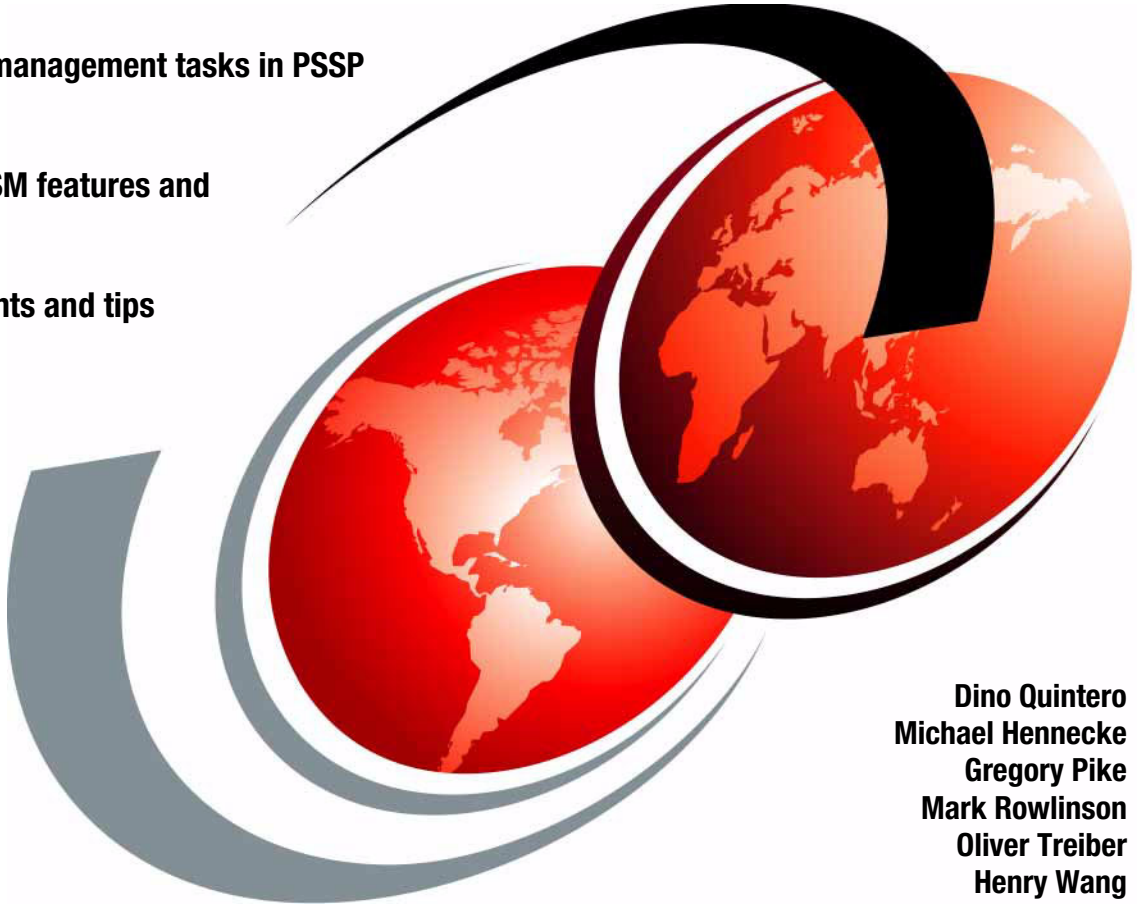
IBM

CSM Guide for the PSSP System Administrator

- Compares management tasks in PSSP and CSM
- Explores CSM features and commands
- Provides hints and tips



Dino Quintero
Michael Hennecke
Gregory Pike
Mark Rowlinson
Oliver Treiber
Henry Wang

Redbooks

ibm.com/redbooks

International Technical Support Organization

CSM Guide for the PSSP System Administrator

October 2003

SG24-6953-00

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

First Edition (October 2003)

This edition applies to Version 1, Release 3, Modification 2 of Cluster Systems Management for use with the AIX Operating System Version 5, Release 2, Modification 1.

Contents

Notices

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

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

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

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

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

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

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

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

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

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

Trademarks

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

AFS®	LoadLeveler®	RMF™
AIX 5L™	POWERparallel®	RS/6000®
AIX®	POWER4™	Tivoli®
BladeCenter™	pSeries®	VisualAge®
IntelliStation®	Redbooks™	xSeries®
IBM®	Redbooks (logo)  ™	

The following terms are trademarks of other companies:

Intel, Intel Inside (logos), MMX, and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

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

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

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

Preface

This IBM® Redbook compares and contrasts the methods, tools, and interfaces provided by PSSP and CSM to perform typical system administrator tasks. It is aimed at experienced PSSP system administrators who need to grow their knowledge about administering of PSSP clustered systems into equivalent knowledge about CSM clustered systems, regardless of whether or not they need to convert a system from PSSP to CSM system management.

This publication is a "task-oriented guide" that will help seasoned PSSP administrators to quickly figure out how to accomplish, in the new CSM environment, tasks they are already familiar with in PSSP.

The team that wrote this redbook

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

Dino Quintero is an IBM Certified Senior IT Specialist at the International Technical Support Organization, Poughkeepsie Center. He assesses, designs, and implements pSeries® and AIX® technical solutions for various customer sets, including those in clustered environments. He also writes Redbooks™ and teaches workshops.

Michael Hennecke is an IBM Certified AIX and RS/6000® SP Specialist at the Computing Center of the University of Karlsruhe, Germany. He is responsible for the system management of Karlsruhe's SP and Cluster 1600 systems. Michael has 12 years of experience with parallel computing, including eight years on the RS/6000 SP and Cluster 1600. He holds a degree in Physics from the University of Bochum (RUB), Germany.

Gregory Pike is a Research and Development Associate at Oak Ridge National Laboratory in Oak Ridge, Tennessee. He holds a degree in Computer Science from the University of Maine. He has over 15 years of experience in UNIX® system support and programming and owned a consulting company in Maine for eight years before joining ORNL.

Mark Rowlinson is a Senior IT Specialist working in the Pan EMEA Infrastructure and Technology group in IBM UK. He has 19 years of support experience, the last six years with RS/6000-SP and pSeries systems, focusing

on automation and application integration. He holds a B.Sc. degree in Mathematics and Computing from Bath University, England.

Oliver Treiber is a Systems Engineer with IBM Presales Technical Support in Munich, Germany. He has worked for IBM since 2001.

Henry Wang is an I/T Architect for IBM Global Services in Boulder, Colorado. He holds a M.Sc. degree in Computing and Information Science from the University of Guelph, Canada and a B.A. in Library and Information Science from Beijing University, China. He is an IBM Certified Advanced Technical Expert (CATE) - pSeries (AIX/PSSP/HACMP/p690) and a Red Hat Certified Engineer (RHCE). Currently he focuses on providing AIX and Linux clustering solutions and support to customers.

Team members (from left to right):

Dino Quintero (project leader), Michael Hennecke, Oliver Treiber, Mark Rowlinson, Gregory Pike, Henry Wang

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

Alfred Schwab, editor
International Technical Support Organization, Poughkeepsie Center

Margarita Hunt, Pete Bertolozzi, Cheryl Gera
International Technical Support Organization, Poughkeepsie Center

Jennifer Cranfill, Norm Nott, Skip Russell, Mike Stancampiano, Mike Schmidt, Janet Ellsworth, Brian Croswell, Les Vigotty, Kurt Sulzback, Chris Derobertis, Eric Agar, Serban Maerean, Ning-Wu Wang, John Simpson, Aruna Ramanan, Marty Fullam, Joseph Chaky, Paul Swiatocha Jr., Keshav Ranganathan, Brian Herr, Kwan Cheung, Linda Mellor, Mark Gurevich, Robert Curran, Susan Yang, William LePera, Geoffrey Lubold
IBM Poughkeepsie

Margaret Momberger
IBM Watson Research Center

Thomas Braunbeck
IBM Germany

Marc Genty
NCAR, Boulder Colorado

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

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

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

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

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

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

  ibm.com/redbooks

- Send your comments in an Internet note to:

  redbook@us.ibm.com

- Mail your comments to:

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

1

Introduction

This chapter contains a high-level overview of the history of Cluster Systems Management (CSM), and the features and functions of CSM.

