3a41
        signature =
4c1bb73e696706b0e3f0df6a6717f0d5fe93eac77c438ab062cc885f177ed93e83533220ecf3ed29cffe33479057db7621b3
fd325e3d799d351e08cd8107040214ab6d9de779b7ac319ea43e331289f40ae962bdb5b49061402dcb93f9043fb3dceb66a6
691f469dec12219fee9943b22c2d3905cc060977714db84e9b723ff05580613f3047a6b2bb99136133ed1bfc05721f8b0f62
4ae1703d90fc24c69a7c9d0dda51162770326fd93013be4f2f7027b741089d290592de9fc18780fa61776d4b61ba4f3cbc2d
fa81724c4e77ba97a0a25b83b512fec1981ab14bfa045246349d2371700c82126993bbc774a3713b1a5078aac0eeeb77399d
fdd8230b6335
        hash_value = d3d99f476a39ae02b5d39f0406c3d013468bf0fdf10639cabe2da96fc281cd53
        minslabel =
        maxslabel =
```

```
        intlabel =
        accessauths =
        innateprivs =
        inheritprivs =
        authprivs =
        secflags =

/usr/sbin/cfgmgr:
        owner = root
        group = system
        mode = TCB,SUID,550
        type = FILE
        hardlinks =
        symlinks =
        size = 59128
        cert_tag = 49424d4149583a31324331342d33314332303a324b3a41
        signature =
3cbf23f2aae4416bec2d23e2447b885ac54e655a2d47a88fe423a579a75183ca0fbcabe16c9d276fe6a2ea6d69324baedbe2
7a42571b87f8a1f49e8526d7d7db35a2313228a0cd389530b94f5e56a08d2d902f432110968bdbf09fca909461dbc363a5f9
4ed977aec986781701fdedc606e13710a8f7ddcd53fd53d78f2447da6aec088dfeb1cb1d1f785faa9a6ac8dcbb16ff7d14ef
2c4825c5cc2be6a5c6cff6ddd007241b5ba680517eb9a3f8dabdfeebe30027564249eaf03097df4dee446925aa7b878ce831
68e6c7113335c15fd1e6814ddbde7c520dddcc2130c66c31d3a0ceec3a5d89fe9920b2053ed22b112584230bd46b8feef172
40cc28ccd2dd
        hash_value = d5d0adbbebea2c55ede61a35061d478e4847357eeaf04250eb3b60850429468b
        minslabel =
        maxslabel =
        intlabel =
        accessauths = aix.device.config
        innateprivs = PV_AU_ADD,PV_AU_PROC,PV_DAC_R,PV_DAC_W,PV_DAC_X,PV_DAC_O,PV_DEV_CONFIG
        inheritprivs =
PV_DAC_R,PV_DAC_W,PV_DAC_X,PV_DEV_CONFIG,PV_DEV_LOAD,PV_DEV_QUERY,PV_KER_VARS,PV_ROOT
        authprivs =
        secflags = FSF_EPS
        egid = 0
```

The above output shows information about two commonly used executable files: */usr/bin/mv* and */usr/sbin/cfgmgr*. You can see several types of parameters:

- ***owner***, ***group***, ***mode***, ***type***, ***hardlinks***, ***symlinks***, ***size*** - File parameters that describe the user, group, rights, etc.
- ***cert_tag***, ***signature***, ***hash_value*** - The certificate used to sign the file (the certificates are in the */etc/security/certificates* directory), its file signature, and the cryptographic hash.
- ***minslabel***, ***maxlabel***, ***intlabel***, ***accessauths*** - Parameters used after installing the system with the *Trusted AIX* option.
- ***accessauths***, ***innateprixs***, ***inheritprivs***, ***authprivs secflags*** - Specific permissions used in RBAC (*Role Based Access Control*).
- For some files, the ***VOLATILE*** value may appear in some parameters. Typically, this applies to files that often change their size, and it is related to the parameters *size*, *hash_value*, and *signature*.

- ***/etc/security/certificates/**** - A directory that contains the certificates used to sign individual files in the TSD database.

- */etc/security/tsd/tepolicies.dat* - A file with online policies.

Information about *Trusted Execution* is usually stored in local files within the file system. However, you can transfer them to the LDAP and then use that location as a data source. A suitable scheme is prepared for this purpose, which must be loaded into the LDAP (*/etc/security/ldap/sec.ldif*). There is also a command for exporting data from local structures to a file (*/usr/sbin/tetoldif*), which can then be imported to the created schema in the LDAP.

Integrity checking - Offline mode

You can perform offline integrity checks at the entire system level (using the word ALL instead of the file name in the command). The current parameters of all the files listed in the Trusted Signature Database along with their parameters stored in this database will be compared. You can also check specific files that appear suspicious. In both modes, you can either correct the file properties automatically (*trustchk -y*) or wait for a question about correcting certain attributes (*trustchk -t*).

Example of the verification process:

```
# ls -l /usr/bin/rcp   # We check the file parameters.
-r-sr-xr-x    1 root     system        51283 Mar 04 2016  /usr/bin/rcp

# chmod o+w /usr/bin/rcp   # We change the access parameter for "other users."

# ls -l /usr/bin/rcp   # The file with the changed permission.
-r-sr-xrwx    1 root     system        51283 Mar 04 2016  /usr/bin/rcp

# trustchk -t /usr/bin/rcp   # Comparison with the TSD database.
trustchk: Verification of attributes failed: mode
Change the file mode for /usr/bin/rcp? [(y)es,(n)o,(i)gnore all errors]: y
trustchk: Verification of stanza failed:

# ls -l /usr/bin/rcp   # Trustchk has corrected the file to the same value as in the TSD.
-r-sr-xr-x    1 root     system        51283 Mar 04 2016  /usr/bin/rcp
```

In the above example, the file */usr/bin/rcp* permission has been changed. Next, the *trustchk -t* command was run, which compared the current parameters of the file with its parameters as recorded in the TSD. After the detection of discrepancies, the *trustchk* asked whether to correct the file mode.

Depending on the differences between the current file parameters and those in the TSD, *trustcheck* takes different actions. In the case of minor problems, such as the incorrect owner, group, or file permissions, the parameters are corrected, as shown in the example above. In case of major problems, such as the incorrect file size or the incorrect *hash_value* or *signature*, access to the file is blocked. Blocking is the correct approach because there is a high probability that a file with a different size than the one registered in the TSD is a Trojan, so it is likely very dangerous for the system.

Example of verification with the automatic correction of parameters:

```
# ls -l /usr/bin/rcp   # We check the file parameters.
-r-sr-xr-x    1 root     system        51283 Mar 04 2016  /usr/bin/rcp

# cp /usr/bin/telnet /usr/bin/rcp   # We replace the rcp file with the telnet file.
```

```
# ls -l /usr/bin/rcp   # The file size changes.
-r-sr-xr-x    1 root     system       281277 Sep 08 09:41 /usr/bin/rcp

# trustchk -y /usr/bin/rcp   # Comparison with the TSD database (-y - Correction without asking).
trustchk: Verification of attributes failed: size
trustchk: Verification of attributes failed: hash
trustchk: Verification of attributes failed: signature
trustchk: Verification of stanza failed:

# ls -l /usr/bin/rcp   # Access to the file is blocked.
---------T    1 root     system       281277 Sep 08 09:41 /usr/bin/rcp
```

In the above example, the file */usr/bin/rcp* was changed to */usr/bin/telnet*, so the size of the file changed. This is a major change, which suggests that the */usr/bin/rcp* file has been replaced with malware. All the rights have been removed from it, so the file cannot be used in the system.

The most frequently used method is the periodical checking of all the files described in the TSD database:

```
# trustchk -t ALL       # Check all files from the TSD, ask wether to change  the files.
# trustchk -y ALL       # Check all files from the TSD with automatic change.
```

Integrity checking - Online mode

Trusted execution can be used to perform specific online checks (i.e., before the software is started). This form of verification may concern such aspects as the compliance of executable files, shared libraries, or kernel extensions with the TSD. It may also affect the location of given files and prevent them from being run from paths other than trusted ones.

This mechanism is managed by changing policies. The policies that you can modify are detailed in Table 14-3.

Table 14-3: Settings for changing policies.

Policy	Description
TE=ON/OFF	Enabling Trusted Execution policy (*trustchk -p te = on*).
CHKEXEC=NO/OFF	Checking the integrity (compliance with the information in the TSD) of each executable program before it is run (*trustchk -p chkexec = on*).
CHKSHLIB=ON/OFF	Checking the integrity (compliance with the description in the TSD) of each shared library before it is loaded.
CHKSCRIPT=ON/OFF	Checking the integrity (compliance with the description in the TSD) of each script before it is run.
CHKKERNEXT=ON/OFF	Checking the integrity (compliance with the description in the TSD) of each kernel extension before it is loaded.
The following policies determine how the system will behave if it detects the incompliance described above.	
STOP_ON_CHKFAIL=ON/OFF	If the file matches its definition in the TSD - Operate normally. If the file does not match its definition in the TSD - Stop

	further actions (e.g., do not run the program). If the checked file is not described in the TSD - Operate normally.
STOP_UNTRUSTD=ON/OFF	If the file matches its definition in the TSD - Operate normally. If the file does not match its definition in the TSD - Stop further actions (e.g., do not run the program). If the checked file is not in the TSD - Stop further actions (e.g., do not start the program).
The following policies apply to paths.	
TEP=Paths(like_$PATH)	Trusted Execution Path (TEP). If the policy is set, only programs on the listed paths can be run.
TLP=Paths(like_$PATH)	Trusted Library Path (TLP). If the policy is set, only libraries on the listed paths can be loaded.
Policies related to locking.	
LOCK_KERN_POLICIES=ON/OFF	Enabling this policy will freeze the current policy settings. From now on, you will not be able to change them until the system is restarted.
TSD_FILES_LOCK=ON/OFF	Enabling this policy prevents the files listed in the TSD from being opened in write mode.
TSD_LOCK=OFF	Enabling this policy prevents the TSD database from being opened in write mode. It is not possible to change the database.

Examples - Setting policies:

```
# trustchk -p CHKEXEC=ON STOP_UNTRUSTD=ON

# trustchk -p TSD_LOCK=ON

# trustchk -p TE=on   # Enabling policies.

# trustchk -p TEP=/usr/bin:/usr/sbin:/etc:/bin:/sbin:/sbin/helpers/jfs2:/usr/lib/instl
:/usr/ccs/bin:/usr/lib:/usr/lib/security:/etc/secure    # Setting trusted paths.

# trustchk -p tep=on   # Enabling the mechanism of trusted paths.

# trustchk -p   # Displaying the current settings.
TE=ON
CHKEXEC=ON
CHKSHLIB=OFF
CHKSCRIPT=OFF
CHKKERNEXT=OFF
STOP_UNTRUSTD=ON
STOP_ON_CHKFAIL=OFF
LOCK_KERN_POLICIES=OFF
TSD_FILES_LOCK=OFF
TSD_LOCK=ON
TEP=ON
TLP=OFF

# grep TEPPATH /etc/security/tsd/tepolicies.dat   # Checking the active TEP paths.
        TEPPATH = /usr/bin:/usr/sbin:/etc:/bin:/sbin:/sbin/helpers/jfs2:/usr/lib/instl
:/usr/ccs/bin:/usr/lib:/usr/lib/security:/etc/security
```

An example of the operation of the policies enabled above (*CHKEXEC*, *STOP_UNTRUSTED*):

```
# id   # The id program works correctly.
uid=0(root) gid=0(system) groups=2(bin),3(sys),7(security),8(cron),10(audit),11(lp)

# cp /usr/bin/id /usr/bin/id2   # We create a copy of the id.

# id2   # The copy cannot run due to policies (CHKEXEC = ON, STOP_UNTRUSTD = ON).
ksh: id2: 0403-006 Execute permission denied.
```

In the example above, we cannot run the program */usr/bin/id2*. When we try to start it, the program is checked in the TSD (*CHKEXEC* policy). The program is not listed in the TSD, so the launch is blocked due to the *STOP_UNTRUSTD* policy. To run the program, add the relevant entries concerning it to the TSD.

An example of the operation of the *TSD_LOCK* policy enabled above:

```
# ls -l /etc/security/tsd/tsd.dat   # The root user has write access to the TSD file.
-rw-r-----    1 root     security    2718277 Sep 08 14:11 /etc/security/tsd/tsd.dat

# echo "writeover tsd" > /etc/security/tsd/tsd.dat
System call error number -1.   # Writing fails (TSD_LOCK).
ksh: /etc/security/tsd/tsd.dat: 0403-005 Cannot create the specified file.
```

Example of using the *Trusted Execution Path* (TEP):

```
# trustchk -p
TE=ON
CHKEXEC=OFF
CHKSHLIB=OFF
CHKSCRIPT=OFF
CHKKERNEXT=OFF
STOP_UNTRUSTD=OFF
STOP_ON_CHKFAIL=OFF
LOCK_KERN_POLICIES=OFF
TSD_FILES_LOCK=OFF
TSD_LOCK=ON
TEP=ON
TLP=OFF

# grep TEPPATH /etc/security/tsd/tepolicies.dat   # Checking the active TEP paths.
        TEPPATH = /usr/bin:/usr/sbin:/etc:/bin:/sbin:/sbin/helpers/jfs2:/usr/lib/instl
:/usr/ccs/bin:/usr/lib:/usr/lib/security:/etc/security

# cp /usr/bin/id /   # Copying an id file to an unauthorized path: "/".

# ls -l /id   # Verification of the permissions of a new file.
-r-xr-xr-x    1 root     system        10530 Sep 09 02:40 /id

# /id   # The file does not start; no "/" path in the TEPPATH.
ksh: /id: 0403-006 Execute permission denied.
```

Modification of the TSD

The TSD can be freely changed by adding, deleting, and modifying its contents. If you plan to add entries to the TSD, you should create a certificate that will be used to add new files to the database.

Example of creating a certificate:

```
# openssl genrsa -out /home/cert/TSDprivkey.pem 2048

# openssl req -new -x509 -key /home/cert/TSDprivkey.pem -outform DER -out
/home/cert/TSDcertificate.der -days 3650
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:

# openssl pkcs8 -inform PEM -in /home/cert/TSDprivkey.pem -topk8 -nocrypt -outform DER -out
/home/cert/TSDprivkey.der

# ls -l /home/cert/TSDprivkey.der
-rw-r--r--    1 root     system         1219 Sep 09 03:49 /home/cert/TSDprivkey.der
```

Example of adding a */id* file to the TSD:

```
# trustchk -s /home/cert/TSDprivkey.der -v /home/cert/TSDcertificate.der  -a /id

# trustchk -q /id
/id:
        type = FILE
        owner = root
        group = system
        mode = 555
        size = 10530
        hash_value = ada2a07bd3a52b633d8623c6b669723811495cf044493394975b55abbb44d4b8
        cert_tag = 00b10dc3fa3c549ebf
        signature =
56eb4a48b3b1024d8bce4b571ce0d6a8f0ac50007e5a123e69713cf1b6184d59c8289c44d88d1de052423beb1858805468d4
86b2e4437dd893e84dbe64a0df9dbbec0713e7ae6cce4c12f2e05c9de2e227eeef6d43d34dfd4a8c25127a57b8a325e3da86
4936c69f1fbd881677131a7fd0be920ffd6d67c23fc9966312c8fa7ee82bca9b122b9cca11af6d7daf5797b4a5cd2ff589d2
fc9529c290e90c4b7e8b93723c773a67da43c5f48c03613cd6847145903cb5655eec16ac968a45eb4b5db4add25526d46fb9
fa3e41842d261aef1a24b117e47ad3a9d15f27723d49405fdb4ffed731922b85bf1d59eacc2f6fe87413ca1b75d3aa04dbef
8b4eab577d8a

# ls -l /etc/security/certificates
total 40
-rw-r-----    1 root     system          865 Sep 09 04:06 00b10dc3fa3c549ebf
-rw-r-----    1 root     security        776 Sep 18 2014  certificate_61
-rw-r-----    1 root     security        776 Sep 18 2014  certificate_610
-rw-r-----    1 root     security        776 Sep 18 2014  certificate_71
-rw-r-----    1 root     security        983 Feb 16 2015  certificate_72
drwxr-x---    2 root     security        256 Jun 23 10:06 tnc
```

The file is added to the *Trusted Signature Database*. The parameters *type*, *owner*, *group*, *mode*, and *size* are loaded from the current file parameters. The *hash_value*, *cert_tag*, and *signature* are rewritten/generated on the basis of the attached certificate. The certificate used is copied to the */etc/security/certificates* directory, and its name is *cert_tag*.

You can add entries to the TSD without using certificates, although you can consider such entries to be less trusted.

Other uses of *trustchk*:

```
trustchk -q /usr/bin/id   # Displays data about the file /usr/bin/id from TSD.

trustchk -d /usr/bin/id   # Removes information about the file /usr/bin/id from the TSD.