This chapter discusses:

1.1 History and roots of CSM for AIX

Although CSM is a relatively new product in AIX 5L™, it borrows from a long line of clustering technologies. IBM has been a leader in multiprocessing since early multi-processor mainframe systems, then with clustering technology in the RS/6000 Scalable POWERparallel® systems (RS/6000 SP).

Using much of the technology and experience developed for the RS/6000 SP, such as Reliable Scalable Cluster Technology (RSCT) and combining it with industry tested and accepted “open source” software, IBM has created a unique cluster management package that addresses the needs of managing a cluster.

The clustering product from IBM used in the RS/6000 SP systems and now in the Cluster 1600 to this point has been PSSP. PSSP is a robust and well-respected product in the High Performance Computing (HPC) community. It is currently at Version 3.5.

Technology moves very fast these days, and PSSP releases to support new hardware and software have always been behind. PSSP is built separately from AIX, yet needs to interface tightly with it to assure compatibility and reliability. In addition, PSSP is integral to the functioning of the high-speed interconnect switches. So, as new switch technologies are developed, a new PSSP release needs to be developed and tested along with it.

CSM for AIX has been developed taking into consideration the previous PSSP challenges, and addresses them in the following ways:

- CSM for AIX is now part of AIX development and is included on the media shipped with AIX 5L Version 5.2. RSCT and CSM have been shipped as part of AIX since Version 5.1.
- CSM has been developed independent of the switch technology, allowing new switch technologies to be developed separately from the cluster software.
- New switch device drivers will be provided by AIX, whereas they used to be provided by PSSP.

Note: Since the *current* SP Switch technology is dependent on PSSP, it is not supported by CSM for AIX.

1.2 CSM 1.3 for AIX 5L

Cluster Systems Management (CSM) Version 1.3 for AIX 5L is an IBM product package that contains a group of tools and utilities developed by IBM and by the

open source software community, combined in a way to provide management functions for pSeries servers with AIX 5L and xSeries® servers with Linux clusters in the same Cluster 1600.

A cluster is more than just physically connected systems (nodes) on a network. The point of a cluster is to be able to manage multiple systems. In a small network of only a few systems, this can be done by simply logging in and administering each one, but this is not practical with a large number of them. A cluster with 128 systems, or more (special bid required), is not uncommon.

Note: Unlike PSSP, where there is a notion of frames and nodes that make up a cluster, CSM has no concept of frames and nodes and treats each instance of an operating system as a node regardless of its physical location.

Therefore, the need to manage the clusters in a simpler and consistent manner has evolved. This is where a management server comes into play. Some of the tasks that need to be done in a cluster from a central point of control are:

- Install and update software on all machines in the cluster
- Detect and diagnose problems that can lead to system errors
- Handle problems that occur when no one is around
- Control or reboot servers remotely
- Keep administrative and system costs down

CSM for AIX handles these challenges and provides additional functions to efficiently manage a cluster. Here are some of the functions:

- Centralized node management from a single point of control: AIX and Linux nodes.
- Simplifies the tasks of installing, operating, and maintaining clusters—enabling clusters to scale up easily.
- Modular architecture, providing a comprehensive cluster management solution with the flexibility to choose among needed functions
- Highly reliable infrastructure and event monitoring capabilities
- Script-based, which can be easily extended
- Heterogeneous clusters. The heterogeneous CSM Cluster 1600 with an AIX management server can contain:
  - AIX 5L Version 5.2 nodes on pSeries and RS/6000 servers
  - AIX 5L Version 5.1 nodes with Recommended Maintenance package 5100-03 or later
  - Red Hat Linux 7.2 and 7.3 on xSeries nodes

1.3 Our test environment

Our test environment contained three different Cluster 1600s managed by CSM. The first Cluster 1600 contained a traditional frame with legacy RS/6000 SP supported nodes managed by CSM, as shown in Figure 1-1.

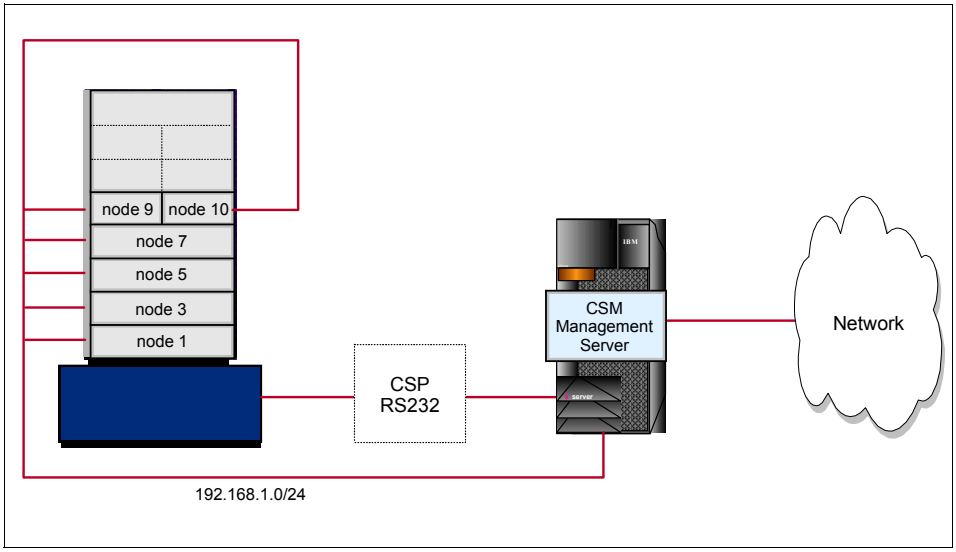


Figure 1-1 Frame with nodes managed by CSM

Our second Cluster 1600 managed by CSM contained a p690 system with two LPARs, as shown in Figure 1-2.

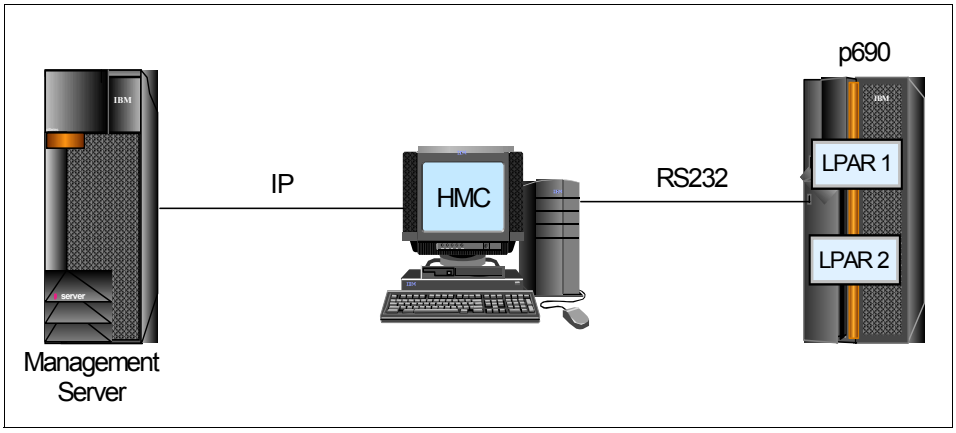


Figure 1-2 p690 with two LPARs in a Cluster 1600 managed by CSM

And our third Cluster 1600 managed by CSM contained a p630 system with two LPARs, as shown in Figure 1-3.

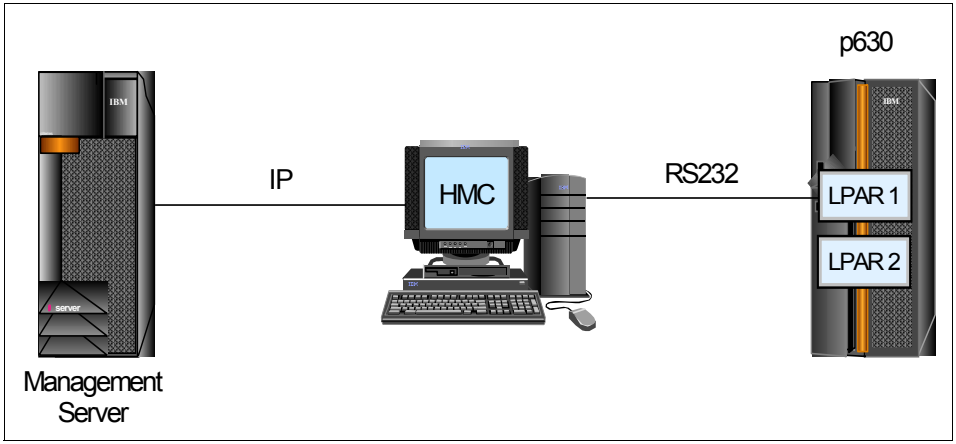


Figure 1-3 A p630 with two LPARs in a Cluster 1600 managed by CSM

1.4 Topics not covered

We want to mention those topics that are *not* dealt with in this redbook. This is important for several reasons. It explains what our main focus is, and it also avoids misconceptions on the part of the reader about the subject of this redbook. We give references to other materials that cover some of the topics that we do not address.

- We do not teach PSSP.

  This is a redbook to guide the PSSP system administrator in a CSM environment, so we assume that you have a solid PSSP background. If you are familiar with PSSP but need an update about the new features in the latest PSSP release, refer to *IBM (e)server Cluster 1600 Managed by PSSP 3.5: What's New,* SG24-6617. A more general coverage of PSSP in a cluster can be found in *RS/6000 SP Cluster: The Path to Universal Clustering,* SG24-5374.

- We do not teach CSM.

  While we do not require an in-depth knowledge of CSM, we refer to the existing documentation rather than duplicating it. The definitive references to CSM are the CSM product manuals. In addition, several redbooks exist that focus on CSM, for example *An Introduction to CSM 1.3 for AIX 5L,* SG24-6859 and *An Introduction to Security in a CSM 1.3 for AIX 5L Environment,* SG24-6873.

- This redbook is not an RSCT and RMC primer.

  The RSCT infrastructure is an important prerequisite for CSM, and the CSM system administrator has to be familiar with its architecture and functionality. For reference, see the RSCT product manuals. The redbook *A Practical Guide for Resource Monitoring and Control (RMC),* SG24-6615 is also a useful resource. In this redbook, we assume a general understanding of RSCT and focus on those RSCT features that are specific to a CSM cluster (a *management domain* in RSCT jargon). In particular, we explain how the RSCT infrastructure is exploited by CSM and other cluster software components such as VSD. We also discuss RSCT security with respect to a management cluster.

- We do not cover the transition of a cluster from PSSP to CSM.

  This is an important question, and it came up many times during the residency. However, it is important to understand that this redbook is about "transitioning" a PSSP *administrator* to CSM, not a PSSP *cluster*. At the time of this writing, IBM is working on the transitioning of a cluster from PSSP to CSM, and CSM transition support is intended to be delivered in a future release of CSM as well as in upcoming redbook projects.

- We do not touch CSM for Linux.

  The possibility to manage both AIX 5L and Linux nodes within one cluster is one of the strengths of CSM. But since this book is written for system administrators with an AIX and PSSP background, we prefer to keep our focus on CSM for AIX.

- The High Performance Computing (HPC) stack is not discussed.

  In addition to the main cluster management functions, customers in the HPC market rely on a number of hardware and software components that are collectively referred to as the *HPC stack*. These include a high-performance switch interconnect and the software components GPFS, LoadLeveler®, Parallel Environment (PE), and PESSL. Support for these products in a CSM cluster is planned but not yet available.

1.5 Additional references

For additional information and reference materials, refer to:

- *IBM (e)server Cluster 1600 Managed by PSSP 3.5: What's New,* SG24-6617
- *An Introduction to CSM 1.3 for AIX 5L,* SG24-6859
- *An Introduction to Security in a CSM 1.3 for AIX 5L Environment,* SG24-6873
- *A Practical Guide for Resource Monitoring and Control (RMC),* SG24-6615

- *RSCT for AIX 5L: Guide and Reference,* SA22-7889
- *RSCT for AIX 5L: Messages,* GA22-7891
- *CSM for AIX 5L: Command and Technical Reference,* SA22-7934
- *CSM for AIX 5L: Hardware Control Guide,* SA22-7920
- *CSM for AIX 5L: Software Planning and Installation Guide,* SA22-7919
- *CSM for AIX 5L: Administration Guide,* SA22-7918

The remaining chapters of this book will go into detail about planning, installation, PSSP SRD versus RSCT SR, administration, monitoring, security and access control for CSM 1.3.1 for AIX 5L.

2

Planning

This chapter describes the differences between planning a PSSP cluster and planning a CSM cluster. It is intended for technical professionals who are experienced in planning the installation of a Cluster 1600 system managed by PSSP. It provides information on planning a new CSM cluster and may also help if you are thinking about converting an existing PSSP cluster to a CSM cluster, and would like to know if your existing hardware and software are supported by CSM.

Clustering has become more popular in recent years thanks to its rich functionality and ease of management. It has grown more complex, however, due to the growing numbers and types of hardware and software that make up the clusters. Good planning is essential to making sure the clustered system has all the necessary hardware and software, all the hardware and software are compatible, and the cluster is configured in the most efficient manner to best suit your particular needs.

This chapter covers the following more topics:

- 2.7, “Cluster-related licensed programs” on page 30

The following planning books are available for PSSP:

- *RS/6000 SP: Planning, Volume 1, Hardware and Physical Environment,* GA22-7280
- *RS/6000 SP Planning, Volume 2: Control Workstation and Software Environment,* GA22-7281
- *IBM (e)server Cluster 1600: Planning, Installation, and Service,* GA22-7863

Planning information for CSM can be found in:

- *CSM for AIX 5L: Software Planning and Installation Guide,* SA22-7919
- *CSM for AIX 5L: Hardware Control Guide,* SA22-7920

This chapter is not a replacement for any of these books. We focus on the differences between planning for a PSSP cluster and planning for a CSM cluster. We base our comparison on PSSP V3.5 and CSM V1.3.1, which are the latest versions of the respective software at the time of the development of this redbook. For the latest information on PSSP and CSM, refer to:

- The latest PSSP “Read This First” document at:

  `http://www.ibm.com/servers/eserver/pseries/library/sp_books/pssp.html`
- The latest CSM “Read This First” document at:

  `http://www.ibm.com/servers/eserver/pseries/library/clusters/aix.html`

2.1 Differences between PSSP and CSM

PSSP is an integrated suite of software finely tuned to manage an SP system as a full-function parallel processing system. Comprehensive planning is necessary for building an SP cluster. There are over twenty “SP system planning worksheets” found in Appendix C of *RS/6000 SP Planning, Volume 2: Control Workstation and Software Environment,* GA22-7281.

CSM has a modular design and can be used on a wide variety of hardware and software platforms. Many modules are optional and can be added later, allowing for a much quicker start.

Keep in mind that your choice of hardware can lead to requirements on software, and your choice of software can lead to requirements on hardware. So as you make modifications to one part of your earlier plan, other parts may have to be adjusted as well.

2.1.1 PSSP transition

PSSP and CSM cannot coexist in the same cluster. You must remove PSSP from a host before installing CSM. Most of the hardware that runs PSSP can also run CSM, and CSM provides support for additional hardware as well. The following sections provide more details.

2.1.2 Hardware requirements

CSM offers distributed management services for any machine running AIX 5L, and offers remote hardware control. The network operating system installs to certain types of hardware only. Remote hardware control means remote power (query status, power on, power off) and remote console connection.

The CSM single-point-of-control server is known as the management server and needs to be a pSeries or RS/6000 machine running AIX V5.2. The other machines in the cluster are referred to as managed nodes and can be pSeries, or RS/6000 machines, or LPARs running AIX V5.2 with the recommended maintenance package 5200-01 or later; or AIX 5L V5.1 with the recommended maintenance package 5100-03 or later; or xSeries™ machines running CSM for Linux V1.3.1.

The following additional hardware is required for CSM 1.3.1 for AIX 5L:

- On the CSM management server, one Ethernet adapter for each management virtual LAN (VLAN) used for hardware control, and one Ethernet adapter for each cluster VLAN used for CSM installation and administration.