trustchk -a -f  /tmp/file_description   # Adding the file described in /tmp/file_description to the
TSD (description in the format that the trustchk -q command displays).
```

Encrypted File System (EFS)

On the AIX system, it is possible to encrypt the file system. However, if you take a closer look at the implemented mechanism, you will see it is not actually the whole file system that is encrypted, but rather individual files. The ability to encrypt files appeared with AIX 6.1, and it only applies to the JFS2 file system. The following elements are required for its operation:

1. RBAC (*Role Based Access Control*) enabled - In AIX 7.2, it is enabled by default.
2. Installed *clic.rte* fileset - In AIX 7.2, it is installed during the standard system installation.
3. Initialization of the encryption environment (*efsenable -a* command).
4. Set the appropriate attribute on the file systems to be encrypted (*efs=yes*).

The encryption mechanism is based on two elements, namely the keystore and encryption keys.

Keystore

Each user or group of users has their own keystore. It is usually located on the */var/efs/username_or_groupname* path. It can also be stored on the LDAP server if you load the appropriate schema, which is located on the */etc/security/ldap/sec.ldif* path. It is also necessary to export the EFS data (command */usr/sbin/efskstoldif*) from the local file system to the LDAP.

The keystore stores the key pair (private and public) of the user or group. The private key is encrypted with the user/group password. The key must be loaded into the memory so that the user can use it. This can be done in the following ways:

- Automatically during the login process - If the user's password in the system is the same as the password to his keystore.
- Manually after running the *efskeymgr -o ksh* command - If the user's password in the system is different than the password to his keystore.

Encryption keys

Each file is encrypted with a separate, randomly generated key. This key is stored in the file's metadata, in the place where the extended file attributes are stored. It is kept there in the form encrypted with the user's public key. The file encryption key is kept in as many copies as the number of users/groups that have access to it. Each copy is encrypted with the public key of another user/group. This is shown in the example below:

```
# efsmgr -l /efsfs/file
EFS File information:
 Algorithm: AES_128_CBC
 Block Size: 4096
List of keys that can open the file:
 Key #1:
  Algorithm       : RSA_2048
  Who             : uid 204
  Key fingerprint : c6477592:274cb49f:7eb5a36d:1888325e:5b43b63f
 Key #2:
  Algorithm       : RSA_2048
  Who             : uid 205
```

```
  Key fingerprint : 347059df:a8a1b90e:29150763:2ceda98e:86cf4633
 Key #3:
  Algorithm       : RSA_2048
  Who             : gid 1
  Key fingerprint : 2c2d8347:01006d89:644b8c22:e3c8becb:e4c76165
```

As you can see above, two users with *uid 204* and *205* as well as a group of users with *gid 1* have access to the */efsfs/file*. Thus, three copies of the file encryption key are stored in the extended attributes of the file. Each copy is encrypted with the public key of another user/group. There may be a situation wherein a user with access to a file from the encryption point of view will not have access to it in reality. To access an encrypted file, the user must meet two requirements:

1. Must comply with the rules of standard file access (*rwx* privileges) or have access granted via the ACL (*Access Control List*).

2. The extended file attributes must contain a file key encrypted with its (or its group's) public key.

The user's access to the encrypted file is achieved in the following way:

1. After logging in, the user reads the keys from his keystore (if the keystore password is the same as the system password, this is done automatically):

```
# efskeymgr -o ksh # Starting a new shell instance with loaded keys.
```

2. The user reads the file:
   a. The file encryption key is decrypted using the user's private key.
   b. The data read from the file by the user are decrypted using the file encryption key.

It is worth noting that the file is never saved to the disk in an unencrypted form. It is only processed in this form in the RAM.

Starting work with encryption

Before initiating the system to work with encrypted files, check that RBAC is switched on and that the appropriate filesets (*clic.rte*) are installed. On most systems, these conditions are met by default.

```
# lsattr -El sys0 | grep RBAC
enhanced_RBAC   true                            Enhanced RBAC Mode     True

# lslpp -L | grep clic
  clic.rte.kernext          4.10.0.1    C     F    CryptoLite for C Kernel
  clic.rte.lib              4.10.0.1    C     F    CryptoLite for C Library
```

The next step is to initialize the system to work with encrypted files (*efsenable* command):

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

This command asks you to enter the initial keystore password. This is the *efs* administrator password. Usually, the *root* user is an *efs* administrator, but it can also be another user. After the command is run,

the following changes take place in the system:

1. A directory structure to store user keystores is created:

```
# ls -l /var/efs
total 8
drwx------    2 root     system          256 Sep 09 07:22 efs_admin
-rw-r--r--    1 root     system            0 Sep 09 07:22 efsenabled
drwx------    5 root     system          256 Sep 09 07:37 groups
drwx------   11 root     system         4096 Sep 09 07:42 users
```

efs_admin - EFS administrator keystore encrypted with the password provided when running the *efsenable* command.
efsenable - A tag that indicates that encryption is enabled.
groups - Directory for user group's keystores.
users - Directory for the user's keystores.

2. Keystores are created for the *root* user and for the *security* group, with a pair of keys (private and public). From the moment the encryption is enabled, any user who logs in to the system will automatically generate a keystore for himself.

```
# ls -l /var/efs/users
total 0
-rw-------    1 root     system            0 Sep 09 07:22 .lock
drwx------    2 root     system          256 Sep 09 07:22 root

# ls -l /var/efs/groups
total 0
-rw-------    1 root     system            0 Sep 09 07:22 .lock
drwx------    2 root     system          256 Sep 09 07:22 security
```

3. Information about the encryption is added to the default section of the */etc/security/users* file.

```
# grep efs /etc/security/user
        efs_keystore_access = file   # Location of the user's keystore.
        efs_adminks_access = file   # Location of EFS administrator's keystore.
        efs_initialks_mode = admin   # Operating mode (admin, guard).
        efs_allowksmodechangebyuser = yes   # Possibility of changing the above mode.
        efs_keystore_algo = RSA_2048   # The algorithm used for keystore. Choose from: RSA_1024,
RSA_2048, and RSA_4096.
efs_file_algo = AES_128_CBC   # The algorithm for file encryption. Choose from:
                              # AES_128_CBC,AES_192_CBC, AES_256_CBC,
                              # AES_128_ECB, AES_192_ECB, and AES_256_ECB.
```

The keystore can be stored in local files (the method described in this chapter), although it can also be stored in the LDAP. The encryption algorithms can be selected as needed. The operating mode (*admin*, *guard*) will be described later in the chapter.

4. Information about the encryption is added in the /etc/security/group file, for example:

```
security:
        admin = true
        efs_keystore_access = file
        efs_initialks_mode = admin
        efs_keystore_algo = RSA_2048
```

5. Information about the encryption within the system is inserted into the ODM.

```
# odmget Config_Rules | grep efsenable
        rule = "/usr/sbin/efsenable -p"
```

After completing the above steps, you can start creating the new file system in which you will operate on encrypted files. You can also allow an existing file system to work with encrypted files. Any file system can be encrypted, except for those necessary to boot the operating system (i.e., */*, */usr*, */var*, and */opt*).

```
# crfs -v jfs2 -g rootvg -m /efsfs -a size=100M -a efs=yes   # Creating an encrypted file system.

# chfs -a efs=yes /home   # Enabling the encryption of the /home file system does not require a
restart.
```

The encrypted file system stores the keys in the extended file attributes (version 2 of the extended attributes). If the file system used extended attributes in version 1, they will automatically be converted to version 2 together with the current contents.

It is worth remembering that encryption is performed at the file level, not the entire file system level. Thus, a file system can contain both encrypted and unencrypted files.

Key commands

The standard commands used on a daily basis in the operating system are adapted to work with encrypted files. Their use is usually transparent to the user. However, there are several specific commands related only to encryption:

```
# efsenable    # Enabling encryption options in the system, as described above.
# efsmgr  # File management.
# efskeymgr    # Encryption keys management.
```

Examples:

```
$ efsmgr -l encfile1   # Displays file information.
EFS File information:
 Algorithm: AES_128_CBC   # The algorithm used to encrypt the file.
 Block Size: 4096
List of keys that can open the file:
 Key #1:
  Algorithm       : RSA_2048   # The algorithm used to encrypt the file encryption key.
  Who             : uid 0
  Key fingerprint : 41899dca:38cddaeb:b9853eaf:6230cae9:3de18eae
 Key #2:
  Algorithm       : RSA_2048
  Who             : uid 207
  Key fingerprint : e071e083:d51772ff:a0d93d76:6a24c527:ca37e5e7
```

As you can see in the above output, two users with *uid: 0* and *207* have access to the file. There are hence two copies of the file key in the file's metadata. One copy is encrypted with the key of the user with *uid 0*, while the second copy is encrypted with the key of the user with *uid 207*.

```
# efsmgr -e file_name   # Encrypts the file.

# efsmgr -a file_name -u user_name   # Gives file permissions to the user - Adds the version of the
key that is encrypted with the user's public key.

# efsmgr -a file_name -g group_name   # Gives file permissions to the group - Adds the version of
the key that is encrypted with the group's public key.

# efsmgr -E directory   # Sets the encryption inheritance on the directory.

# efsmgr -D directory   # Removes the inheritance of encryption from the directory.

# efsmgr -L directory   # Checks the inheritance of encryption on the directory.

$ efskeymgr -V   # Displays the list of loaded user keys.
List of keys loaded in the current process:
 Key #0:
                          Kind ..................... User key
                          Id   (uid / gid) ......... 207
                          Type ..................... Private key
                          Algorithm ................ RSA_2048
                          Validity ................. Key is valid
                          Fingerprint .............. e071e083:d51772ff:a0d93d76:6a24c527:ca37e5e7

$ efskeymgr -R RSA_4096   # Creating a new user key.

$ efskeymgr -o ksh   # Loading user keys.

$ efskeymgr -V
List of keys loaded in the current process:
 Key #0:
                          Kind ..................... User key
                          Id   (uid / gid) ......... 207
                          Type ..................... Private key
                          Algorithm ................ RSA_2048
                          Validity ................. Key is deprecated
                          Fingerprint .............. e071e083:d51772ff:a0d93d76:6a24c527:ca37e5e7
 Key #1:
                          Kind ..................... User key
                          Id   (uid / gid) ......... 207
                          Type ..................... Private key
                          Algorithm ................ RSA_4096
                          Validity ................. Key is valid
                          Fingerprint .............. d5b62fd2:d762c6c1:0ccd3e73:7b5e689f:9dbcc45a
```

As you can see in the above output, a new user key was created using the RSA_4096 algorithm. From now on, all new files to which the user will have access will contain a new version of the key, while all files that existed prior to the change will still have an old version of the user key. To maintain the ability to work with new and old files, both keys are in the user's keystore. The old key has the *deprecated* status, which means that it is not used for new files.

Standard, everyday commands in the context of the encryption mechanism behave as follows:

- ***cp***, ***mv*** - Have additional parameters that must be used in specific situations:
  - ***-d*** - Decrypt the file. For use, for example, when copying an encrypted file to a file system that does not support encryption.
  - ***-e*** - Encrypt the file. For use, for example, when copying a file from a file system that does not support encryption to the EFS.
- ***su*** - Unlike the standard login process, it does not load the user keystore. The keystore is loaded manually using a password. The standard login process loads the keystore, so long as it

is protected with the same password as the user's login.

- ***chown*** - To make it work on an encrypted file, a key of the future owner of the file must be located in the file metadata. After running the *chown* command, the former owner's key is deleted (even if the file has access rights for all users - *others* section).
- ***chgrp*** - Changing the file group results in adding a new group key to the extended attributes and removing the old group key (if the groups have a keystore created).
- ***chmod*** - If classic access rights are revoked, the corresponding keys are deleted, while if the permissions are granted, then the keys are added.
- ***rmuser*** - The command does not delete the user keystore. If you delete the keystore, all the encrypted user files would be unrecoverable.

Scenarios for EFS activities

The easiest way to become familiar with the mechanisms is by analyzing examples of their use. Therefore, there are several scenarios presented below that show the mechanisms of file encryption. In the following scenarios, we assume that the EFS is enabled (*efsenable -a*).

Scenario 1: Encrypting and assigning access to other users

We create a test file:

```
# echo "encryption test" > encfile1
```

The file is created in the file system with the encryption option. However, it is created as an unencrypted file, since the file system does not have the encryption inheritance option set:

```
# efsmgr -l /efsfs/encfile1 # The created file is unencrypted.
Error getting EFS attributes: Can not find the requested security attribute.
```

We encrypt the file:

```
# efsmgr -e /efsfs/encfile1  # File encryption.
```

Only one user has access to the encrypted file. There is hence only one copy of the file encryption key in the file's metadata, which is encrypted with the key of *root* user (*uid 0*):

```
# efsmgr -l /efsfs/encfile1    # List the encryption parameters of the file.
EFS File information:
 Algorithm: AES_128_CBC
 Block Size: 4096
List of keys that can open the file:
 Key #1:       # One user (one key) has access to the file.
  Algorithm       : RSA_2048
  Who             : uid 0
  Key fingerprint : 41899dca:38cddaeb:b9853eaf:6230cae9:3de18eae
```

We give permissions for the *encfile1* file to the *encuser1* user. This fails because the user does not have a keystore. His keystore will be created after the first login:

```
# efsmgr -a /efsfs/encfile1 -u encuser1   # Granting file permissions for encuser1.
Unable to get public key from user "encuser1" (skipped): Keystore does not exist
```

```
encuser1: A file or directory in the path name does not exist.
```

After logging in as the user *encuser1* using another session, we try again, this time with success. In the file metadata, an additional copy of the encryption key appears, which is encrypted with the *encuser1* user key:

```
# efsmgr -a /efsfs/encfile1 -u encuser1   # Granting file permissions for encuser1.

# efsmgr -l /efsfs/encfile1
EFS File information:
 Algorithm: AES_128_CBC
 Block Size: 4096
List of keys that can open the file:
 Key #1:
  Algorithm       : RSA_2048
  Who             : uid 0
  Key fingerprint : 41899dca:38cddaeb:b9853eaf:6230cae9:3de18eae
 Key #2:       # The encuser1 user key is added to the file.
  Algorithm       : RSA_2048
  Who             : uid 207
  Key fingerprint : e071e083:d51772ff:a0d93d76:6a24c527:ca37e5e7
```

We switch to *encuser1* and read the file. The attempt is only successful after the keys from the user keystore have been loaded into the memory:

```
# ls -l /efsfs/encfile1
-rw-r--r--    1 root     system           16 Sep 10 03:01 /efsfs/encfile1

# su - encuser1   # We switch to the encuser1 user.

$ efskeymgr -V   # The user keys are not loaded, so he cannot work with encrypted files.
There is no key loaded in the current process.

$ efskeymgr -o ksh   # Loading user keys.
encuser1's EFS password:   # Providing a password to the user's keystore.

$ cat /efsfs/encfile1
encryption test
```

Scenario 2: Standard permissions versus encryption keys

The file permissions change. Revoking read permission from the *others* section revokes read access to the *encuser1* user:

```
# chmod 640 /efsfs/encfile1   # Changing the access permissions to the file.

# ls -l /efsfs/encfile1   # Encuser1 no longer has access to the file.
-rw-r-----    1 root     system           16 Sep 10 03:01 /efsfs/encfile1
```

We switch to the user *encuser1* and try to read the file (this attempt failed).

```
# su - encuser1   # Switch to user encuser1.

$ efskeymgr -o ksh   # Loading encuser1 user keys.
encuser1's EFS password:

$ cat /efsfs/encfile1   # encuser1 is unable to read the file.
cat: 0652-050 Cannot open /efsfs/encfile1.
```

```
# efsmgr -l /efsfs/encfile1
EFS File information:
 Algorithm: AES_128_CBC
 Block Size: 4096
List of keys that can open the file:
 Key #1:
  Algorithm       : RSA_2048
  Who             : uid 0
  Key fingerprint : 41899dca:38cddaeb:b9853eaf:6230cae9:3de18eae
 Key #2:
  Algorithm       : RSA_2048   # The encuser1 user key exists in the file's metadata.
  Who             : uid 207
  Key fingerprint : e071e083:d51772ff:a0d93d76:6a24c527:ca37e5e7
```

The above scenario shows that to operate on encrypted files, the user must have access to the file in the form of both *rwx* permissions as well as the appropriate version of the encryption key stored in the extended attributes of the file.

Scenario 3: User groups

We log in as *encuser1*, create a new file, encrypt it, and verify the keys that appeared in the metadata:

```
$ echo "encrypted file2" > /efsfs/encfile2

$ efsmgr -e /efsfs/encfile2   # File encryption.