- On the management server, a minimum of 1024 MB of memory and 120 MB of disk space to install CSM.
- On the management server and on each node, additional disk space for the base AIX 5L operating system and filesets.
- On each managed node, one Ethernet adapter for the cluster VLAN used for CSM installation and administration.
- On each managed node, a minimum of 128 MB of memory and 20 MB of disk space to install and run CSM.
- On the NIM resource server, 2 GB of disk space for each version of AIX 5L installation images.

2.1.3 Software requirements

PSSP V3.5 supports AIX 5L V5.1 and V5.2. An IBM C or C++ compiler is required for the PSSP software to use. The compiler is necessary for service of the PSSP software. Also, without the compiler preprocessor, dump diagnosis tools like kdb will not work effectively. You need at least one concurrent user license for the C or C++ compilers of VisualAge® C++ Professional for AIX, Version 6.0 or later.

The CSM management server must be running AIX 5L V5.2 with recommended maintenance package 5200-01 or later (APAR IY39795). The other machines in the cluster are referred to as managed nodes and can be running AIX 5L V5.2 with recommended maintenance package 5200-01 or later, or AIX 5L V5.1 with recommended maintenance package 5100-03 or later (APAR IY39794). A management server running AIX can also manage xSeries nodes running supported Linux versions.

For all AIX servers, the following CSM for AIX 5L 1.3.1 service is required:

```
Description                                              APAR number
-------------------------------------------------------------------
Added Service for RSCT on AIX 5.1                        IY42782
Added Service for RSCT on AIX 5.2                        IY42783
CSM for AIX 5L added service                             IY42353

Depending on your hardware, you may also need one or more of the following
APARs to provide CSM support for that hardware:

CSM Support for p670 & p690 POWER4+                      IY42356
CSM Support for 7026 servers, 9076 SP nodes & p660       IY42379
CSM Support of p655 Power4+                              IY42377
```

Note: The support for 9076 SP Node Features 2054/2058 IY42847, and the SP Expansion I/O unit support is shipped in CSM 1.3.2.

These updates must be applied before setting up a CSM management server or running any CSM commands.

Tip: If HMC-controlled servers are included in the cluster, the HMC firmware needs to be at Release 3 Version 1.0 or later, and Java131.rte and Java131.ext.xml4 need to be installed on the management server.

2.1.4 License

PSSP has no requirement for a license key. CSM requires a license key. The full CSM license key is shipped on the CSM product CD-ROM. (The CSM software itself resides on the AIX 5L media. The CSM media contains only the license key.)

The license key file must be copied from the CSM medium CD to a temporary directory. The name of the key file contained on the CD is csmlum.full. Your temporary file must be used as input to the **csmconfig** command to accept, install, and activate the CSM license during the CSM installation process.

A 60 day Try and Buy license is available; see 2.5.1, “Try and Buy license” on page 25 for more information.

2.2 Control workstation and boot-install server

Running the PSSP software requires an IBM @server pSeries or RS/6000 workstation as a point of control for managing, monitoring, and maintaining the cluster. This point of control is referred to as the Control Workstation (CWS). In a CSM cluster, similar function is provided by a management server (MS).

Like the CWS, the management server connects directly to each SP frame and attached server if they are included in the cluster. The management server can manage AIX and Linux servers in geographically dispersed data centers if no SP or attached servers exist in the cluster.

Attention: Note that an IBM pSeries p660 is considered an attached server to a Cluster 1600 managed by PSSP. However, in a Cluster 1600 managed by CSM, the p660 is considered another node in the cluster.

Under PSSP, by default, the CWS is a boot-install server and additional servers can be defined as boot-install servers as well. Boot-install function has been replaced by the NIM resource server under CSM and the management server does not have to be a NIM resource server. In addition, because network

performance has significantly improved since the early years of PSSP, it is no longer required to use more than one boot-install server (or NIM resource server), even for large systems.

2.2.1 Management server

Any server running AIX 5L V5.2 can be a management server if it has enough capacity, for example, the PCI models listed in Table 33, “Supported control workstations” in *RS/6000 SP Planning, Volume 2: Control Workstation and Software Environment,* GA22-7281. Common Hardware Reference Platform (CHRP) is only supported by AIX 5L V5.2.

CSM supports the use of an LPAR as a management server. However, some limitations need to be taken into consideration if you choose to use an LPAR as a management server:

1. The CSM management server can be brought down inadvertently by someone on the HMC deactivating that LPAR. Even if someone does not have access to the CSM management server, if they have access to the HMC, they can power off the management server. It can also be affected by someone moving resources such as CPU or I/O from that LPAR.
2. If the firmware needs to be upgraded, the LPAR management server may go down along with the rest of the CEC in order to upgrade the firmware. However, upon bringing the CEC back up, the system will return to normal.
3. There is no direct manual hardware control of the CSM management server. The administrator must go through the HMC for power control of the management server.
4. In many cases a physically separate CSM management server can be safer, because an LPAR management server belongs to a CEC that could be down from a hardware or power failure, thus losing access to your management server is more likely if an LPAR is used.
5. It is not possible to start a local CSM GUI because the management server has no graphical display.
6. An LPAR management server may not contain media devices such as CD, tape, or diskette drives. This can affect your back-up strategy. In machines such as the p690, you may be able to assign a CD-ROM drive to the management server LPAR.
7. It is recommended that a management server should not also be defined as a managed node of the cluster it manages.
8. An LPAR management server cannot connect SP nodes or p660 servers to a management server running on an LPAR because the LPAR lacks the serial tty ports necessary for such a connection.

A managed node can be a management server for another cluster, thus allowing a hierarchy cluster structure. This alternative helps avoid building a single big cluster, and improves performance since each management server only has to talk to a much smaller number of managed nodes.

2.2.2 NIM resource server

The PSSP CD-ROM includes minimal system images for the node installation, CSM does not. In PSSP, the CWS is always a boot-install server; additional boot-install servers can be defined to prevent concurrent network installation from exceeding the capacity of the CWS and/or the SP administrative LAN. Some limitations, however, exist:

- The NIM clients that are served by boot-install servers must be on the same subnet as the boot-install server's Ethernet adapter.
- NIM clients must have a route to the control workstation over the SP Ethernet.
- The control workstation must have a route to the NIM clients over the SP Ethernet.
- Every boot/install server node must have the control workstation as its boot/install server.

CSM does not provide a minimum installation image. You need to create your own *mksysb* image if that is your installation method. Note that with CSM, other methods, such as runtime (*rte)* installation, can also be used.

You can set up the management server as your NIM resource server or you can use another NIM server or servers to install your cluster nodes. Unlike PSSP, CSM does not manage your NIM installation environment. You need to set up your own NIM environment but you are no longer limited to what NIM functions PSSP provides. You can take full advantage of all the features that NIM has to offer. Appendix C, "NIM examples" on page 209 provides a sample NIM configuration.

2.2.3 HACWS

High Availability Control Workstation (HACWS) provides an option for a PSSP cluster to have a redundant CWS that effectively eliminates the CWS as a single point of failure. This is important since many cluster functions are dependent on the CWS availability. For example, a processor node will query the SDR during reboot, and if the CWS is not available, then the reboot will hang.

CSM has removed many of the dependencies on the management server, thus the management server is no longer critical for the operation of the cluster. However, tasks including monitoring, automated responses for detecting

problems in the cluster, and scheduled file and software updates will not occur while the management server is down.

Note: IBM intends to provide a high availability option for the management server in the future.

2.3 SP frames and nodes

PSSP is based on SP hardware, frames, and nodes, which are the building blocks for a PSSP cluster. The building blocks for CSM are the clustered nodes.

2.3.1 The SP model of frames and nodes

The challenge to provide a dense packaging of individual nodes was addressed by the familiar SP frame concept: The SP frame contains 16 slots that can house nodes, a 17th slot in the bottom to house an (optional) SP Switch for high-speed interconnections between nodes, power supplies for all these components, and a system of frame and node supervisors for hardware control.

Note: In this section, we use the term “SP Switch” to denote any of the high-speed interconnects High Performance Switch (HPS), SP Switch, and SP Switch2. For the purpose of this section, there is no need to differentiate between them because they have the same size and connectivity (16 node connections, 16 connections to other switches).

Here is a short history of how this original model evolved:

- The only node type that was available for the original SP was a *thin* node (based on the model 390 workstation). So each of the 16 node slots in the SP frame could be occupied by one node, numbered 1 to 16 like the slots they occupied. Note that the (optional) SP Switch also has 16 node connections to accommodate the 16 thin nodes per frame.
- As new workstation models were released, corresponding SP nodes were built—often with a time lag because of the required repackaging to fit them into the SP frames’ slots.
- The demand for more I/O capabilities resulted in *wide* nodes, which occupied two adjacent slots and left “holes” in the node (and switch) numbering.
- With the introduction of SMP nodes, *high* nodes were built. They occupied four node slots and left even bigger holes in the node (and switch) numbering schemes.

Note that for switched systems, the bigger nodes caused some of the node connections of the switch to be unused. To overcome this costly problem, the concept of non-switched *expansion frames* was introduced: For example, the 8 unused switch ports in a wide node frame with an SP Switch could be connected to the 8 wide nodes in a second frame without a switch. This concept makes it possible to reuse the switch ports, but introduces complicated node placement rules as well as power-on/off sequencing issues between frames.

- Enterprise servers like the S70 were so large that it made little sense to repackage them to fit into the SP frames' slots. To be able to include them in an SP, the concept of *SP-attached servers* was introduced: Because the whole SP design was based on node numbers, an SP-attached server still had to logically occupy a slot and node number in an SP frame (and the corresponding SP Switch number), but physically it was an external server. Since those servers had their own power and service processor, they also had a "frame" personality in addition to a "node" personality.
- Different SP-attached servers employ various service processors that differ from the SP's supervisor cards, requiring PSSP *hardmon* support for a number of different hardware protocols, including the original hardmon protocol, SAMI, and CSP.
- The situation became interesting when requirements arose for clustering of enterprise servers without any SP nodes. Due to limitations in PSSP, this was not possible initially: An SP frame with a node in slot 1 was needed as a minimum, and only on top of this could external servers be attached. This limitation was eventually removed, allowing true *Clustered Enterprise Servers* without any SP nodes.

In principle, the introduction of large POWER4™-based servers such as the p690 was similar to the SP attachment of enterprise servers like the S70. But the POWER4-based systems differ from the S70 in two important aspects:

- The introduction of a hardware management console (HMC) separates hardware control from the control workstation (CWS). For the PSSP world this means that the hardmon daemon has to use yet another protocol to communicate with the HMC, but on the other hand the separation of hardware and software functions allows for a better modularity.
- The possibility to create logical partitions (LPARs) gives rise to a *variable* number of "nodes" for a given piece of HW. This is not easily accommodated by the SP's rigid frame and node model. It can be handled through PSSP, but the limitations of the underlying model of "16 nodes and a switch per frame" become more and more apparent in an LPAR (and eventually dynamic LPAR) environment.

In summary, the PSSP model has evolved substantially over the last decade to be able to accommodate the advancing node hardware. While this has enabled a

seamless integration of SP nodes and external servers, it has also made the limitations of the original SP model more and more obvious and has led to complicated configuration rules.

Note: Chapter 9 of *RS/6000 SP Cluster: The Path to Universal Clustering,* SG24-5374 contains a detailed discussion of all the configuration rules in PSSP with respect to frame numbers, node numbers, and switch node numbers.

One of the technical implications of the change of terminology from the RS/6000 SP to Cluster 1600 is that it indicates the end of special “SP node” packaging and the SP model of frames and nodes in general. It addresses several limitations of the “SP” model:

- Specialized SP node hardware, requiring a repackaging effort and so causing a time lag between the availability of a standalone machine and the corresponding SP node.
- Larger and larger servers made the original concept of a frame with many nodes less useful. Today, the LPAR in a server logically serves as a node.

In PSSP, however, the system management software architecture still has the concept of frames and nodes built into it. With CSM, the change to the more general “cluster” model is also made on the software level.

2.3.2 Frame

For pSeries systems, the hardware control functionality of the SP’s frame supervisor is replaced by the HMC. For CSM support of legacy SP systems, there is still a physical serial connection from the management server to each SP frame, but each frame is merely referenced by a *tty* device name and is controlled by a single cspd daemon that runs on the management server.

2.3.3 Node numbering and naming

SP processor nodes have a node number corresponding to their slot location in the SP frame. CSM does not use node numbering.

Under PSSP, you can specify a node range by using node numbers, for example, “3-5” refers to node 3, node 4 and node 5 in frame 1.

The node range has been enhanced under CSM V1.3.1. You can use node names and node groups in your range specification, for example, “node3-5” refers to managed nodes named node3, node4 and node5. You can also define an environment variable CSM_NODE_PREFIX=node. Then, when you specify a

range that is only integers, the prefix will be auto-prefixed to the integer range so that “3-5” also refers to the managed nodes named node3, node4, and node5.

Tip: Plan your node naming conversion to take advantage of the node range feature.

For more information, see the man pages for the **noderange** command or refer to Section 5.1, “Node groupings” on page 82.

2.3.4 Managed nodes

There are different types of servers as far as their hardware control protocol is concerned. PSSP supports SP, SAMI, CSP, and HMC managed servers.

A hardware control point is the specific piece of hardware through which the management server controls node hardware. For example, for pSeries, the hardware control point is the HMC.

For hardware control, CSM 1.3.0 supports HMC-managed servers. CSM 1.3.1 and CSM 1.3.2 add additional support for SP and CSP servers, as shown in Table 2-1.

Table 2-1 Hardware control protocols supported by CSM

Hardware Control Protocol	Hardware supported by the protocol	Machine types	Available date in CSM
SP	SP frame/node supervisor	FC 2050, 2051, 2052, 2053, 2056, 2057	July 25, 2003
		FC 2054, 2058	Sept. 26, 2003
SAMI	Service and Manufacturing Interface	S70, S7A, S80	Not supported
CSP	Servers with Common Service Processor (CSP) connected by one RS-232 line to the management server	H80, M80, p660	July 25, 2003
HMC	Servers with a Hardware Management Console (HMC) use a trusted network connection to the management server	POWER4 (p670, p690)	Oct 25, 2002
		p615, p630, p655	May 23, 2003
		POWER4+ (p650, p690, p670, p655)	July 25, 2003
xseries	xSeries using IBM Remote Supervisor Adapter	xSeries (power only)	May 23, 2003

Hardware Control Protocol	Hardware supported by the protocol	Machine types	Available date in CSM
APC	APC MasterSwitch	IntelliStations (power only)	
MM	Management Module	BladeCenter™ (power only)	
SPM	Serial Port Module	BladeCenter (console only)	

PSSP V3.5 in the May 2003 update adds support for automatic failover of redundant HMCs; earlier offerings of PSSP only supported manual failover. CSM supports automatic failover of redundant HMCs.

CSM also supports some models of xSeries, IntelliStation®, and BladeCenter servers running supported versions of Linux through supported hardware control points.

Extension nodes are non-processor nodes that extend the capabilities of the SP system, but cannot be used in the same ways as SP processor nodes. A specific type of extension node is a dependent node. A dependent node depends on SP processor nodes for certain functions, for example the SP Switch router. Extension nodes are not supported under CSM.

Attention: SAMI servers are not supported in a cluster with CSM. IBM has no plans to support these servers with CSM in the future.

An SP Expansion I/O Unit is designed to satisfy the needs of customers running applications with a greater demand for internal DASD, external DASD, and network connectivity than is available in the node alone. The unit expands the capacity of a node by providing eight PCI slots and up to four hard disks. IBM intents to support SP expansion nodes in the second half of 2003.

AIX nodes, within a PSSP system, that have AIX 5L installed can have CSM software added (after removing PSSP and upgrading to the proper AIX supported level) and then added into a CSM cluster. You need to make sure all software needed on the nodes is supported by CSM.

2.3.5 Remote power

The **rpower** command communicates with hardware control points to request node power status, reboot, and power on and off functions. In addition to supporting HMC, SP and CSP, CSM supports RSA and APC for xSeries servers.