$ ls -l /efsfs/encfile2
-rw-r--r--    1 encuser1 staff            16 Sep 10 04:53 /efsfs/encfile2

$ efsmgr -l /efsfs/encfile2   # Verification of the file encryption.
EFS File information:
 Algorithm: AES_128_CBC
 Block Size: 4096
List of keys that can open the file:
 Key #1:
  Algorithm       : RSA_4096
  Who             : uid 207
  Key fingerprint : d5b62fd2:d762c6c1:0ccd3e73:7b5e689f:9dbcc45a
```

As you can see, even though members of the *staff* group have classic access to the file (*ls -l*), they have no real access. The metadata of the file */efsfs/encfile2* does not include the key for this group. The reason for this is the lack of a keystore for this user group. Keystores for groups should be created manually.

We create a keystore for the *staff* group and add access (encryption key) on the part of the *staff* group to the file metadata:

```
# su - root
# efskeymgr -C Staff   # Create a keystore for the staff group as the root user.
# su – encuser1

$ efsmgr -a /efsfs/encfile2 -g staff   # Add a staff group user key to the file.

$ efsmgr -l /efsfs/encfile2
EFS File information:
 Algorithm: AES_128_CBC
 Block Size: 4096
List of keys that can open the file:
 Key #1:
```

```
  Algorithm       : RSA_4096
  Who             : uid 207
  Key fingerprint : d5b62fd2:d762c6c1:0ccd3e73:7b5e689f:9dbcc45a
 Key #2:       # Staff group user key added to the file.
  Algorithm       : RSA_2048
  Who             : gid 1
  Key fingerprint : 48bb7897:cef9bfeb:4a517cde:4e376445:830d001d
```

As you can see above, the file key of the *staff* group, which is encrypted with the public key, has been added to the extended attributes of the file.

As at the beginning of the scenario, we create a new file and then manually encrypt it:

```
$ echo "encrypted file3" > /efsfs/encfile3   # We create a new file.

$ efsmgr -e /efsfs/encfile3

$ efsmgr -l /efsfs/encfile3
EFS File information:
 Algorithm: AES_128_CBC
 Block Size: 4096
List of keys that can open the file:
 Key #1:
  Algorithm       : RSA_4096
  Who             : uid 207
  Key fingerprint : d5b62fd2:d762c6c1:0ccd3e73:7b5e689f:9dbcc45a
 Key #2:
  Algorithm       : RSA_2048
  Who             : gid 1
  Key fingerprint : 48bb7897:cef9bfeb:4a517cde:4e376445:830d001d
```

The above section of the example shows that after encrypting the file, the user and the *staff* group immediately gain access to it. This is because the keystore for the *staff* group existed at the time the file was created.

The above example showed how to work with user groups. It is important to note that encryption operations do not automatically grant access to the file to all users, even if they have access through the *rwx* permissions in the "*other*" section. If it is necessary for other users to obtain such access, you must explicitly assign it to them using the following command:

```
# efsmgr -a path_to_file -u user_name
```

Scenario 4: Inheritance

In all the above scenarios, the files were encrypted individually. If you need the encryption to take place automatically, you can set the inheritance of encryption at the directory level or at the file system level. The following example shows how to do this.

We prepare a keystore for the system group:

```
# efskeymgr -C system
```

We create a directory and check its inheritance setting:

```
# mkdir /efsfs/d1
```

```
# efsmgr -L /efsfs/d1   # The directory does not have encryption inheritance set.
Error getting EFS attributes: Cannot find the requested security attribute.
```

We set the encryption inheritance option. All new files in the directory will be created in an encrypted form:

```
# efsmgr -E /efsfs/d1   # Setting the encryption inheritance.

# efsmgr -L /efsfs/d1   # Verification.
EFS inheritance is set with algorithm: AES_128_CBC
```

We create a subdirectory that automatically inherits the encryption options:

```
# mkdir /efsfs/d1/d1.1   # Create a subdirectory.

# efsmgr -L /efsfs/d1/d1.1        # The created sub-folder inherited the encryption.
EFS inheritance is set with algorithm: AES_128_CBC
```

We create a file in the subdirectory:

```
# echo "encrypted file1" >/efsfs/d1/encfile1   # Create a new file.

# ls -l /efsfs/d1/encfile1
-rw-r--r--    1 root     system           16 Sep 10 05:58 /efsfs/d1/encfile1

# efsmgr -l /efsfs/d1/encfile1   # The file is encrypted straightaway.
EFS File information:
 Algorithm: AES_128_CBC
 Block Size: 4096
List of keys that can open the file:
 Key #1:
  Algorithm       : RSA_2048
  Who             : uid 0
  Key fingerprint : 41899dca:38cddaeb:b9853eaf:6230cae9:3de18eae
 Key #2:
  Algorithm       : RSA_2048
  Who             : gid 0
  Key fingerprint : 76a98ae4:460533de:bc530bba:acf9cb54:1912de10
```

The above file, since it was created in the directory with the attribute of encryption inheritance, was created in an encrypted form. There are two keys within its metadata - the *root* user key (the creator of the file) and the *system* group key (this group has a keystore created).

Setting the encryption attribute on the directory does not affect the files that previously existed there. If the files were unencrypted, they will remain unencrypted, while all new files will be encrypted.

Scenario 5: File operations

We create two directories, one with the option of encryption inheritance and the other without:

```
# mkdir /efsfs/noenc

# mkdir /efsfs/enc

# efsmgr -E /efsfs/enc   # We set up encryption inheritance.
```

We create two files, one in each directory:

```
# echo "file 1" >/efsfs/enc/file1

# echo "file 2" >/efsfs/noenc/file2

# ls -U /efsfs/enc/file1
-rw-r--r--e    1 root     system            7 Sep 10 06:11 /efsfs/enc/file1

# ls -U /efsfs/noenc/file2
-rw-r--r---    1 root     system            7 Sep 10 06:12 /efsfs/noenc/file2
```

The file created in the directory with encryption inheritance was encrypted (the attribute *e* in a result of the *ls -U* command). The file created in the directory without encryption inheritance was created in an unencrypted form.

We copy the unencrypted file to the directory with encryption inheritance:

```
# cp /efsfs/noenc/file2  /efsfs/enc/file2
cp: /efsfs/enc/file2: The file access permissions do not allow the specified action.

# cp -e /efsfs/noenc/file2  /efsfs/enc/file2

#  ls -U /efsfs/enc/file2
-rw-r--r--e    1 root     system            7 Sep 10 06:14 /efsfs/enc/file2
```

The attempt to copy the unencrypted file to the directory with the option to inherit encryption (using the standard command) failed. For the file to be encrypted, use the *-e* flag.

We copy the encrypted file:

```
# cp  /efsfs/enc/file1  /efsfs/noenc/file1

# ls -U /efsfs/enc/file1
-rw-r--r--e    1 root     system            7 Sep 10 06:11 /efsfs/enc/file1
```

The attempt to copy the encrypted file to the directory without the encryption inheritance option (using the standard command) succeeded. There is no need to use the *-e* flag, since the file is encrypted.

We are trying to copy the encrypted file to a file system that does not support encryption:

```
# cp  /efsfs/enc/file1  /tmp/file1
cp: 0990-006 Target filesystem is not efs-enabled

# cp -d  /efsfs/enc/file1  /tmp/file1
```

The use of a standard command to copy an encrypted file to a file system that does not support encryption causes an error. It is necessary to use the *-d* flag to decrypt the file when copying it.

EFS operation modes

In an encrypted file system, the root user is not able to easily gain access to the encrypted files of another user. After logging in from the *root* account to another user (*su* command), the user's keys do not load. In order to load them, the user must know the password for his keystore.

There are two EFS operation modes intended to provide a balance between a flexible approach and root user isolation. The appropriate mode can be selected during the EFS initialization:

```
# efsenable -a –m [admin | guard]
```

- ***admin*** - This is the default operation mode. If, during the EFS initialization, the mode is not explicitly selected, the admin mode is selected automatically. In this mode, the *root* user (EFS administrator) does not have direct access to the user keystore, although he can change the keystore password of the user. In this way, he can access encrypted user files. The option is available through special administrative keys stored in the keystore of the EFS administrator (*efs_admin*). Access to this keystore has a *root* user and a *security* group, or users with RBAC authorization *aix.security.efs*. Most commonly, encrypted file systems work in this mode.

  Resetting the user's keystore password (*admin* mode):

```
# efskeymgr -k user/encuser1 -n
Enter new password for encuser1's keystore:
Enter the same password again:
```

- ***Guard*** - A more restrictive mode of operation. In this mode, only the owner of the keystore has access to his keys. The *root* user (EFS administrator) cannot change the user's keystore password or access the encrypted user files in any other way. This results in high data security, but in the case of a user losing the password to his keystore, the files to which only he had access will be unrecoverable. It's easy to imagine a frustrated user who "forgets the password" or whose password becomes inaccessible due to various random events. Hence, this mode must only be used with full awareness and caution.

Backup and restore

File backups can be created in an encrypted or unencrypted form. By default, the unencrypted form is used (i.e., files are decrypted during backup). To make a backup in an encrypted form, use specially adapted *backup*, *mksysb*, or *savevg* commands. They feature the *-Z* parameter, which causes the backup of files in the form in which they appear in the file system. This means that both encrypted and unencrypted files can be included in one backup.

When making a backup in an encrypted form, there are a few additional things to keep in mind:

- ***User and group keystore*** - The user's files are in an encrypted form. No one will gain access to the files without the keys stored in the user or group keystore. The keystore should be periodically backed up.
- ***Keystore passwords*** - You can imagine a situation in which the files and keystore are restored from the backup. In the meantime, the user has changed the password to the keystore (*efskeymgr -n*). He no longer remembers the old password used to secure the keystore in the backup. In the admin mode, you can deal with this by changing the user's password. In the guard mode, such files are no longer recoverable. Therefore, it is also necessary to back up the user's keystore password.

An example of an encrypted and an unencrypted backup:

We make two backups of the encrypted file. One backup should be unencrypted and the other encrypted:

```
# efsmgr -l  /efsfs/encfile2   # The file is in an encrypted form.
EFS File information:
 Algorithm: AES_128_CBC
 Block Size: 4096
List of keys that can open the file:
 Key #1:
  Algorithm       : RSA_2048
  Who             : uid 0
  Key fingerprint : 41899dca:38cddaeb:b9853eaf:6230cae9:3de18eae
 Key #2:
  Algorithm       : RSA_4096
  Who             : uid 207
  Key fingerprint : d5b62fd2:d762c6c1:0ccd3e73:7b5e689f:9dbcc45a

# ls /efsfs/encfile2 | backup -i -Z -f /tmp/backup.enc   # Encrypted backup.
Mount volume 1 on /tmp/backup.enc.
        Press Enter to continue.
You have mail in /usr/spool/mail/root

# ls /efsfs/encfile2 | backup -i -f /tmp/backup.noenc   # Unencrypted backup.
Mount volume 1 on /tmp/backup.noenc.
        Press Enter to continue.

# ls -l /tmp/backup.*enc
-rw-r--r--    1 root     system        51200 Sep 11 03:11 /tmp/backup.enc
-rw-r--r--    1 root     system        51200 Sep 11 03:11 /tmp/backup.noenc
```

We delete the original file and then restore the backup file in an unencrypted form:

```
# rm /efsfs/encfile2

# restore -rvqf /tmp/backup.noenc
New volume on /tmp/backup.noenc:
Cluster size is 51200 bytes (100 blocks).
The volume number is 1.
The backup date is: Sun Sep 11 03:11:21 CDT 2016
Files are backed up by name.
The user is root.
x           16 /efsfs/encfile2
The total size is 16 bytes.
The number of restored files is 1.

# efsmgr -l  encfile2
Error getting EFS attributes: Cannot find the requested security attribute.
```

The file has been restored and is unencrypted.

We again delete the file and then restore it from the backup in an encrypted form:

```
# rm /efsfs/encfile2

# restore -rvqf /tmp/backup.enc
New volume on /tmp/backup.enc:
Cluster size is 51200 bytes (100 blocks).
The volume number is 1.
```

```
The backup date is: Sun Sep 11 03:11:05 CDT 2016
Files are backed up by name.
The user is root.
x         4096 /efsfs/encfile2
The total size is 4096 bytes.
The number of restored files is 1.

# efsmgr -l  /efsfs/encfile2
EFS File information:
 Algorithm: AES_128_CBC
 Block Size: 4096
List of keys that can open the file:
 Key #1:
  Algorithm       : RSA_2048
  Who             : uid 0
  Key fingerprint : 41899dca:38cddaeb:b9853eaf:6230cae9:3de18eae
 Key #2:
  Algorithm       : RSA_4096
  Who             : uid 207
  Key fingerprint : d5b62fd2:d762c6c1:0ccd3e73:7b5e689f:9dbcc45a
```

The file is restored in an encrypted form, with all the attributes and keys stored in its metadata.

The SMIT menu screens for the *mksysb* and *savevg* backups also have the option of backing up encrypted files (parameter *-Z* from the commands discussed above). This is presented in Figure 14-4.

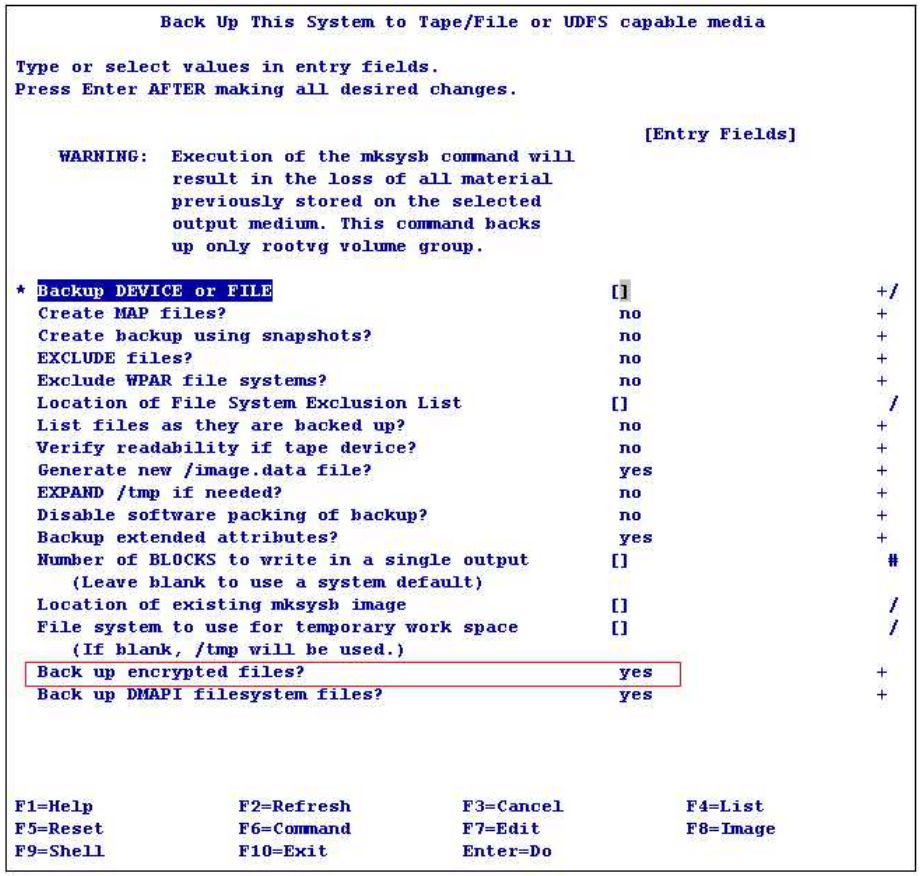

Figure 14-4: SMIT - mksysb.

Firewall

A firewall is an important part of any system's operation. In traditional configurations, traffic filtering does not take place at the level of the operating system, but rather at the level of external devices. This means that systems that communicate with each other send information outside even if they are on the same server. The external device determines whether the transmission can reach the destination or if it should be rejected. This approach makes sense from a security point of view, since it enables the management of communication between systems by a different group of people than the administrators of those systems. This increases the organization's security, and it is easy to control and audit. However, the reduction of performance is a negative aspect of this central approach. It can be seen in several aspects:

- Network traffic sometimes takes longer routes in order to be filtered through the firewall.
- Filtering devices have their own performance limitations in the form of computing power and the bandwidth.
- The use of external devices requires the implementation of network segmentation. Without segmentation, especially in the virtual world, systems could communicate internally through the hypervisor. Placing the systems in the same VLAN and thus eliminating routing in the communication between them would allow for an increase in performance, resulting in a reduction in packet delay.

A comparison of network traffic filtering at the levels of the operating system and external devices is shown in Figure 14-5.

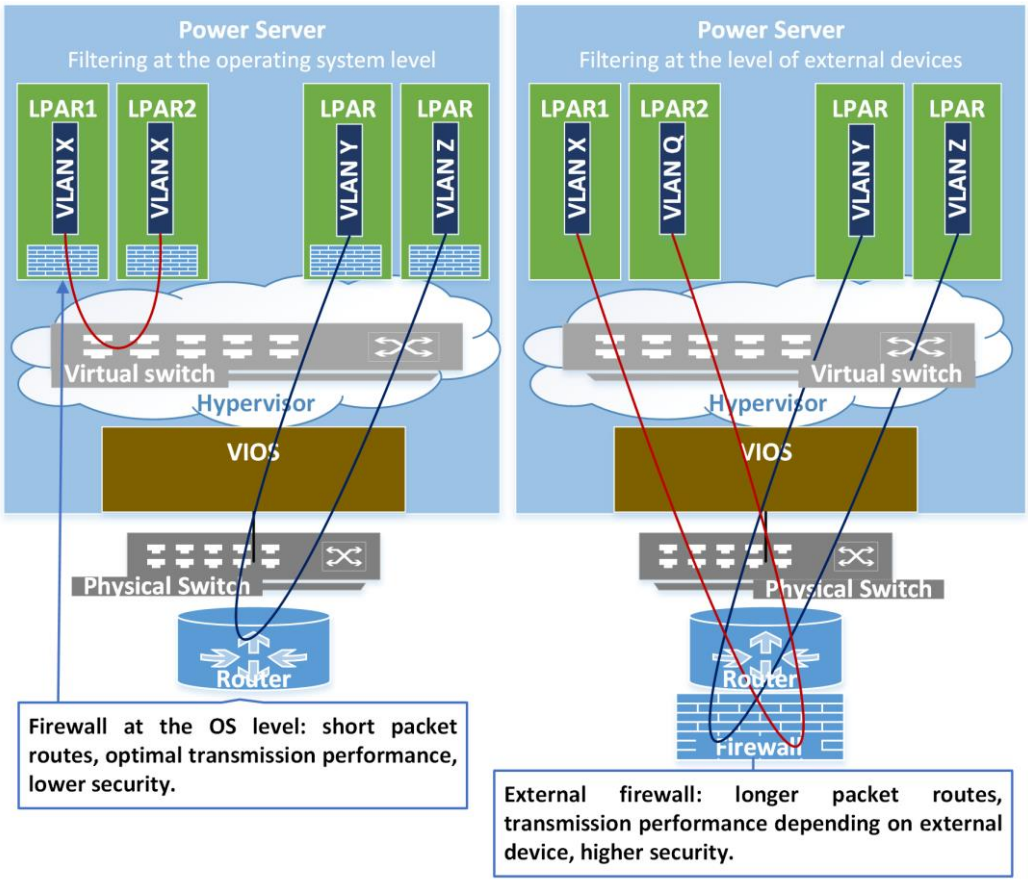

Figure 14-5: Internal firewall versus external firewall.

Both approaches to firewalling have their own advantages and disadvantages. The best approach is to maintain the centralized management of traffic filtering with the implementation of exceptions where important from the performance point of view. The use of exceptions in the form of traffic filtering within the operating system allows for the optimization of the network traffic of several systems. Optimization can be achieved by maintaining the addressing of systems in one subnet and one VLAN. This way, routing is eliminated, and systems can communicate via the hypervisor.

An important choice to be made when implementing rules is determining the rule that will end the list. The only safe rule that gives you full control over network traffic is the "*deny all*" rule, which blocks any traffic other than the traffic allowed in the preceding rules. However, in some situations, you can decide on the "*allow all*" rule, which allows any traffic that has not been explicitly blocked in the preceding rules. This approach is not completely safe, since you do not know exactly what you allow, although it is easier to manage and causes fewer problems in terms of accessing the system.

Traffic filtering in AIX

The operating system allows the implementation of a software firewall. This mechanism is part of the IPsec functionality described later in this chapter, although it can also be used independently. You can create multiple filtering rules in the operating system. These rules are automatically numbered when created (their order can be changed). They are processed in the context of each IP packet. Such processing follows the numbering until it encounters a rule that specifies the action to be performed.

The rules can be simple, such as *allow*/*deny*. They can also be more advanced and enable mechanisms to prevent port scanning, use patterns to filter packets by their contents, or have a complex structure (use of *if*, *else*, and *endif*). The possibilities on offer in relation to the rules configuration can be described based on the SMIT menu shown in Figure 14-6.