2.3.6 Remote console

The **rconsole** command, which utilizes **conserver**, communicates with console server hardware to open a console for a node on the CSM management server. Console servers must be on the management VLAN, which connects the management server to the cluster hardware control points. The hardware control points connect to node serial ports.

In addition to HMC, SP and CSP, many other console servers have been enabled to work with CSM. For example:

- MRV IR-8020, IR-8040, LX-4008S, LX-4016S, and LX-4032S
- Equinox ESP-8, ESP-16, and ELS 16 II
- Computone IntelliServer RCM4, RCM8, and RCM24
- Avocent CPS1600

These console servers are typically used with xSeries nodes, and are not supported with AIX nodes.

2.3.7 Node definition

With PSSP, hardware needs to be properly connected before a frame can be defined to the system and nodes auto-discovered.

In CSM, you can define the nodes to the cluster without hardware being present. Since CSM supports many different types of hardware, the hardware type and hardware control point for each node have to be defined to support hardware control function. However, since hardware control is optional, you do not have to define them to include them in your cluster. You can also define the hardware control information later.

Information used to define nodes is referred to as node attributes. Table 2-2 contains the general node attributes for all nodes.

Table 2-2 General node attributes

Attribute	Sample value
Hostname	node3

Attribute	Sample value
ManagementServer	sp6cws-en0
UserComment	third node, comment is optional

Table 2-3 contains the node attributes for remote power.

Table 2-3 Remote power attributes

Attribute	Sample value
PowerMethod	csp
HWControlPoint	/dev/tty1
HWControlNodeID	3

Table 2-4 contains the node attributes for remote console. Recall that the requirements for remote power are separate and distinct from the requirements for remote console.

Table 2-4 Remote console attributes

Attribute	Sample value
ConsoleMethod	csp
ConsoleServerName	/dev/tty1
ConsolePortNum	3
ConsoleSerialDevice	n/a
ConsoleServerNumber	n/a

Table 2-5 shows some of the installation attributes you may use. Appendix B of *CSM for AIX 5L: Software Planning and Installation Guide,* SA22-7919 contains a sample node attribute template. The complete list of node attributes and descriptions are documented in the *nodeattributes* man page.

Table 2-5 Node installation attributes

Attribute	Sample value
InstallAdapterDuplex	half
InstallAdapterMacaddr	0004AC49BA46
InstallAdapterSpeed	10
InstallAdapterType	ent

Attribute	Sample value
InstallCSMVersion	1.3.1
InstallOSName	AIX
InstallDistributionVersion	5.2.0

An alternative to creating a hard copy template for each node would be to create a node definition stanza file. This is a file that contains information for each node, and you can pass this file directly to the **definenode** command during the CSM installation process. You can find a sample node definition file at /opt/csm/install/nodedef.sample.

2.4 Networking

SP systems have several communication requirements, including the following:

- All SP systems require an SP Ethernet LAN for system administration.
- Switch-configured systems require a switch cable network.
- SP systems connected to external networks (or with networks between SP system partitions) require additional communication adapters.

Note: PSSP supports SP system partitioning with or without the SP Switch:

- SP system partitioning is supported by default on SP systems without a switch and on Cluster 1600 systems managed by PSSP with the SP Switch.
- SP system partitioning on a switchless SP system can be disabled.
- SP system partitioning is not supported on clustered server systems without a switch or on Cluster 1600 systems managed by PSSP with the SP Switch2.

The SP switches and system partitioning are not supported in CSM.

CSM does not support the SP Switch, the SP Switch routers, or the SP Switch2. CSM needs an administrative VLAN for system administration, a management VLAN for hardware control, and a public VLAN for applications. However, if your administrative VLAN is a trusted network, it can be used as the management VLAN.

2.4.1 Administrative VLAN

The SP Ethernet is the administrative LAN that connects all nodes in one system running PSSP to the control workstation. It is used for PSSP installation and communication among the control workstation, boot-install servers, and other nodes in the network. The CSM administrative VLAN, likewise, connects all the managed nodes to the management server, and is used for installation and communication.

2.4.2 Management VLAN

The management VLAN is for management server connection to hardware control points such as the HMC (hardware management control). It needs additional security planning and configuration to protect password data transferred from the management server to the HMC. If your administrative VLAN is a trusted network, it can be used as the management VLAN.

2.4.3 Public VLAN

The public VLAN connects the cluster nodes and management server to the site public network. Applications are accessed and run on cluster nodes over the public VLAN. The public VLAN can be connected to nodes through additional adapters.

PSSP can support additional adapters other than the SP Ethernet. Additional Ethernet, FDDI, and token-ring adapters are automatically configured on each node. CSM can support additional adapters through NIM configuration. Refer to 3.4, “Customization of managed nodes” on page 43, which summarizes the supported adapters in CSM 1.3.2.

2.5 CSM license and components

Installing IBM Cluster Systems Management involves installing both IBM CSM software and prerequisite non-IBM software. Before installing CSM, you will have to gather together all the necessary software. This includes CSM, open source prerequisites, software updates, and, optionally, software provided on the Linux distribution CD-ROM. Most of the required software will be available from AIX, CSM, or the Linux product media, but you will need to download additional software from the Internet. For more information on the CSM licensing components, open source prerequisites, and software updates, refer to *CSM for AIX 5L: Software Planning and Installation Guide,* SA22-7919.

2.5.1 Try and Buy license

You can experience the power and value of CSM and clustering at no charge for 60 days under the “Try and Buy” option. You can use all functions of CSM for 60 days, after which CSM will no longer function. Whenever you purchase and receive a full production license and key for CSM and accept its License Agreement, you may then use CSM for production purposes. CSM configuration data stored during Try and Buy will remain available for later production use.

Simply run **/opt/csm/bin/csmconfig -L** and accept the 60-day IBM License Agreement, which will be presented for your acceptance. For more information, refer to 3.2.5, “Configure the management server” on page 37.

2.5.2 CSM LPP filesets

CSM filesets for AIX 5.2 are listed in Table 2-6 on page 25. The filesets csm.core, csm.client, csm.dsh, and csm.msg are located on CD-ROM 1 and are automatically installed with AIX. Filesets csm.server, csm.diagnostics, csm.gui.dcem, and csm.gui.websm are located on CD-ROM 2 and are not automatically installed. csm.server is required on the management server; the other CSM filesets on CD-ROM 2 are optional.

On managed nodes running AIX 5.1, filesets csm.client 1.1.1 and csm.core 1.1.1 are automatically installed with AIX and fileset csm.diagnostics can be optionally installed.

Table 2-6 CSM filesets for AIX V 5.2

Filesets	Level	Description
csm.client	1.3.1.0	Cluster Systems Management Client
csm.core	1.3.1.0	Cluster Systems Management Core
csm.diagnostics	1.3.1.0	Cluster Systems Management Probe Manager / Diagnostics
csm.dsh	1.3.1.0	Cluster Systems Management dsh
csm.gui.dcem	1.3.1.0	Distributed Command Execution Manager Runtime Environment
csm.gui.websm	1.3.1.0	CSM Graphical User Interface
csm.msg.EN_US.core	1.3.0.0	CSM Core Func Msgs - U.S. English (UTF)
csm.msg.en_US.core	1.3.0.0	CSM Core Func Msgs - U.S. English
csm.server	1.3.1.0	Cluster Systems Management Server

CSM also requires that some RSCT filesets be installed. Under AIX 5.2, the RSCT filesets should be at version 2.3.1 or later, and under AIX 5.1, the RSCT filesets should be at version 2.2.1 or later:

- Parts of RSCT installed by AIX 5L

  rsct.core.auditrm - audit log resource manager

  rsct.core.errm - event response resource manager (ERRM)

  rsct.core.fsrm - file system resource manager

  rsct.core.gui - Web-based System Manager RSCT Monitoring GUI

  rsct.core.hostrm - host resource manager

  rsct.core.rmc - resource monitoring and control (RMC)

  rsct.core.sec - cluster security services

  rsct.core.sr - system registry

  rsct.core.utils - miscellaneous utilities

- Additional RSCT filesets required for cluster RMC

  rsct.basic.rte - configuration resource manager, group & topology services

  rsct.compat.basic.rte

  rsct.compat.clients.rte

- Additional RSCT filesets required for custom monitoring

  rsct.core.sensorrm - sensor resource manager

Tips:

- RSCT fixes can be downloaded from:

  http://techsupport.services.ibm.com/server/aix.fdc

- CSM updates and fixes can be downloaded from:

  http://techsupport.services.ibm.com/server/cluster/fixes

2.5.3 Other AIX software

The fileset bos.sysmgt.nim is needed on the NIM server.

Tip: If HMC-controlled servers are included in the cluster, the HMC firmware needs to be at Release 3 Version 1.0 or later, and Java131.rte and Java131.ext.xml4 need to be installed on the management server.

The fileset bos.net.uucp is required for using Equinox Ethernet Serial Provider (ESP) on Linux nodes with an AIX management server in a mixed CSM cluster.

2.5.4 Open source software

The required open source software shipped on the AIX 5L Version 5.2 CD-ROM includes:

- Conserver 7.2.2-6
- Expect 5.32-1
- tcl 8.3.3-1
- tk 8.3.3-1

Open source software not included on the AIX, CSM, or Linux media:

- AutoUpdate V4.3.4 and Perl-libnet V1.0703 - Necessary if you want to perform the software maintenance installation and upgrade of non-CSM RPMs on your Linux-managed nodes from the AIX management server. It can be downloaded from:

  http://freshmeat.net/projects/autoupdate

- openCIMOM Version 7 - Necessary if you want to perform remote hardware control operations for IBM pSeries servers attached with a hardware management console (HMC), found on the “AIX toolbox for Linux applications” CD, or it can be downloaded from:

  http://www.ibm.com/servers/aix/products/aixos/linux/download.html

If you want to use OpenSSH as your remote shell, OpenSSH software can be found on the AIX BonusPak CD. The minimum list of filesets that should be installed (assuming the desired language is English) is:

- openssh.base
- openssh.license
- openssh.man.en_US
- openssh.msg.en_US
- openssh.msg.EN_US

Its prerequisite OpenSSL (openssl.rpm) can be found from "the AIX toolbox for Linux applications" CD or download it from:

http://www.ibm.com/servers/aix/products/aixos/linux/download.html

2.6 Configuration options that fit your needs

Topics in this section cover the configuration options you may choose during the PSSP installation process with the **spsitenv** command, and which comparable options exist for CSM.

2.6.1 Installation choices

PSSP uses the install_image attribute to specify the name of the default network install image to be used for any PSSP node when the install image field is not set. Under CSM, you use NIM to manually create your mksysb resources and allocate them to the NIM machines. See Appendix C, “NIM examples” on page 209 for more information on NIM.

PSSP uses the cw_lppsource_name attribute to specify the name of the directory the necessary NIM filesets are installed from on the control workstation during initial setup (the first time setup_server runs and the control workstation is not configured as a NIM master). In CSM, you use NIM to manually define your lppsource NIM resources and allocate them to the NIM machines.

2.6.2 Security

PSSP has many security service components built-in. You have the choice of using different security services for different PSSP components. For example, you can choose the security service to use for AIX remote command authentication, and you can choose the security service to use for SP trusted services. In CSM, the PSSP services are replaced by RSCT’s security infrastructure. See Chapter 7, “Security and access control” on page 149 for more information.

PSSP gives you an option to have restricted root access for an enhanced level of security (restrict_root_rcmd). In addition, you can choose to use a secure remote command process to replace the **rsh** and **rcp** commands issued by PSSP system management software running on the control workstation (rcmd_pgm, dsh_remote_cmd, remote_copy_cmd).

You must acquire and install the secure remote command software on the control workstation before you can enable a secure remote command process to be used by the PSSP software. The secure remote command software must be running on the control workstation and root must have the ability to successfully use it to issue remote commands to the nodes without being prompted for passwords or passphrases.

Under CSM, you can also replace the default **rsh** with a Kerberos-based one or with **ssh**. We show an example implementing OpenSSH in C.5, “A working example of openssh installation” on page 227.

CSM’s default behavior is that nodes do not need remote command access to the management server. This is comparable to PSSP’s optional restricted root access, so there is no need in CSM to specify restricted root access.

2.6.3 Time service choices - Network Time Protocol (NTP)

PSSP clusters require that time be synchronized between the CWS and the nodes. Your options are as follows:

- If you already have an established NTP server, you can use it.
- You can choose an NTP server from the Internet.
- You can use the NTP server that comes with AIX by default.
- You can choose not to use NTP at all, in which case you must have another way to manage clock synchronization.

If you choose to use NTP (ntp_config, ntp_server), PSSP can set it up for you automatically. CSM does not provide such function at this time, but we show how you can implement NTP in 3.2.6, “NTP configuration” on page 38.

2.6.4 User directory mounting choices - automounter

Automounter is a daemon that dynamically mounts users’ home directories and other file systems when a user accesses the files and unmounts them after a specified period of inactivity. You can choose to have PSSP configure and manage (amd_config) automounter for you. CSM does not provide such function at this time.

2.6.5 System file management choices - file collections

The PSSP file collection component simplifies the task of maintaining duplicate files across the nodes of the SP system. File collections provide a single point of control for maintaining a consistent version of one or more files across the entire system. Configuration attributes include filecoll_config, supman_uid, and supfilesrv_port control the setup of file collections.

Similar function is implemented by Configuration File Manager (CFM) in CSM. PSSP uses the *supman* ID and pulls files from the CWS on a specified port while CFM runs under root on the management server and uses **rdist** to push files out to selected nodes or node groups.

CFM components include a server file repository under /cfmroot, a **cfmupdatenode** command to push updated files out, a cronjob to automatically distribute updates out daily, and event response resource manager (ERRM) conditions and responses to monitor file modifications and push them out whenever they are updated.

2.6.6 User account management choices

If you do not use NIS, LDAP, and so on, to manage user accounts, the SP user account management facility can be used for that purpose under PSSP. It replicates user password files across all nodes in the SP system using the SP file collection facilities.

Configuration attributes include usermgmt_config, passwd_file, passwd_file_loc, homedir_server, and homedir_path.

In CSM, you can still use NIS, LDAP, or your choice of user account management. The SP user account management function can be implemented with CFM, as illustrated in 5.3.1, “User management with CFM” on page 86. An LDAP implementation example can be found in C.4, “Using NIM to install and configure LDAP clients” on page 223.

2.6.7 Accounting choices

The SP accounting utility lets you collect and report on individual and group use of the SP system. This accounting information can be used to bill users of the system resources or monitor selected aspects of the system’s operation.

Configuration attributes include spacct_enable, spacct_actnode_thresh, spacct_exclusive_enable, and acct_master.

No such built-in function exists for CSM. We show how you might implement this in 5.7, “Accounting” on page 94.

2.7 Cluster-related licensed programs

CSM or Cluster Systems Management, as the name suggests, aims at doing clustered system management and doing it well. PSSP is more than a system management software for SP or clustered servers. Many IBM licensed products were built on top of the various PSSP components. IBM is now in the process of porting them to rely on the support of RSCT.

The related licensed program products include the following:

- IBM General Parallel File System for AIX (GPFS)
- IBM High Availability Cluster Multi-Processing for AIX (HACMP)
- IBM LoadLeveler for AIX 5L (LoadLeveler)
- IBM Parallel Environment for AIX (PE)
- IBM Parallel Engineering and Scientific Subroutine Library (PESSL) for AIX
- IBM Virtual Shared Disk (VSD)
- IBM Recoverable Virtual Shared Disk (RVSD)

2.7.1 GPFS

GPFS provides a cluster-wide file system allowing shared access to files spanning multiple disk drives on multiple servers. GPFS is based on a shared disk model, providing low overhead access to disks not directly attached to the application nodes, and using a distributed protocol to provide data coherence for access from any node.

- PSSP V3.5 supports GPFS V2.1 (5765-F64).
- CSM V1.3.2 supports GPFS V2.1 in a RSCT peer domain or in an HACMP cluster.