```
# smit ips4_add_filter
```

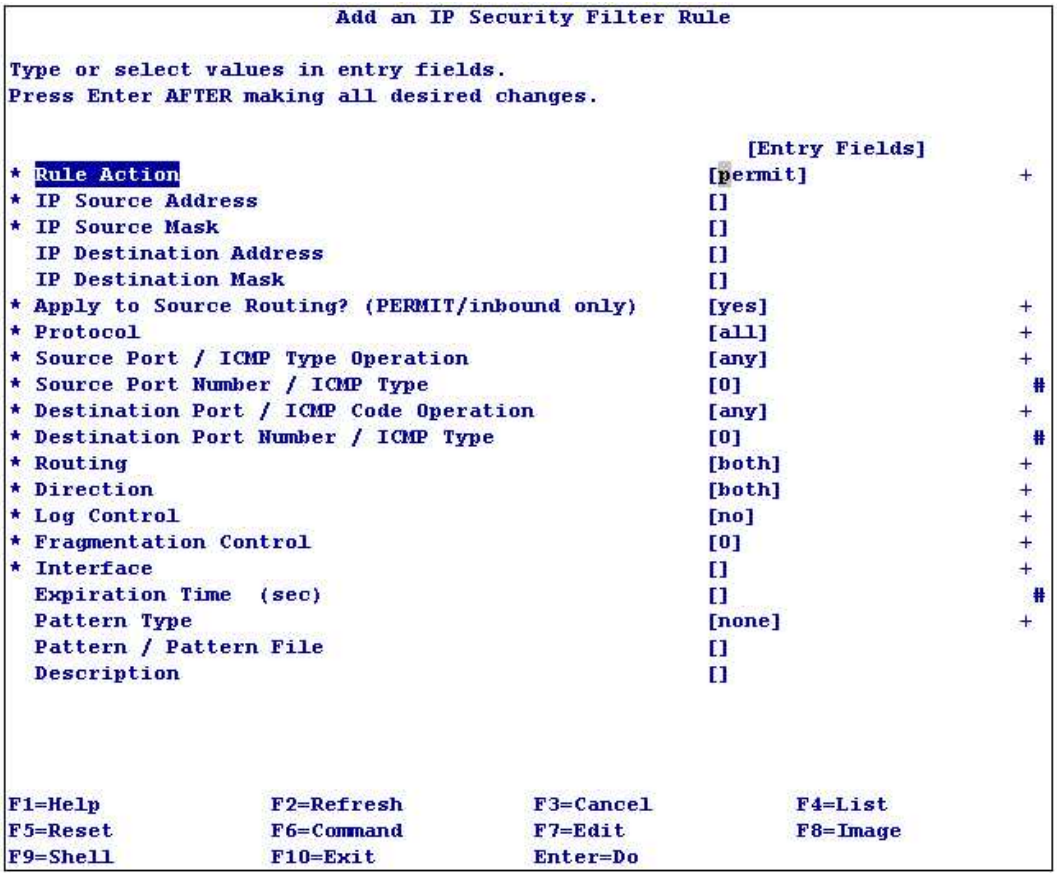

Figure 14-6: SMIT - Add an IP Security Filter Rule.

Description of individual fields:

- ***Rule Action*** - The action that the firewall has to take. There are several options to choose from:
  - ***permit***.
  - ***deny***.
  - ***shun_host, shun_port*** - Enabling the mechanism to prevent attacks. It is based on the typical characteristics of this type of event. Typically, the attack is preceded by port scanning by the host from which it is to occur. The individual system ports are scanned to identify whether a service is running on them. If a service is running, one can carry out the attack by knowing the service characteristics. As you also know this mechanism, you can set up a kind of trap on given ports. The trap is such that if there is an attempt to access a specific port, the system will behave as follows:
    - ***shun_host*** - Any transmission from the initiating host will be blocked for the time defined in the *Expiration Time* field.
    - ***shun_port*** - Traffic from the initiating host will be blocked, but only from the source port used for scanning. With this mechanism, you are able to protect yourself against attacks by setting traps on ports that your organization does not use.
  - ***if, else, endif*** - With these actions, you can build more complicated filtering rules based not only on a single packet, but on the whole session.

- ***IP Source Address/Mask, IP Destination Address/Mask*** - Specifies the IP addresses to which the rule applies. The rule can apply to individual addresses as well as to specific subnets, as indicated by the mask:
  - An example of a single host rule: *IP address 10.10.10.11*, *IP Mask 255.255.255.255*. The mask indicates that all the address bits are taken into account, so this is the address of a single host.
  - An example of a subnet rule: *IP address 10.10.10.0*, *IP Mask 255.255.255.0*. The mask indicates that the first three octets of the address should be taken into account, so this is the address of the subnet.

- ***Apply to Source Routing?*** - Specifies whether the rule applies to packages referred to as *source routed*. The routers decide on which path a given packet goes from the source to the destination in the IP network. Nevertheless, there is a method, namely *source routing*, that allows you to take this task away from routers in the context of a given transmission. In this case, the source determines the whole or partial path that the packet will travel to the destination using the additional information provided in the transmission. *Source routing* is a tool that supports hackers in attacks, and it is therefore blocked in many environments.

- ***Protocol*** - Protocol to which the rule applies.

- ***Source Port/Source Port Number*** - Two parameters that define the source ports to which the rule applies. In the *Source Port Number* field, a specific port number is indicated. In the *Source Port* field, the interpretation of this number is indicated:
  - ***any*** - Any port (*Source Port Number* should be empty).
  - ***eq*** - Applies only to the port listed in the *Source Port Number* field.
  - ***neq*** - Applies to all ports except the one mentioned.

  - ***lt*** - Applies to ports equal or less than the specified one.
  - ***gt*** - Applies to ports equal or greater than the one listed.
  - ***le*** - Applies to ports with a value less than the one specified.
  - ***ge*** - Applies to ports with a value greater than the one specified.

- ***Destination Port/Destination Port Number*** - Two parameters that define the destination ports affected by the rule. The method of interpretation is analogous to that of source ports.

- ***Routing*** - Defines the type of traffic to which the rule applies (*both*, *local*, or *route*).

- ***Direction*** - Defines the direction of the traffic for which the rule is supposed to work: *inbound*, *outbound*, or *both*.

- ***Log Control*** - Specifies whether to create an entry in the syslog whenever the package being processed matches this rule.

- ***Fragmentation Control*** - The parameter is related to the IPsec protocol. It specifies the tunnel to which the packet should be sent/received. A value of *0* indicates that the rule is not associated with any IPsec tunnel.

- ***Interface*** - Defines which interface the rule is related to (*all*, *en0*, *en1*, ...).

- ***Pattern Type, Pattern/Pattern File*** - Parameters that allow advanced packet filtering from the point of view of the data in the packet. The data transmitted in the packet can be compared with the patterns and, on this basis, the decision to either allow or deny the transmission can be made.

Tools for managing rules

You can use both the SMIT menu and the command line to manage the firewall and the associated rules. Two screens are available for this purpose in the SMIT menu:

- ***smit ips4_advanced*** - The options to enable, disable, and list the current rules and enter the configuration menu.
- ***smit ips4_conf_filter*** - The configuration menu, which is shown in Figure 14-7.

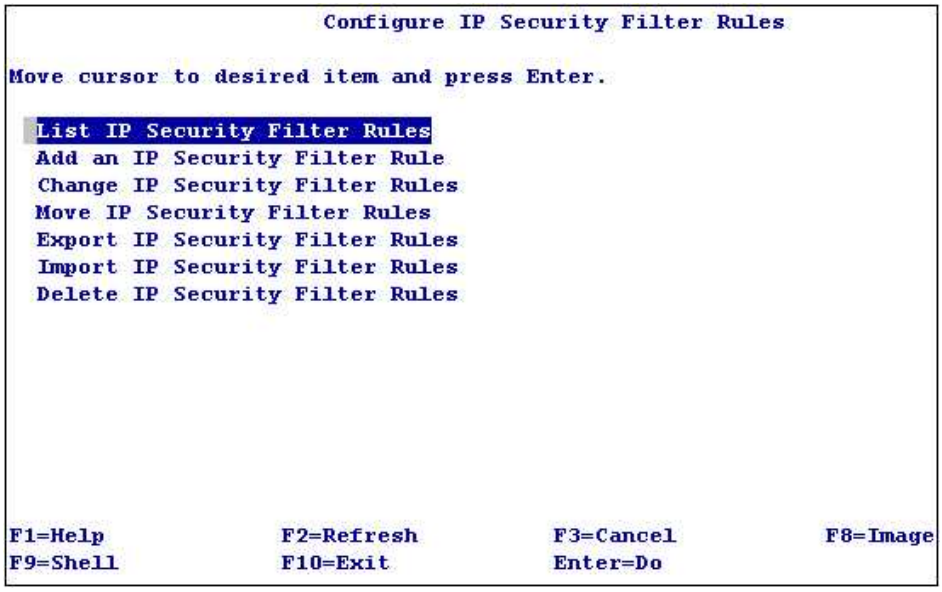

Figure 14-7: SMIT - Firewall options.

The SMIT menu works by using system commands. Therefore, the same options (usually extended) are available from the command line:

- */usr/sbin/lsfilt*, */usr/sbin/genfilt*, */usr/sbin/rmfilt* - These commands are used to list, create, and delete the rules that filter the traffic.

- */usr/sbin/mkfilt* - A key command for activating rules. After making changes to the rules table, this command will let you activate those changes.

- */usr/sbin/mvfilt* - Moving rules in the list. The order of the rules is important. The processing of the rules for a given package ends when the system encounters the first rule that meets the criteria. Thus, the efficiency of both packet processing and system load depends on the placement of rules that most often meet the criteria at the beginning of the list.

- */usr/sbin/ckfilt* - Verification of the rules. An option that is especially needed to verify the correctness of writing complex rules that were created using conditions (*if*, *else*).

- */usr/sbin/expfilt, /usr/sbin/impfilt* - Exports rules to a file and imports from the file in the rules table. This option is especially useful when you implement the same rules on many systems.

Scenarios

Starting the traffic filtering:

We start the packet filtering mechanism, without adding any rules, using the SMIT menu (Figure 14-8):

```
# smit ips4_start
```

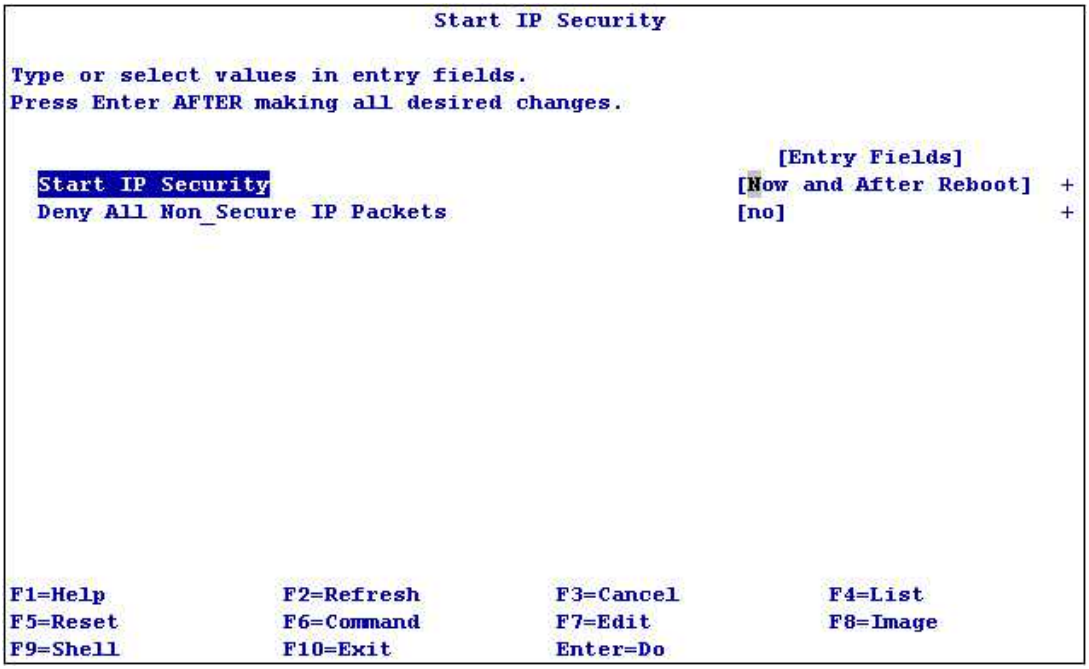

Figure 14-8: SMIT - Start IP Security.

We have a choice whether to run permanently (*Now and After Reboot*) or only until the restart. We can also decide how the rules should work:

- ***Deny All Non_Secure IP Packets = no*** - Sets the last rule to *Allow ALL*. This setting allows any traffic, except for the traffic that we explicitly deny by adding *Deny* rules.
- ***Deny All Non_Secure IP Packets = yes*** - Sets the last rule to *Deny ALL*. This is a more secure option. This setting denies all traffic, except for the traffic we explicitly allow by adding *Allow* rules.

In this scenario, we choose the first option, which does not block traffic by default. We run the menu and we obtain the following output:

```
ipsec_v4 Available
Default rule for IPv4 in ODM has been changed.
Successfully set default action to PERMIT
```

We verify the active rules:

```
# lsfilt -v4 -a
Beginning of IPv4 filter rules.
Rule 1:
*** Dynamic filter placement rule for IKE tunnels ***
Logging control     : no

Rule 2:
Rule action         : permit
Source Address      : 0.0.0.0
Source Mask         : 0.0.0.0
Destination Address : 0.0.0.0
Destination Mask    : 0.0.0.0
Source Routing      : yes
Protocol            : all
Source Port         : any 0
Destination Port    : any 0
Scope               : both
Direction           : both
Logging control     : no
Fragment control    : all packets
Tunnel ID number    : 0
Interface           : all
Auto-Generated      : no
Expiration Time     : 0
Description         :

End of IPv4 filter rules.
```

As you can see, only one rule is active, which allows all IP traffic. By starting the mechanism, we have not changed the way the system behaves. From now on, we can add detailed rules that block selected traffic.

Implementation of simple rules that block traffic:

To the above scenario, we add a rule that blocks all inbound traffic related to Telnet (Telnet works on port 23). Our server will not be able to act as a Telnet server, although we can run a Telnet client and connect to another device using this application. We add a rule that blocks ingoing traffic on port 23 from all hosts, as shown in Figure 14-9.

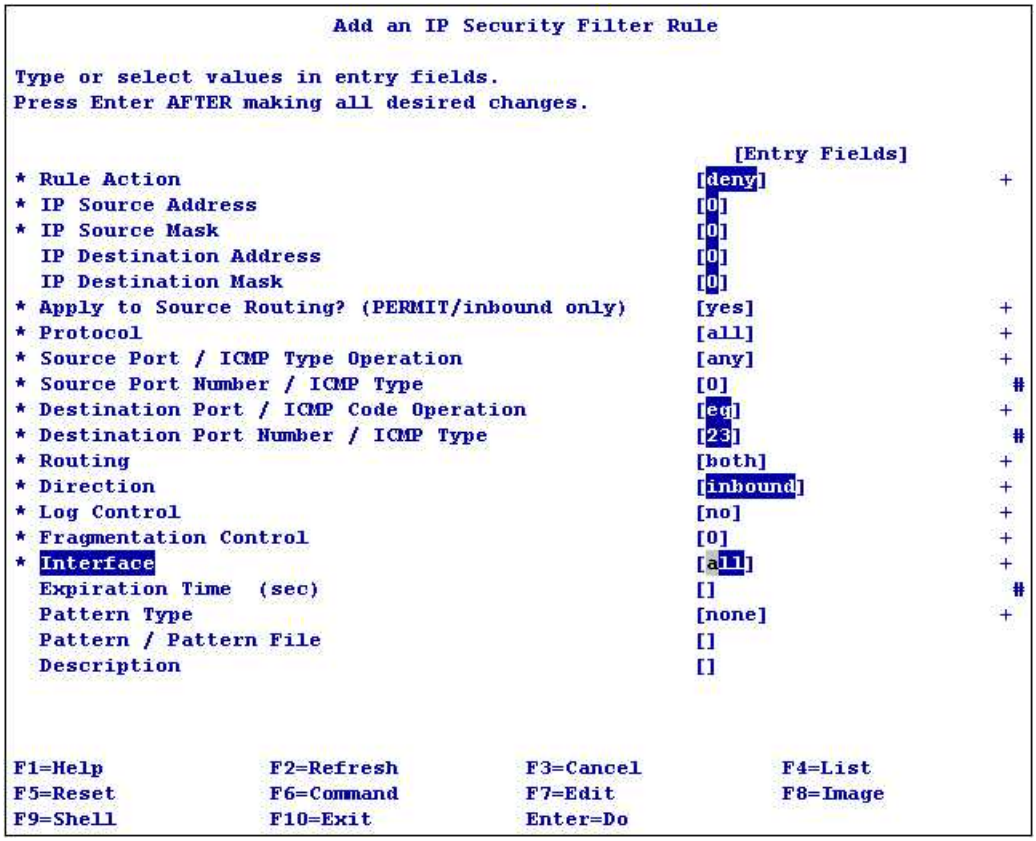

Figure 14-9: SMIT - Add an IP Security Filter Rule.

Instead of choosing the above options from the SMIT menu, you can run the following command.

```
# /usr/sbin/genfilt -v 4  -a 'D' -s '0' -m '0' -d '0' -M '0' -g 'y' -c 'all' -o 'any' -p '0' -O 'eq'
-P '23' -r 'B' -w 'I' -l 'N' -t '0' -i 'all'
```

After starting the command, the rule is saved, although it is not yet active. We check this using the *lsfilt* command:

```
# /usr/sbin/lsfilt -s -v 4 -a -O   # -a active rules.
1 *** Dynamic filter placement rule for IKE tunnels *** no
2 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 yes all any 0 any 0 both both no all packets 0 all 0 none
```

As you can see, only the default rule that allows all traffic is active. We thus activate all the defined rules:

```
# mkfilt -u
```

An active rule appears that blocks inbound traffic on port 23. From now on, nobody will be able to log on to the server using *Telnet*:

```
# /usr/sbin/lsfilt -s -v 4 -a -O
1 *** Dynamic filter placement rule for IKE tunnels *** no
2 deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 yes all any 0 eq 23 both inbound no all packets 0 all 0 none
```

```
3 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 yes all any 0 any 0 both both no all packets 0 all 0 none
```

Block all outbound traffic except SSH (port 22):

In this scenario, we will cut off the server from all traffic except the SSH traffic on port 22. The initial state shows that we have the following rules defined:

```
#  /usr/sbin/lsfilt -s -v 4 -O   # Displays all the defined rules.
1 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 no udp eq 4001 eq 4001 both both no all packets 0 all 0
none  Default Rule
2 *** Dynamic filter placement rule for IKE tunnels *** no
3 deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 yes all any 0 eq 23 both inbound no all packets 0 all 0 none
0 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 yes all any 0 any 0 both both no all packets 0 all 0 none
Default Rule
```

The active rules are:

```
# /usr/sbin/lsfilt -s -v 4 -a -O   # Display of the active rules.
1 *** Dynamic filter placement rule for IKE tunnels *** no
2 deny 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 yes all any 0 eq 23 both inbound no all packets 0 all 0 none
3 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 yes all any 0 any 0 both both no all packets 0 all 0 none
```

It is important to note that both command variants display a different rule numbering. When changing the rules, use the numbering displayed by */usr/sbin/lsfilt -s -v 4 -O*.

First, we remove the deny rule that was added in the above scenario:

```
#/usr/sbin/rmfilt -v 4  -n '2'   # Deleting rule number 2 (blocking Telnet).
```

Then, we change the default rule that allows all traffic to a rule that blocks all traffic:

```
#/usr/sbin/chfilt -v 4  -n '0'   # Change of the default rule (0).
```

Making changes does not affect the system until those changes are activated, so you can easily create a configuration. Two rules are added:

1. A rule that allows traffic to port 22 from any address and port. It allows *ssh* clients to connect to the server:

```
# /usr/sbin/genfilt -v 4 -a P -s 0 -m 0 -d 0 -M 0 -g y -c all -o any -p 0 -O eq -P 22 -r B -w B -l N
-t 0 -i all
```

2. A rule that allows traffic from port 22 to any address and port in the network. It allows the *ssh* service to respond to clients:

```
# /usr/sbin/genfilt -v 4 -a P -s 0 -m 0 -d 0 -M 0 -g y -c all -o eq -p 22 -O any -P 0 -r B -w B -l N
-t 0 -i all
```

The changes we have made are activated:

```
# mkfilt -u
```

As a result, all traffic is blocked, except for SSH traffic. To render this scenario more useful, it would be necessary to add further rules for the running applications.

IPSec

Nowadays, data transmission in an unsecured form is becoming less and less common. Due to using your network every day, you often do not even realize that most of your traffic is secured. When you connect to servers, you use the SSH protocol, while connections to applications usually use HTTPS. If our application connects to a database, it usually also uses an encrypted connection. Additionally, when you use the Internet at home, most transmissions are automatically secured.

The security methods mentioned above, such as SSH and HTTPS, apply to specific types of applications; hence, you cannot use them to secure any kind of traffic. If you want to secure the entire communication channel regardless of the type of traffic, then you can use IPsec.

IPsec is a set of protocols that allow you to create secure connections. The use of this set of protocols provides you with:

- **Authentication** - Guarantees that the packet you have received comes from the source declared within it.
- **Data integrity** - Ensures that the content of the information in the transmitted packet has not been changed during transmission.
- **Data confidentiality** (encryption) - Guarantees that no data are sent in a form that is readable to someone who intercepts it. Data encryption ensures this feature.
- **Replay protection** - Protects against attacks in which the attacker intercepts the packet and then uses it at a later point to hack the system.

With the help of IPsec, you can create virtual private networks (VPN) in a potentially dangerous environment. With the VPN, you can safely connect to the server through a dangerous environment, namely the Internet. In the same way, you can connect the company's headquarters without generating additional risks. Although a transmission in the VPN tunnel can be intercepted, you can still feel safe, since unauthorized persons will not be able to read or use it in any way.

Reports and operation modes

IPsec is mainly based on two protocols that allow you to build secure connections. These protocols are the *Authentication Header* (AH) and the *Encapsulating Security Payload* (ESP). Both protocols operate on the IP protocol and extend the IP packet with additional information to ensure security.

Authentication Header (AH) - Provides mechanisms for authentication, data integrity, and replay protection. However, it does not ensure data confidentiality. Therefore, when using only this protocol, we could be sure:

- About with whom we exchange information;
- That the information is not changed in the transmission path; and
- That an attack with the repetition of the captured information will fail.

However, the transmitted information could be read because it is not encrypted.

Encapsulating Security Payload (ESP) - In addition to the mechanisms listed in the AH protocol, which also guarantees the ESP itself, it also provides data confidentiality (encryption). Encryption is carried out using a symmetric key, which is known to both sides. Potentially, when creating a tunnel with all the security features, you can use only the ESP protocol. Nevertheless, such tunnels are usually created based on a combination of the AH and ESP.

Both protocols can operate in two modes (i.e., in transport mode and in tunnel mode):

Tunnel mode - The entire packet is encrypted or authenticated. Each packet passing through the tunnel is encapsulated within a new packet. It is worth noting here that the source and destination IP addresses are in the internal and external packets. This mode is used to create a VPN. It allows you to connect different locations, branches, and data centers. In this mode, you can set up different tunnel types: network to network, host to network, or between hosts. When connecting networks or data centers via a public network and IPsec, the tunnel mode is more secure than the transport mode. This is because it does not provide information about which IP addresses are communicating with each other. You can only see that two gateways communicate, while the addresses of specific hosts are encrypted in the encapsulated packet.

Transport mode - Unlike the tunnel mode, there is no encapsulation of the entire IP packet into a second packet. Instead, the packet sent in this mode is modified with additional information (in the ESP, it is encrypted) for security. In this mode, the source and destination IP addresses are only located in one place. Thus, you should mainly use it in host-to-host transmissions.

A comparison of the protocols and their possible modes of operation is presented in Figure 14-10.

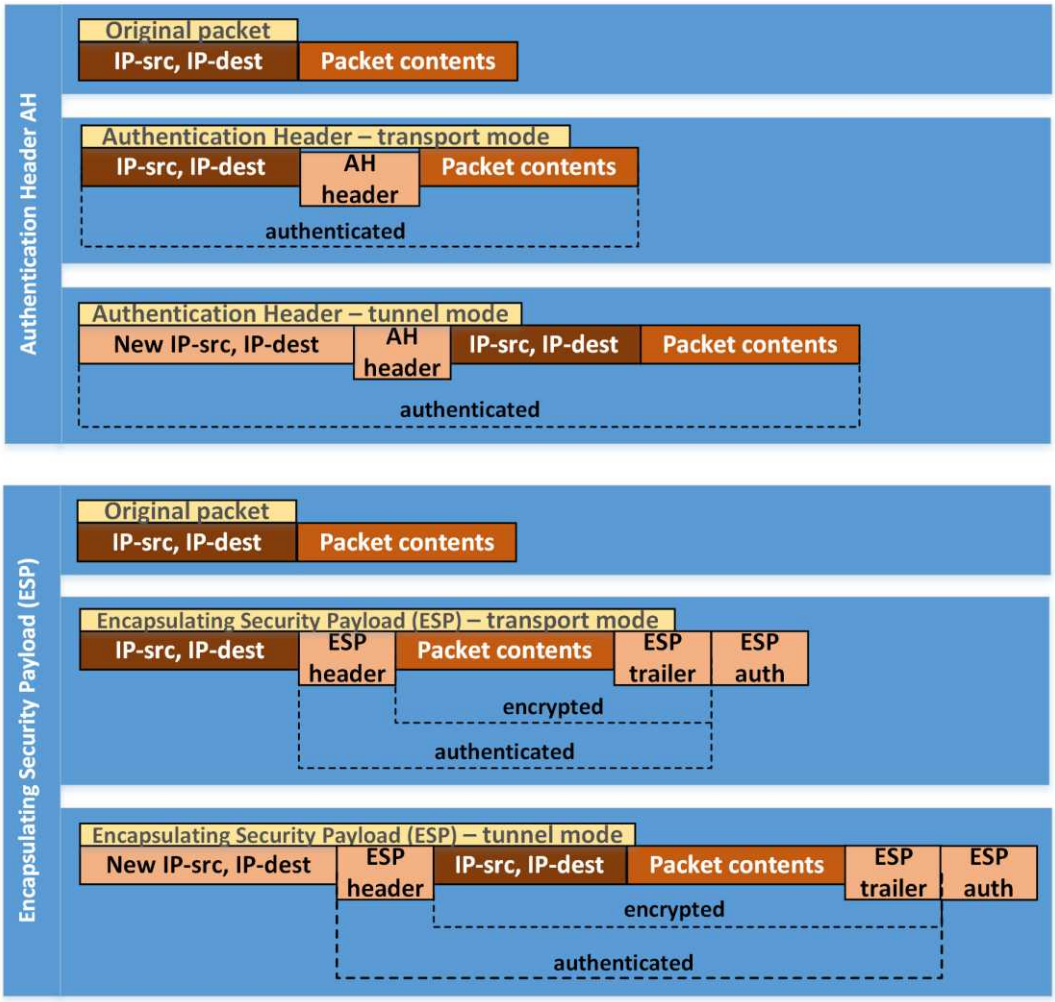

Figure 14-10: The AH and ESP.

IPsec tunnels in AIX

AIX allows you to set up encrypted tunnels. You can create two types of tunnels:

Manual tunnel - A tunnel created manually between devices. The way it works, as well as the keys used during transmission, are determined at the time of its creation. This type of tunnel is simple to implement, although it presents the disadvantage that it is difficult to change the keys or other parameters of the tunnel.

IKE (Internet Key Exchange) tunnel - A tunnel that is run using the Internet Key Exchange protocol set. It requires a much more complex configuration, and the creation of the tunnel takes place in two phases:

- Phase 1 - Setting a secure communication channel to meet the needs of phase 2.
- Phase 2 - Negotiation of the tunnel operation parameters (the so-called Security Association - SA).

With an approach that allows the negotiation of the parameters of a given tunnel, these parameters can be periodically and automatically changed. In particular, this applies to the encryption keys, since periodic changes in this regard increase the security of the connection.

In this chapter, we will focus on a small part of the possibilities of AIX in relation to IPsec (i.e., on manual tunnels). In the AIX system, you can configure a tunnel using the SMIT menu as well as the command line. The options shown in Figure 14-11 are available for creating and managing a manual tunnel.

```
# smit ips4_manual
```

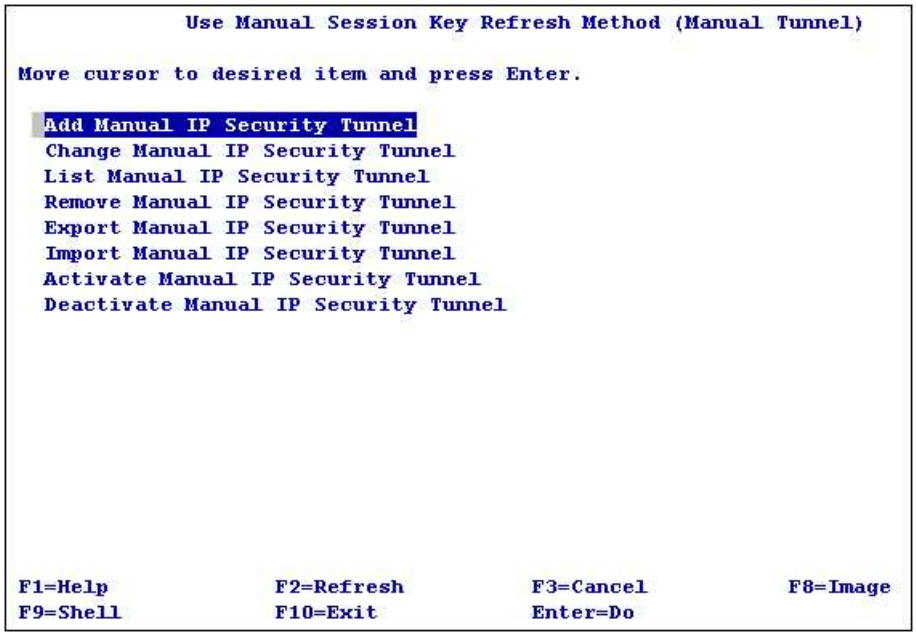

Figure 14-11: SMIT - Manual tunnel.

The descriptions of most of the above options are clear. However, the export/import option for the tunnel deserves attention. It allows you to create a tunnel from the perspective of one host and then import it on another, thereby simplifying the configuration process. Instead of a menu, you have a number of commands:

- */usr/sbin/chtun* - Change the tunnel definition.
- */usr/sbin/gentun* - Create the tunnel definition.
- */usr/sbin/lstun* - Show the definition of the tunnel.
- */usr/sbin/mktun* - Activate the tunnel.
- */usr/sbin/rmtun* - Deactivate the tunnel, remove the tunnel definition.

- */usr/sbin/exptun* - Export the tunnel definition.
- */usr/sbin/imptun* - Import the tunnel definition.
- */usr/sbin/ike* - Manage the IKE tunnels.

In the interests of clarity, it is best to describe the method of creating a manual tunnel based on the example of the SMIT menu. The menu shown in Figure 14-12 displays the data necessary for creating a manual tunnel between hosts. In this tunnel, the AH protocol will be used for authentication, and the ESP for encryption:

```
# smit ips4_add_man_tunnel_hh_ahesp
```

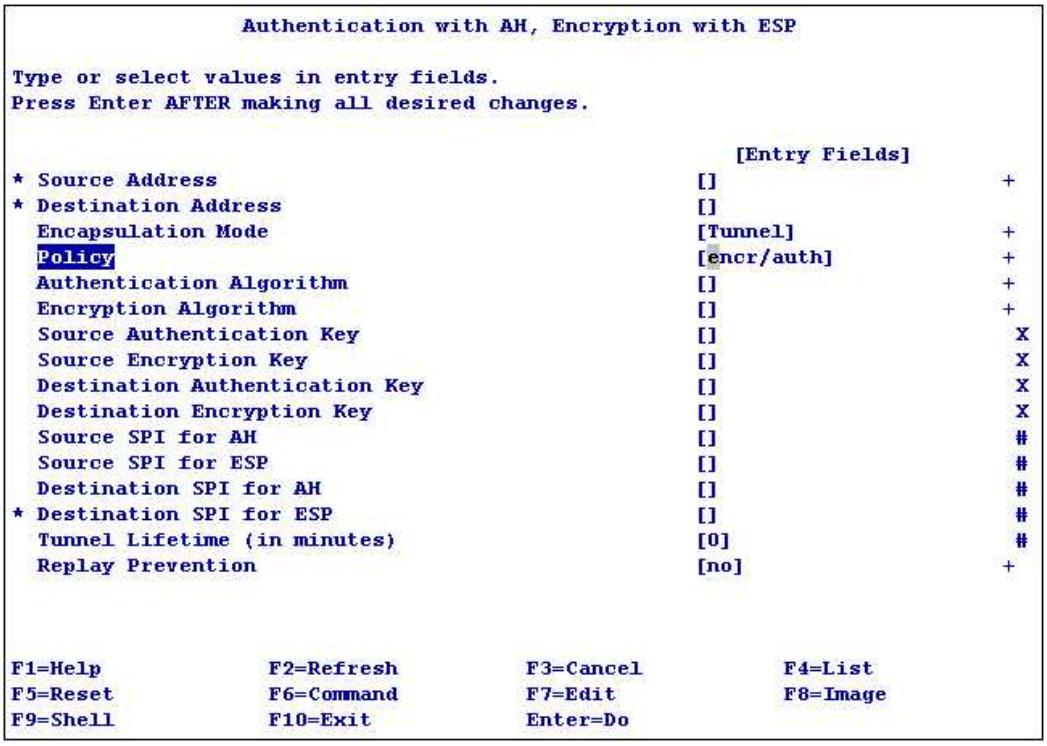

Figure 14-12: SMIT - Creation of the AH/ESP manual tunnel.

The menu options to be filled in:

- ***Source Address, Destination Address*** - The source and destination addresses of the hosts between which the tunnel will be built. A name may be used instead of the address, provided that it is interpreted by both hosts in the same way.
- ***Encapsulation Mode*** - The mode of operation described earlier. The tunnel and transport modes are available.
- ***Policy*** - Indicates the order in which the AH and ESP protocols are used. There are two options to choose from, *encr/auth* and *auth/encr*, indicating that the packet should be encrypted first (ESP) and then authenticated (AH), or vice versa.
- ***Authentication Algorithm*** - The algorithm to be used for authentication (the AH protocol). There are four values to choose from: *CMAC_AES_XCBC*, *HMAC_MD5*, *HMAC_SHA*,

and *KEYED_MD5*.

- ***Encryption Algorithm*** - The algorithm to be used to encrypt the packet. There are seven values to choose from: *3DES_CBC*, *AES_CBC_128*, *AES_CBC_192*, *AES_CBC_256*, *DES_CBC_4*, *DES_CBC_8*, and *NULL*.
- ***Source Authentication Key***, ***Source Encryption Key***, ***Destination Authentication Key***, ***Destination Encryption Key*** - The encryption keys that should be entered in the form of a hexadecimal notation. The fields can be left blank, which means that the keys will be generated automatically.
- ***Source SPI for AH***, ***Destination SPI for AH*** - The source and destination SPI (*Security Parameter Index*) for the *Authentication Header* protocol. The parameters used to identify the tunnel. The source SPI on the local machine must be equal to the target SPI on the remote machine, and vice versa. If the values are not given, the system will generate them automatically.
- ***Source SPI for ESP***, ***Destination SPI for ESP*** - The source and target SPIs (*Security Parameter Index*) for the *Encapsulating Security Payload* protocol. The parameters used to identify the tunnel. The source SPI on the local machine must be equal to the target SPI on the remote machine, and vice versa. If no values are given, the system will generate them automatically, except for the *Destination* value, which must be explicitly specified (any 32-bit value greater than 255).
- ***Tunnel Lifetime (in minutes)*** - Lifetime of the tunnel. After the time has elapsed, the tunnel will be deactivated. For manual tunnels, the value 0 (*never expire*) is usually used.
- ***Replay Prevention*** - Use the counter in the AH header to eliminate attacks with repeated transmission. Due to the specificity of manual tunnels, setting this value is not recommended.

During the activation of the tunnel, the appropriate filters are automatically activated (the functionality was described earlier in this chapter). Filters are necessary to redirect the appropriate traffic to the appropriate IPsec tunnel.

Tunnel and filter operation monitoring is available via the *syslogd* subsystem. Tunnel and filter events in the context of the IPv4 protocol are available in the *local4* class. The method chosen to enable the monitoring will show one of the following scenarios.

Scenarios

Starting IPsec:

This scenario is analogous to the "Starting the traffic filtering" scenario described in the chapter concerning the Firewll. We start IPsec on the hosts that are to set up the IPsec tunnel:

```
# smit ips4_start
```

After starting, we verify the information about the tunnel:

```
# lstun -v4 -p manual   # We list information about the IPv4 manual tunnel.
Start list manual tunnel for IPv4
No manual tunnel found for IPv4
End of manual tunnel for IPv4
```

As you can see above, there is no tunnel available.

Creation of an AH/ESP manual tunnel between two nodes:

In this scenario, we create a secure connection (AH/ESP manual tunnel) between two hosts with AIX 7.2:

- Host1 with the IP address: 10.10.177.3.
- Host2 with the IP address: 10.10.177.4.

IPsec is running on both hosts. We create a tunnel on Host1, which is shown in Figure 14-13.

```
# smit ips4_add_man_tunnel_hh_ahesp
```

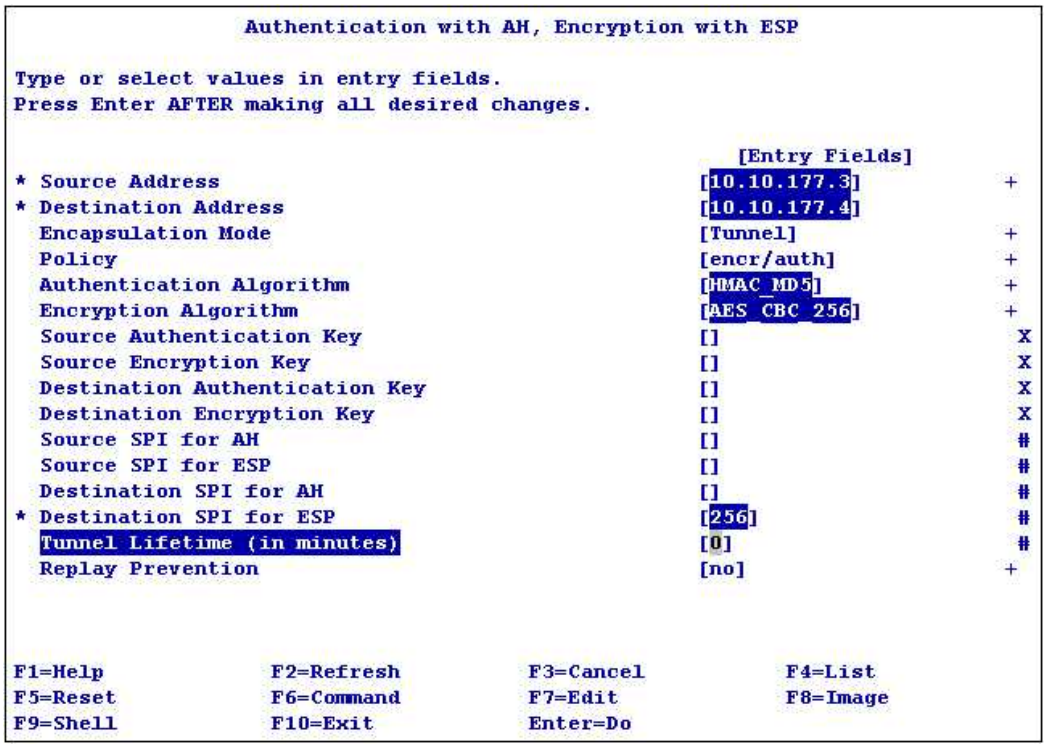

Figure 14-13: SMIT - Tunnel creation, scenario.

When creating the tunnel, we provide the source and destination addresses of the hosts. We do not change the tunnel operation mode or the policy, which is: first to encrypt (ESP), then to authenticate (AH). We select the algorithms that will be used by the tunnel and select the *Destination SPI* (any 32-bit number above 255). We leave the remaining fields blank for automatic use by the system. After running, we receive the following output:

```
Tunnel 1 for IPv4 has been added successfully.
Filter rule 5 automatically generated for this tunnel.
Filter rule 6 automatically generated for this tunnel.
```

The information concerns the creation of tunnel number 1 and the two filtering rules that direct traffic

between the hosts to the IPsec tunnel.

The tunnel has been created, but it is not yet active. The fields that we left empty were automatically filled:

```
# lstun -v4 -p manual

Start list manual tunnel for IPv4

Tunnel ID        : 1
IP Version       : IPv4
Source           : 10.10.177.3
Destination      : 10.10.177.4
Policy           : eaae
Tunnel Mode      : Tunnel
Source AH Algo   : HMAC_MD5
Source ESP Algo  : AES_CBC_256
Dest AH Algo     : HMAC_MD5
Dest ESP Algo    : AES_CBC_256
Source AH SPI    : 256   # SPI parameters are filled automatically.
Source ESP SPI   : 256
Dest AH SPI      : 256
Dest ESP SPI     : 256
Tunnel Life Time : 0
Status           : Inactive    # The tunnel is inactive.
Target           : -
Target Mask      : -
Replay           : No
New Header       : Yes
Snd ENC-MAC Algo : -
Dst ENC-MAC Algo : -

End of manual tunnel for IPv4
```

Two filter rules have been created that redirect the traffic between the hosts to tunnel number 1 (the rules are also inactive):

```
# /usr/sbin/lsfilt -s -v 4  -O | grep 10.10.177.3
5 permit 10.10.177.3 255.255.255.255 10.10.177.4 255.255.255.255 yes all any 0 any 0 both inbound no
all packets 1 all 0 none
6 permit 10.10.177.4 255.255.255.255 10.10.177.3 255.255.255.255 yes all any 0 any 0 both outbound
no all packets 1 all 0 none
```

The tunnel is defined on *Host1*. The next step is to copy the definition to *Host2*, so we export the definition of the tunnel to a file:

```
# exptun -v 4 -f /tmp -t 1
Manual tunnel(s) has been exported to /tmp/ipsec_tun_manu.exp successfully.
```

We copy the file to *Host2* and then import the tunnel definition:

```
# imptun -v 4 -f /tmp
IBM tunnels not found.
IPv4 tunnel 1 imported as 1.
Filter rule 5 automatically generated for tunnel 1.
Filter rule 6 automatically generated for tunnel 1.
```

Tunnel 1 has been imported and filter rules have been automatically created to redirect the traffic between the two defined hosts to the IPsec tunnel. The tunnel and the rules are inactive:

```
# lstun -v4 -p manual

Start list manual tunnel for IPv4

Tunnel ID        : 1
IP Version       : IPv4
Source           : 10.10.177.4
Destination      : 10.10.177.3
Policy           : aeea
Tunnel Mode      : Tunnel
Source AH Algo   : HMAC_MD5
Source ESP Algo  : AES_CBC_256
Dest AH Algo     : HMAC_MD5
Dest ESP Algo    : AES_CBC_256
Source AH SPI    : 256
Source ESP SPI   : 256
Dest AH SPI      : 256
Dest ESP SPI     : 256
Tunnel Life Time : 0
Status           : Inactive
Target           : -
Target Mask      : -
Replay           : No
New Header       : Yes
Snd ENC-MAC Algo : -
Dst ENC-MAC Algo : -

End of manual tunnel for IPv4
```

With definitions on both hosts, we can proceed to start the tunnel. We activate the tunnel at *Host1*:

```
# mktun -v 4 -t 1
```

After activation on one host, we verify the communication:

```
# ping 10.10.177.4
PING 10.10.177.4: (10.10.177.4): 56 data bytes
```

There was no communication between the hosts, since the tunnel was only activated on one of them. Two filter rules were activated when the tunnel was activated on the host. All the traffic to Host2 was sent through an IPsec tunnel that was not yet compiled, so it could not reach the destination.

We activate the tunnel on Host2:

```
# mktun -v 4 -t 1
# ping 10.10.177.4
PING 10.10.177.4: (10.10.177.4): 56 data bytes
64 bytes from 10.10.177.4: icmp_seq=0 ttl=255 time=0 ms
64 bytes from 10.10.177.4: icmp_seq=1 ttl=255 time=0 ms
```

From that moment, the tunnel is activated on both hosts, so the IP traffic between them can go through in a secure way.

Activation of monitoring:

IPsec events are monitored using the *syslogd* subsystem. So, we add the following line to */etc/syslog.conf*:

```
local4.debug    /var/adm/ipsec_debug.log   # Log all information - Debug.
```

We create a log file and refresh the configuration of the *syslogd* daemon:

```
# touch /var/adm/ipsec_debug.log
# refresh -s syslogd
0513-095 The request for subsystem refresh was completed successfully.
```

To verify the monitoring, we stop the tunnel and observe the entries that appear in the log:

```
# rmtun -v 4 -t 1
Tunnel (1) for IPv4 has been removed successfully.
Filter support (level 1.00) initialized at 05:27:16 on 09/25/16

Filter rules for IPv4. has been updated.

# cat /var/adm/ipsec_debug.log
Sep 25 05:27:16 AIX72c1 local4:debug rmtun: Setting V4 tunnel 1 in ODM to inactive
Sep 25 05:27:16 AIX72c1 local4:debug rmtun: Deleting V4 tunnel 1
Sep 25 05:27:16 AIX72c1 local4:notice : mkfilt: Filter rules updated at 05:27:16 on 09/25/16
Sep 25 05:27:16 AIX72c1 local4:notice : mkfilt: #:1 PERMIT 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 ah any 0
any 0  both both l=n f=y t=0 t_exp=0
Sep 25 05:27:16 AIX72c1 local4:notice : mkfilt: #:2 PERMIT 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 esp any 0
any 0  both both l=n f=y t=0 t_exp=0
Sep 25 05:27:16 AIX72c1 local4:notice : mkfilt: #:3 PERMIT 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 all any 0
any 0  both both l=n f=y t=0 t_exp=0
Sep 25 05:27:16 AIX72c1 local4:notice : mkfilt: #:4 not known 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 (0)
??? 0 ??? 0  both both l=n f=n t=0 t_exp=0
Sep 25 05:27:16 AIX72c1 local4:notice : mkfilt: #:5 not known 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 (0)
??? 0 ??? 0  both both l=n f=n t=0 t_exp=0
Sep 25 05:27:16 AIX72c1 local4:notice : mkfilt: #:6 not known 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 (0)
??? 0 ??? 0  both both l=n f=n t=0 t_exp=0
Sep 25 05:27:16 AIX72c1 local4:notice : mkfilt: Filter support (level 1.00) initialized at 05:27:16
on 09/25/16
Sep 25 05:27:16 AIX72c1 local4:info rmtun: Filter rules for IPv4. has been updated.
```

As you can see, changes to the tunnel and the related filters are recorded in the prepared file.

Chapter 15. Basic Tools for Testing and Managing Performance

Below, you will find information about the tools that can be used to test and manage system performance. Only their basic features will be described, which show just a fraction of what can be obtained using them. The goal of this chapter is to inform the reader about the existence of certain tools and their basic functionality. If you want to learn more about the tools, consult the existing documentation as well as the publications indicated in the introduction to this book.

Monitoring

Sar (System Activity Report) - A standard tool for monitoring performance in UNIX systems. It is available by default after installing the system. It allows the observation of many aspects of the load that is taking place at a given moment or that took place in the past. Data from the past can be viewed, since it is possible to run the program in the statistics collection mode.

Examples:

```
# sar 1 3   # Shows statistics from one-second periods three times.

AIX aix72c2 2 7 00F621774C00    02/22/17

System configuration: lcpu=8 ent=0.20 mode=Uncapped

05:14:43    %usr    %sys    %wio   %idle   physc   %entc
05:14:44       9      55       0      36    1.09   544.4
05:14:45      10      55       0      35    1.02   507.6
05:14:46       9      55       0      36    1.07   532.6

Average        9      55       0      35    1.06   528.2
```

vmstat - A standard tool for monitoring performance on UNIX systems. It provides statistics about the kernel, virtual memory, disk, and processor threads.

Examples:

```
# vmstat 1 3   # Shows statistics from one-second periods three times.

System configuration: lcpu=8 mem=2048MB ent=0.20

kthr    memory              page              faults              cpu
----- ----------- ------------------------ ------------ -----------------------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa    pc    ec
 4  0 273477 221062   0   0   0   0    0   0 1086 410071 1035  9 55 36  0  1.08 542.1
 4  0 272697 221842   0   0   0   0    0   0 161 411018 333 10 55 36  0  1.02 508.7
 4  0 273477 221062   0   0   0   0    0   0 931 409235 908  9 55 36  0  1.07 535.4
```

iostat - A standard tool for monitoring performance on UNIX systems. It provides statistics on processors as well as input-output operations on terminals and individual disks.

```
# iostat 1 1   # Shows statistics from one-second periods.
```

```
System configuration: lcpu=8 drives=3 ent=0.20 paths=3 vdisks=2

tty:      tin         tout    avg-cpu: % user % sys % idle % iowait physc % entc
          0.0        781.0                8.9  54.5   36.2      0.3   1.1  545.3

Disks:         % tm_act     Kbps      tps    Kb_read   Kb_wrtn
cd0               0.0       0.0       0.0          0         0
hdisk0           99.0     285440.0     1115.0     285440         0
hdisk1            0.0       0.0       0.0          0         0
```

lparstat - A tool that allows the monitoring of all aspects related to virtualization and LPARs. It was discussed in the chapter on virtualization.

Examples:

```
# lparstat   # General statistics from the LPAR perspective.

System configuration: type=Shared mode=Uncapped smt=4 lcpu=8 mem=2048MB psize=28 ent=0.20

%user  %sys  %wait  %idle physc %entc  lbusy  vcsw phint
----- ----- ------ ------ ----- ----- ------ ----- -----
  0.0   0.2    0.8   99.0  0.00   0.3    7.8 3832851   939

# lparstat -I   # Shows the LPAR parameters.
Node Name                                  : aix72c2
Partition Name                             : SB5437_AIX72_c2
Partition Number                           : 12
Type                                       : Shared-SMT-4
Mode                                       : Uncapped
Entitled Capacity                          : 0.20
Partition Group-ID                         : 32780
Shared Pool ID                             : 0
Online Virtual CPUs                        : 2
Maximum Virtual CPUs                       : 4
Minimum Virtual CPUs                       : 1
Online Memory                              : 2048 MB
Maximum Memory                             : 4096 MB
Minimum Memory                             : 256 MB
Variable Capacity Weight                   : 128
Minimum Capacity                           : 0.10
Maximum Capacity                           : 4.00
Capacity Increment                         : 0.01
Maximum Physical CPUs in system            : 32
Active Physical CPUs in system             : 32
Active CPUs in Pool                        : 28
Shared Physical CPUs in system             : 28
Maximum Capacity of Pool                   : 2800
Entitled Capacity of Pool                  : 1220
Unallocated Capacity                       : 0.00
Physical CPU Percentage                    : 10.00%
Unallocated Weight                         : 0
Memory Mode                                : Dedicated
Total I/O Memory Entitlement               : -
Variable Memory Capacity Weight            : -
Memory Pool ID                             : -
Physical Memory in the Pool                : -
Hypervisor Page Size                       : -
Unallocated Variable Memory Capacity Weight: -
Unallocated I/O Memory entitlement         : -
Memory Group ID of LPAR                    : -
Desired Virtual CPUs                       : 2
Desired Memory                             : 2048 MB
Desired Variable Capacity Weight           : 128
```

```
Desired Capacity                           : 0.20
Target Memory Expansion Factor             : -
Target Memory Expansion Size               : -
Power Saving Mode                          : Disabled
Sub Processor Mode                         : -
```

lvmstat - Provides statistics about input-output operations performed on individual logical volumes. You can also view these statistics according to individual logical partitions.

Examples:

```
# lvmstat -v rootvg   # Statistics for I/O operations, broken down by logical volumes.

Logical Volume       iocnt   Kb_read   Kb_wrtn      Kbps
  hd4               156255  39522340       392  44404.39
  hd2                 6445     25480       872     29.61
  hd9var               161       120       636      0.85
  hd8                  152         0       608      0.68
  hd3                  132       480        48      0.59
  hd10opt               18        56        16      0.08
  hd11admin              2         8         0      0.01
  hd1                    2         8         0      0.01
  dane                   0         0         0      0.00
  livedump               0         0         0      0.00
  lg_dumplv              0         0         0      0.00
  hd6                    0         0         0      0.00
  hd5                    0         0         0      0.00
# lvmstat -l hd4   # I/O statistics for the hd4 logical volume, broken down by individual logical
partitions.
Log_part  mirror#  iocnt   Kb_read   Kb_wrtn      Kbps
      10       1   2970     14152      1360      9.92
      12       1   1698     13788       192      8.94
      11       1   1526     14692      1312     10.23
       4       1    975     16384         0     10.48
       9       1    893     15720       576     10.42
       8       1    872     16360         0     10.46
       5       1    491     15900       340     10.39
       1       1    441      4308       452      3.04
       3       1    350     15604       556     10.33
       7       1    283     15616       540     10.33
       6       1    241      1416         0      0.91
       2       1    101      2488         0      1.59
      13       1     93      3320        72      2.17
      14       1      0         0         0      0.00
      15       1      0         0         0      0.00
      16       1      0         0         0      0.00
      17       1      0         0         0      0.00
      18       1      0         0         0      0.00
      19       1      0         0         0      0.00
      20       1      0         0         0      0.00
      21       1      0         0         0      0.00
```

mpstat - Displays performance statistics for the processors in the system.

Examples:

```
# mpstat 10 2   # Displays two general statistics from the subsequent ten-second periods.

System configuration: lcpu=8 ent=0.2 mode=Uncapped

cpu  min  maj  mpc  int   cs  ics   rq  mig lpa sysc us sy wa id   pc  %ec  lcs
  0  113    0    0  268   91    3    1    3 100 15077 15 83  0  3 0.03  2.7  209
```

```
  1    0    0    0   10    0    0    0    1 100    0  0  1  0 99 0.01  0.5   11
  2    0    0    0   10    0    0    0    1 100    0  0  1  0 99 0.01  0.5   11
  3    0    0    0   10    0    0    0    1 100    0  0  1  0 99 0.01  0.5   11
  4  323    0    0  144   14    9    1    1 100 426486 16 84  0  0 0.62 61.4  102
  5    0    0    0   10    0    0    0    1 100    0  0  0  0 100 0.11 11.5   11
  6    0    0    0   10    0    0    0    1 100    0  0  0  0 100 0.11 11.5   11
  7    0    0    0   10    0    0    0    1 100    0  0  0  0 100 0.11 11.5   11
ALL  436    0    0  472  105   12    2   10 100 441563 10 54  0 36 1.00 501.9  377
-------------------------------------------------------------------------------------
  0  305    0    0  308  140   62    2    6 100 339822 15 85  0  0 0.53 53.8  119
  1    0    0    0   11    0    0    0    1 100    0  0  0  0 100 0.10 10.0   12
  2    0    0    0   10    0    0    0    1 100    0  0  0  0 100 0.10 10.0   11
  3    0    0    0   10    0    0    0    1 100    0  0  0  0 100 0.10  9.9   19
  4  107    0    0  111   17    2    0   14 100 68206 15 84  0  0 0.11 10.6  115
  5    0    0    0   10    0    0    0    1 100    0  0  0  0 100 0.02  2.1   11
  6    0    0    0   10    0    0    0    1 100    0  0  0  0 100 0.02  2.0   11
  7    0    0    0   10    0    0    0    1 100    0  0  0  0 100 0.02  2.0   11
ALL  412    0    0  480  157   64    2   26 100 408028 10 55  0 36 1.00 497.9  309
```

topas - A program that displays the current system load. It is the most user-friendly approach, since it presents information in a user-friendly way.

Examples:

```
Topas Monitor for host:aix72c2                  EVENTS/QUEUES    FILE/TTY
Wed Feb 22 05:36:40 2017   Interval:2           Cswitch     205  Readch    33540
                                                Syscall  434.7K  Writech    1159
CPU     User% Kern% Wait% Idle%   Physc  Entc%  Reads        21  Rawin        0
Total    10.4  53.6   0.0  36.0    1.01 502.51  Writes       37  Ttyout     891
                                                Forks         2  Igets        0
Network    BPS  I-Pkts  O-Pkts    B-In   B-Out  Execs         2  Namei    229.6K
Total    1.85K   11.50    6.00   615.5   1.25K  Runqueue   4.50  Dirblk       0
                                                Waitqueue   0.0
Disk    Busy%      BPS     TPS  B-Read  B-Writ                   MEMORY
Total     0.0        0       0       0       0  PAGING           Real,MB    2048
                                                Faults       440  % Comp     53
FileSystem          BPS    TPS  B-Read  B-Writ  Steals         0  % Noncomp  18
Total             32.7K  20.50   32.7K       0  PgspIn         0  % Client   18
                                                PgspOut        0
Name            PID  CPU%  PgSp Owner           PageIn         0  PAGING SPACE
find        9044436  1.5  364K root            PageOut        0  Size,MB     512
ksh         5964164  0.1  568K root            Sios           0  % Used      1
topas       6816086  0.1 2.46M root                              % Free     99
getty       8192256  0.0  640K root            NFS (calls/sec)
gil         1573172  0.0  960K root            SerV2          0  WPAR Activ    0
sshd        5570840  0.0 1.19M root            CliV2          0  WPAR Total    0
clcomd      6619604  0.0 1.71M root            SerV3          0  Press: "h"-help
lock_rcv    3342706  0.0  448K root            CliV3          0         "q"-quit
sec_rcv     3473774  0.0  448K root            SerV4          0
aso         5046690  0.0 1.39M root            CliV4          0
sendmail    5177762  0.0 1.16M root
```

Topas can present various data in the window. To switch between views, use the keyboard. The key that you need to press to go to a specific view can be checked by pressing *h* (help).

nmon (topas_nmon) - A tool that is similar in function to *topas*. It also provides data in a user-friendly way that facilitates the user's ongoing monitoring.

Examples:

```
|_topas_nmon__t=Top-Processes____Host=aix72c2________Refresh=2 secs___05:47.56___|
| CPU-Utilisation-Small-View ___________EntitledCPU=  0.20 UsedCPU=  1.059_______|
|Logical  CPUs              0----------25-----------50----------75----------100  |
|CPU User%  Sys% Wait% Idle%|           |            |           |            |  |
|  0   0.0   4.5  58.0  37.5|ssWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW                 >  |
|  1   0.0   0.0   0.0 100.0|>                                                 |  |
|  2   0.0   0.0   0.0 100.0|  >                                               |  |
|  3   0.0   0.0   0.0 100.0|>                                                 |  |
|  4  12.5  87.5   0.0   0.0|UUUUUUsssssssssssssssssssssssssssssssssssssssssss>  |
|  5   0.0   0.0   0.0 100.0|  >                                               |  |
|  6   0.0   0.0   0.0 100.0|  >                                               |  |
|  7   0.0   0.0   0.0 100.0|  >                                               |  |
|EntitleCapacity/VirtualCPU +-----------|------------|-----------|------------+  |
|EC+   9.2  54.9   0.2  35.7|UUUUsssssssssssssssssssssssssiiiiiiiiiiiiiiii-|  |
| VP   4.9  29.0   0.1  18.9|UUsssssssssssssiiiiiiii-----------------------|  |
|EC= 529.4%  VP=  52.9%     +--No Cap---|------------|-SMT=4-----100% VP=2 CPU+  |
| Disk-KBytes/second-(K=1024,M=1024*1024) _____________________________________|
|Disk     Busy  Read  Write 0----------25-----------50------------75--------100  |
| Name          KB/s   KB/s |           |            |             |          |  |
|cd0        0%      0      0|                                                  |  |
|hdisk0    59% 172024      0|RRRRRRRRRRRRRRRRRRRRRRRRRRRRR     >               |  |
|hdisk1     0%      0      0|                                                  |  |
|Totals       172024      0+-----------|------------|-------------|----------+  |
|_____________________________________________________________________________|
```

It has an advantage over *topas* in the form of the larger number of parameters that it is possible to monitor as well as the availability of a tool that facilitates the analysis of results (i.e., *nmon analyser*). This tool enables the presentation of the collected data in the form of a spreadsheet in Excel, with automatically generated charts concerning various aspects of system operation. An example chart is presented in Figure 15-1.

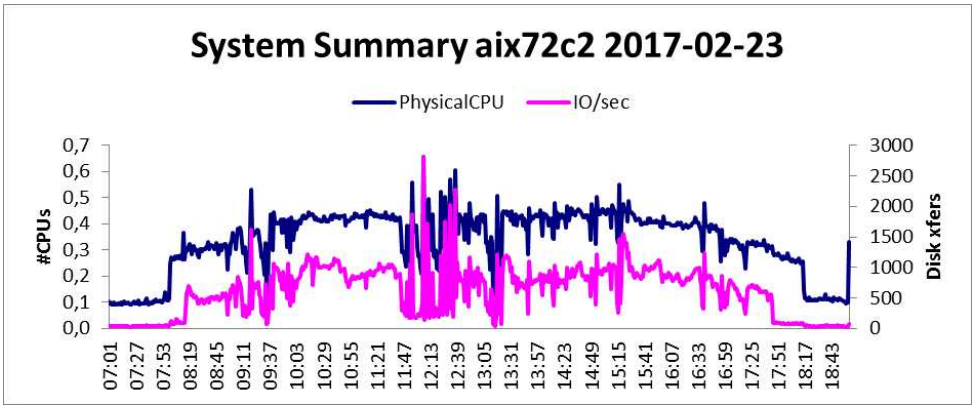

Figure 15-1: Sample chart from the nmon analyzer.

svmon - A command that displays information about the current memory state. To understand a large part of the information displayed by this command, a deep knowledge of memory management on the part of the operating system is required.

Examples:

```
# svmon -P 5964164   # Displays information about the process memory.
------------------------------------------------------------------------------
     Pid Command          Inuse      Pin     Pgsp  Virtual 64-bit Mthrd  16MB
 5964164 ksh              21167    15204        0    21086      N     N     N

     PageSize                Inuse        Pin       Pgsp    Virtual
     s    4 KB                 223          4          0        142
     m   64 KB                1309        950          0       1309
     L   16 MB                   0          0          0          0
     S   16 GB                   0          0          0          0

    Vsid      Esid Type Description              PSize  Inuse   Pin Pgsp Virtual
    8002         0 work kernel segment              m    697   639    0     697
    9000         d work shared library text         m    612   311    0     612
  850454         2 work process private             sm   113     4    0     113
  810184         1 clnt code,/dev/hd2:918           s     76     0    -       -
  87c49f         f work shared library data         sm    29     0    0      29
  8203a8         - clnt /dev/hd4:84                 s      3     0    -       -
  87c1bf         - clnt /dev/hd2:18245              s      2     0    -       -

# svmon -S 8002   # Displays information about segment 8002 (vsid column of the above command).

    Vsid      Esid Type Description              PSize  Inuse   Pin Pgsp Virtual
    8002         - work kernel segment              m    697   639    0     697
```

emstat - A command used to check the number of instructions that the processor does not directly run, but that it must emulate. This is related to the appearance of newer versions of processors, in which some instructions have been eliminated for various reasons. If you run older applications, they can run many instructions emulated by the system. This can cause a large load on the server and, as a result, a decrease in performance. This issue can be solved by recompiling the application or installing a newer version.

Examples:

```
# emstat 5 3   # Displays three statistics from the subsequent five-second periods.

  Emulation  Emulation
  SinceBoot      Delta
          2          0
          2          0
          2          0
```

fileplace - Presents the arrangement of the blocks of a given file. The efficiency of the operations on file depends on the degree to which that file is fragmented.

Examples:

```
# fileplace smit.log

File: smit.log  Size: 407310 bytes  Vol: /dev/hd4
Blk Size: 4096  Frag Size: 4096  Nfrags: 100

  Logical Extent
  --------------
  00000060-00000061              2 frags         8192 Bytes,   2.0%
  00004524                       1 frags         4096 Bytes,   1.0%
  00004952-00004988             37 frags       151552 Bytes,  37.0%
```

```
  00005032-00005055             24 frags        98304 Bytes,  24.0%
  00005146-00005147              2 frags         8192 Bytes,   2.0%
  00005760-00005793             34 frags       139264 Bytes,  34.0%
```

tprof - A program that monitors the usage of processors. It allows the testing of the processors' load with individual processes. The program collects information by running a trace in the background in order to track system operation. Based on the results it receives from the trace program, it generates a report that is saved to a file in the current directory.

Examples:

```
# tprof -x sleep 3   # It collects statistics from the three-second work of the system (sleep 3).
Wed Feb 22 06:06:59 2017
System: AIX 7.2 Node: aix72c2 Machine: 00F621774C00
Starting Command sleep 3
stopping trace collection.
Generating sleep.prof.   # The report is saved to the sleep.prof file.

# head -27 sleep.prof.   # Displays the report generated above.
Configuration information
=========================
System: AIX 7.2 Node: aix72c2 Machine: 00F621774C00
Tprof command was:
    tprof -x sleep 3
Trace command was:
    /usr/bin/trace -ad -M -L 124423372 -T 500000 -j
00A,001,002,003,38F,005,006,134,210,139,5A2,5A5,465,234,5D8, -o -
Total Samples = 782
Traced Time = 3.01s (out of a total execution time of 3.01s)
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

Process                                Freq  Total Kernel   User Shared  Other
=======                                ====  ===== ======   ==== ======  =====
wait                                      8  61.25  61.25   0.00   0.00   0.00
/usr/bin/find                             6  38.11  34.91   1.02   2.17   0.00
/usr/bin/ksh                              2   0.26   0.26   0.00   0.00   0.00
/usr/sbin/getty                           1   0.26   0.26   0.00   0.00   0.00
/usr/bin/trcstop                          1   0.13   0.13   0.00   0.00   0.00
=======                                ====  ===== ======   ==== ======  =====
Total                                    18 100.00  96.80   1.02   2.17   0.00

Process                 PID      TID  Total Kernel   User Shared  Other
=======                 ===      ===  ===== ======   ==== ======  =====
wait                1114406  1638711  23.02  23.02   0.00   0.00   0.00
wait                    260      261  15.22  15.22   0.00   0.00   0.00
/usr/bin/find       6947138 22348111   7.29   6.78   0.38   0.13   0.00
/usr/bin/find       6947140 22348113   7.29   6.65   0.13   0.51   0.00
```

tprof can also be used to verify programs written in Java, C, C++, or FORTRAN. After running as follows, *tprof ./program_name*, it will run the program specified in the parameter and then examine which code sections generate the highest load for the processor. Such information can provide a valuable hint for developers, indicating where the program should be optimized.

pprof - A program that monitors the use of processors by individual kernel threads. It collects information by running a *trace* command in the background that tracks system operation. Based on the results it receives from the *trace* program, it generates several different reports that are saved to files in the current directory.

Examples:

```
# pprof 5   # Collecting statistics for five seconds.
Wed Feb 22 06:12:56 2017
System: AIX 7.2 Node: aix72c2 Machine: 00F621774C00
Starting Command /usr/bin/sleep 5
stopping trace collection.

*** PPROF COMPLETED ***

# ls pprof*    # As a result of the pprof command, several reports are created.
pprof.cpu      pprof.famcpu   pprof.famind   pprof.flow     pprof.namecpu  pprof.start
```

Filemon - A command that allows you to test the load of individual logical volumes, physical volumes, and memory segments, as well as the load of the CPU, by input-output operations. It collects information for a given time by running a *trace* command in the background in order to track system operation. Based on the results of the *trace* command, it extracts the system activity and load information, and then it creates a detailed report.

Examples:

```
# filemon   # Starting statistics collection.

Run trcstop command to signal end of trace.
Wed Feb 22 06:24:55 2017
System: AIX 7.2 Node: aix72c2 Machine: 00F621774C00

# trcstop   # Stopping statistics collection. A detailed report is presented on the screen:
# Cpu utilization:  61.3%
Cpu allocation:   64.6%
[filemon: Reporting started]

59087140 events were lost.  Reported data may have inconsistencies or errors.


Most Active Segments
------------------------------------------------------------------------
  #MBs  #rpgs  #wpgs  segid  segtype                  volume:inode
------------------------------------------------------------------------
   0.0      0      3   8703bc  client
   0.0      0      2   8243c9  client
   0.0      0      1   8203a8  client
   0.0      0      1   840390  client

Most Active Logical Volumes
------------------------------------------------------------------------
  util  #rblk  #wblk   KB/s  volume                   description
------------------------------------------------------------------------
  0.00      0     40    0.4  /dev/hd8                 jfs2log
  0.00      0     80    0.7  /dev/hd4                 /
  0.00      0     48    0.4  /dev/hd9var              /var

Most Active Physical Volumes
------------------------------------------------------------------------
  util  #rblk  #wblk   KB/s  volume                   description
------------------------------------------------------------------------
  0.00      0    168    1.5  /dev/hdisk0              N/A

Most Active Files Process-Wise
------------------------------------------------------------------------
  #MBs  #opns   #rds   #wrs  file                     PID(Process:TID)
------------------------------------------------------------------------
```

```
   1.8      1    451      0  trcfmt                 6947316(ksh:22348033)
   0.3      1     86      0  services               6947316(ksh:22348033)
   0.1      1     29      0  mib.defs               6947316(ksh:22348033)
   0.1      1     25      0  ifconfig               6947316(ksh:22348033)
   0.1      1     17      0  route                  6947316(ksh:22348033)
   0.1      1     16      0  rpc.pcnfsd             6947316(ksh:22348033)
   0.1      1     15      0  sendmail.cf            6947316(ksh:22348033)
   0.1      1     14      0  gated.conf             6947316(ksh:22348033)
   0.1      1     14      0  init                   6947316(ksh:22348033)
   0.0      1     11      0  ping                   6947316(ksh:22348033)
   0.0      1      9      0  dhcpsd.cnf             6947316(ksh:22348033)
# All unnecessary information has been omitted.
```

netpmon - A command that allows you to check the network load. It collects information for a given time by running a *trace* command in the background in order to track system operation. Based on the results of the command, the *trace* extracts information about both the network activity and load, and then it creates a detailed report.:

Examples:

```
# netpmon   # Starting statistics collection.
Wed Feb 22 06:29:04 2017
System: AIX 7.2 Node: aix72c2 Machine: 00F621774C00


Run trcstop command to signal end of trace.

# trcstop   # Stopping statistics collection. A detailed report is presented on the screen:

# [netpmon: Reporting started]

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

Process CPU Usage Statistics:
-----------------------------
                                                    Network
Process (top 20)             PID  CPU Time   CPU %   CPU %
-----------------------------------------------------------
netpmon                  8585608   92.1412  21.231   0.000
syncd                    2425328   44.6074  10.279   0.000
swapper                        0    0.0116   0.003   0.000
getty                    8192256    0.0115   0.003   0.000
gil                      1573172    0.0069   0.002   0.002
rmcd                     8257792    0.0044   0.001   0.000
clcomd                   6619604    0.0039   0.001   0.000
topasrec                 6160856    0.0036   0.001   0.000
xmgc                      852254    0.0030   0.001   0.000
lock_rcv                 3342706    0.0026   0.001   0.000
sec_rcv                  3473774    0.0026   0.001   0.000
trcstop                  6422872    0.0023   0.001   0.000
rpc.lockd                4129084    0.0020   0.000   0.000
ping                     6422868    0.0015   0.000   0.000
ksh                      8913260    0.0013   0.000   0.000
ksh                      5964164    0.0011   0.000   0.000
sshd:                    5570840    0.0010   0.000   0.000
aso                      5046690    0.0010   0.000   0.000
sendmail:                5177762    0.0008   0.000   0.000
netpmon                  8716724    0.0008   0.000   0.000
-----------------------------------------------------------
Total (all processes)              136.8143  31.525   0.002
Idle time                          292.2010  67.330

========================================================================
```

```
First Level Interrupt Handler CPU Usage Statistics:
---------------------------------------------------
                                                  Network
FLIH                               CPU Time   CPU %   CPU %
-----------------------------------------------------------
UNKNOWN                              0.0785   0.018   0.000
PPC decrementer                      0.0269   0.006   0.000
data page fault                      0.0045   0.001   0.000
external device                      0.0005   0.000   0.000
instruction page fault               0.0001   0.000   0.000
queued interrupt                     0.0000   0.000   0.000
-----------------------------------------------------------
Total (all FLIHs)                    0.1105   0.025   0.000

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

TCP Socket Call Statistics (by Process):
----------------------------------------
                                  ------ Read -----   ----- Write -----
Process (top 20)             PID   Calls/s   Bytes/s   Calls/s   Bytes/s
--------------------------------------------------------------------------------
sshd:                    5570840      0.04       653      0.06         4
--------------------------------------------------------------------------------
Total (all processes)                 0.04       653      0.06         4

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

ICMP Socket Call Statistics (by Process):
-----------------------------------------
                                  ------ Read -----   ----- Write -----
Process (top 20)             PID   Calls/s   Bytes/s   Calls/s   Bytes/s
--------------------------------------------------------------------------------
ping                     6422868      0.00         0      0.01         1
--------------------------------------------------------------------------------
Total (all processes)                 0.00         0      0.01         1

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

Detailed TCP Socket Call Statistics (by Process):
-------------------------------------------------

PROCESS: sshd:   PID: 5570840
reads:                  4
  read sizes (bytes):   avg 16384.0 min 16384   max 16384   sdev 0.0
  read times (msec):    avg 0.009   min 0.007   max 0.010   sdev 0.001
writes:                 6
  write sizes (bytes):  avg 72.0    min 64      max 112     sdev 17.9
  write times (msec):   avg 0.031   min 0.020   max 0.050   sdev 0.009

PROTOCOL: TCP (All Processes)
reads:                  4
  read sizes (bytes):   avg 16384.0 min 16384   max 16384   sdev 0.0
  read times (msec):    avg 0.009   min 0.007   max 0.010   sdev 0.001
writes:                 6
  write sizes (bytes):  avg 72.0    min 64      max 112     sdev 17.9
  write times (msec):   avg 0.031   min 0.020   max 0.050   sdev 0.009

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

Detailed ICMP Socket Call Statistics (by Process):
--------------------------------------------------

PROCESS: ping   PID: 6422868
writes:                 1
```

```
  write sizes (bytes):  avg 64.0    min 64      max 64      sdev 0.0
  write times (msec):   avg 0.046   min 0.046   max 0.046   sdev 0.000

PROTOCOL: ICMP (All Processes)
writes:                 1
  write sizes (bytes):  avg 64.0    min 64      max 64      sdev 0.0
  write times (msec):   avg 0.046   min 0.046   max 0.046   sdev 0.000
67242753 Missed Entries found

[netpmon: Reporting completed]
[               8 traced cpus               ]
[         368.481 secs total preempt time   ]

[netpmon: 100.308 secs in measured interval]
```

nfsstat - Displays information about the NFS (*Network File System*) and RPC (*Remote Procedure Call*).

spray - Sends the specified number of packets to the selected host and then presents performance statistics for this operation. The *spray* service must be running on the host where the packets are to be sent. By default, it is turned off. It is supported by the *inetd* daemon, so the starting point is to uncomment the corresponding line in the */etc/inetd.conf* file and refresh the *inetd* daemon configuration (*refresh -s inetd*).

Examples:

```
# spray localhost -c 10000 -l 2000   # Sending 10,000 packets of size 2000 B to the localhost
address.
 sending 10000 packets of length 2000 to
        localhost ...

        2092 packets (20.920%) dropped by localhost
        44468 packets/second, 88937873 bytes/second
```

netstat, entstat - The commands are described in the chapter concerning network management.

Tuning

schedo - A command intended to manipulate the way the system manages the processes. It enables the tuning of the processors. Starting from AIX 5.2, it replaces its predecessor (i.e., *schedtune*).

Examples:

```
# schedo -L   # List of parameters, along with their settings.
NAME                      CUR    DEF    BOOT   MIN    MAX    UNIT           TYPE
     DEPENDENCIES
------------------------------------------------------------------------------
affinity_lim              7      7      7      0      100    dispatches        D
------------------------------------------------------------------------------
big_tick_size             1      1      1      1      100    10 ms             D
------------------------------------------------------------------------------
ded_cpu_donate_thresh     80     80     80     0      100    % busy            D
------------------------------------------------------------------------------
fixed_pri_global          0      0      0      0      1      boolean           D
------------------------------------------------------------------------------
force_grq                 0      0      0      0      1      boolean           D
------------------------------------------------------------------------------
```

```
maxspin                   16K    16K    16K    1      4G-1   spins            D
------------------------------------------------------------------------------
pacefork                  10     10     10     10     2G-1   clock ticks      D
------------------------------------------------------------------------------
proc_disk_stats           0      0      0      0      1      boolean          D
------------------------------------------------------------------------------
sched_D                   16     16     16     0      32                      D
------------------------------------------------------------------------------
sched_R                   16     16     16     0      32                      D
------------------------------------------------------------------------------
tb_balance_S0             2      2      2      0      2      ticks            D
------------------------------------------------------------------------------
tb_balance_S1             2      2      2      0      2      ticks            D
------------------------------------------------------------------------------
tb_threshold              100    100    100    10     1000   ticks            D
------------------------------------------------------------------------------
timeslice                 1      1      1      0      2G-1   clock ticks      D
------------------------------------------------------------------------------
vpm_fold_policy           1      1      1      0      15                      D
------------------------------------------------------------------------------
vpm_throughput_core_threshold
                          1      1      1      0      1K     cores            D
------------------------------------------------------------------------------
vpm_throughput_mode       0      0      0      0      8      level            D
------------------------------------------------------------------------------
vpm_xvcpus                0      0      0      -1     1K     processors       D
------------------------------------------------------------------------------

n/a means parameter not supported by the current platform or kernel

Parameter types:
    S = Static: cannot be changed
    D = Dynamic: can be freely changed
    B = Bosboot: can only be changed using bosboot and reboot
    R = Reboot: can only be changed during reboot
    C = Connect: changes are only effective for future socket connections
    M = Mount: changes are only effective for future mountings
    I = Incremental: can only be incremented
    d = deprecated: deprecated and cannot be changed

Value conventions:
    K = Kilo: 2^10       G = Giga: 2^30       P = Peta: 2^50
    M = Mega: 2^20       T = Tera: 2^40       E = Exa: 2^60
```

It should be noted that some parameters can be changed dynamically, while some require other actions, such as reboot, establishing a new session (network), remounting (file systems), etc. This is shown in the last column of the output (i.e., *Type*). The -L parameter and the parameter listing method are common for this and the following commands:

- ***vmo*** - A command used to tune the memory management. Starting with AIX 5.2, it replaced the *vmtune* command.
- ***ioo*** - A command used to tune the I/O parameters. Starting with AIX 5.2, it replaced the *vmtune* command.
- ***no*** - A command used to tune the network parameters. It was partially discussed in the chapter concerning network management.
- ***nfso*** - A command used to tune the NFS parameters.

The methods of operation, the displayed parameters, and their associated changes are the same for all the above commands. In order to become more familiar with them, all you need to do is read the

section on network management (network options), which describes how to use the *no* command.

Other tools

There are numerous additional tools that can be used for monitoring and tuning. Very often, IBM developers create such tools. The tools can facilitate the collection of information (for example, LPAR2rrd, HMC Scanner), or they can examine the system and suggest changes to parameters (VIOS Performance Advisor, Java Performance Advisor). You can find out more about the available tools and their descriptions by searching for the keywords "AIX Other Performance Tool" or by following the link below:
https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/Power+Systems/page/Other+Performance+Tools