2.7.2 HACMP

HACMP provides continuous access to data and applications typically through component redundancy and failover in mission-critical environments. With HACMP, customers can scale up to 32 nodes and mix and match system sizes and performance levels, as well as network adapters and disk subsystems to satisfy specific application, network and disk performance needs.

- HACMP V4.5 (5765-E54) and V5.1 (5765-F62) support PSSP V3.5.
- HACMP V4.5 (5765-E54) and V5.1 (5765-F62) support CSM V1.3.2.

2.7.3 LoadLeveler

LoadLeveler is used for dynamic workload scheduling, It is a distributed network-wide job management facility designed to dynamically schedule work on servers, such as the IBM @server pSeries and IBM RS/6000 systems.

- PSSP V3.5 supports LoadLeveler V3.1 (5765-E69).
- CSM V1.3.2 supports LoadLeveler V3.2.

2.7.4 PE

PE is used to develop, debug, analyze, tune, and execute parallel processing applications.

- PSSP V3.5 supports PE V3.2 (5765-D93).
- CSM V1.3.2 supports PE V3.2 (5765-D93).

2.7.5 Parallel ESSL

The ESSL libraries provide a variety of complex mathematical functions for many different scientific and engineering applications. Parallel ESSL provides optimum performance for floating-point-intensive engineering and scientific workloads.

- Parallel ESSL V2.3 (5765-C41) supports PSSP V3.5.
- Parallel ESSL V3.1 (5765-C41) supports CSM V1.3.2.

2.7.6 VSD

IBM Virtual Shared Disk is currently part of PSSP.

- VSD makes data on physical disks accessible from multiple nodes in IBM Virtual Shared Disks that you create; otherwise, the data is accessible only from the node connected to the disk.
- RVSD makes IBM Virtual Shared Disks automatically recoverable in the event of a failure.
- Hashed Shared Disk stripes your data across multiple IBM Virtual Shared Disks and multiple nodes.

VSD is currently supported only on system configurations with an SP switch or SP Switch2. Since CSM does not support either switch, VSD is not supported by CSM at this time. IBM intends to release a new version of VSD which is supported in a CSM cluster.

3

Installation

The PSSP administrator is already familiar with the installation process for PSSP. In this chapter we compare and contrast the installation processes for PSSP and CSM clusters so that anyone familiar with PSSP installation can gain an appreciation for CSM.

The comparison is performed by outlining the key installation tasks and showing how they are performed in the various environments. These tasks are subdivided into the following categories:

- Preparing the management server
- Installing the management server
- Defining nodes to be managed
- Customization of managed nodes
- Installing and configuring managed nodes
- Post-install customization

The detailed steps for the installation of the management server are covered in *An Introduction to CSM 1.3 for AIX 5L,* SG24-6859 and *CSM for AIX 5L: Software Planning and Installation Guide,* SA22-7919.

3.1 Preparing the management server

The management server is the focal point for administration of the cluster and requires special software to manage the cluster. With PSSP the required software is shipped with the PSSP product, with CSM the software comes from a number of locations, as shown in 2.5, “CSM license and components” on page 24.

In addition to the required software, the management server’s network and other physical connections must be in place for the managed nodes you wish to support. For managing SP nodes, the management server requires serial links to each of the frames to provide hardware control. Hardware control for other servers is optional. It is performed by connecting to a hardware control point using IP.

The PSSP control workstation (CWS) and CSM management server setup tasks are similar in concept, although the specific tasks are different. The PSSP installation steps are more comprehensive since they include the NIM setup. Some of these steps are beneficial to the CSM management server if it is also going to be a NIM server.

3.1.1 Configure RS-232 control lines

The CSM management server will support SP frames with APAR IY42379. Each frame needs to be configured on a tty device to allow it to be controlled by the cspd daemon.

All references to the SP frames are made using the device names assigned to the tty’s (for examp[le, /dev/tty1) since CSM has no knowledge of frames, just nodes.

Restriction: Since **cspd** is an updated **hardmon**, it references frame numbers internally. Without the SDR to provide the association between tty devices and frame numbers, it uses a value based on the number of the tty. As a result, the tty device names must be of the form “tty” followed by a number.

3.1.2 Tune network settings

The management server uses IP to manage the nodes in the cluster. If it is to be a NIM server also, as in PSSP, then the network and adapter settings should be configured to improve NIM performance. For more information on the basic tunable values for the management server, refer to chapter 2 of *PSSP Installation and Migration Guide,* GA22-7347. We suggest the same values used on the

CWS be used for the management server, because the same basic functions are being performed over the network interfaces.

3.1.3 Load filesets

If the CSM management server will be configured as a NIM server, the NIM filesets must be installed. This is a step that PSSP performed automatically when the first node was prepared for install.

The NIM software levels used in our environment are shown in Example 3-1.

Example 3-1 NIM software levels used

```
Fileset                      Level  State      Description
  ----------------------------------------------------------------------------
Path: /usr/lib/objrepos
  bos.sysmgt.nim.client     5.2.0.10  APPLIED    Network Install Manager -
                                                 Client Tools
  bos.sysmgt.nim.master     5.2.0.10  APPLIED    Network Install Manager -
                                                 Master Tools
  bos.sysmgt.nim.spot       5.2.0.10  APPLIED    Network Install Manager - SPOT

Path: /etc/objrepos
  bos.sysmgt.nim.client      5.2.0.0  COMMITTED  Network Install Manager -
                                                 Client Tools
```

For more information on NIM, refer to Chapter 1 and Chapter 20 of the *AIX 5L V 5.2 Installation Guide and Reference,* SC23-4389.

3.1.4 Create file systems

PSSP holds all the NIM and configuration data in the /spdata directory tree. CSM also has a dedicated directory called /csminstall. We recommend that you create it as a separate file system in a volume group other than rootvg. If your CSM management server is also going to be configured as a NIM server, the lppsource and SPOT can be placed in the same file system.

3.2 Installing the management server

When the preparation steps have been completed, the CSM management server software can be installed. The CSM software is supplied on the AIX CDs but the CSM Licensed Program Product (LPP) needs to be ordered so that a licensed file can be shipped; this comes on a separate CD.

3.2.1 Install prerequisite software

The IBM @server cluster support Web site contains the latest available documentation and service for CSM for all platforms. It is located at:

```
https://techsupport.services.ibm.com/server/cluster/
```

The CSM management server requires the latest version of csm.core. Since this contains information used by the installation process, the latest cluster code level must be downloaded from the support site and csm.core must then be updated before proceeding.

3.2.2 Copy software images

The CSM filesets and the prerequisite RPM software need to be available before installing CSM. Table 3-1 shows a summary of the software to be copied. We recommend copying these filesets under a single directory using the **gencopy** command.

Table 3-1 Software to be copied

Software Name	Source
Base CSM	AIX media - CD1 and CD2
conserver expect tcl tk	AIX media - CD2
openCIMOM	AIX Toolbox CD
CSM latest maintenance	Cluster software download site

Important: The latest level of CSM software should always be downloaded from:

```
https://techsupport.services.ibm.com/server/cluster/
```

even if a more recent level has been downloaded from the IBM @server support fix delivery center at:

```
http://techsupport.services.ibm.com/server/aix.fdc
```

This is because updates to the prerequisite RPM software are not distributed with AIX updates.

3.2.3 Install management software

Before installing the software, make sure there is at least 4 MB of free space (8192 blocks) in the file system containing /opt. If this fills up, the installation process will fail.

The installation process for the CSM management server 1.3.1 software will transfer all the CSM filesets and their prerequisites from the single directory into the /csminstall directory structure and then install them. If any of the prerequisites are missing, the csm.server fileset installation will fail.

The installation process changes in CSM 1.3.2, and the filesets are no longer held in a CSM specific directory within the /csminstall directory structure. This change is made to allow the administrator to perform the installation of CSM using the existing AIX tools. This applies to the management server and the managed nodes.

Since the installation process no longer identifies existing prerequisites, some additional scripts have been provided for the administrator to verify the installation. They are coded as probes, so use the function of the probe manager.

3.2.4 Set up the authentication environment

PSSP provides Kerberos V4 as a default authentication environment with facilities to configure it. There is no remote command authentication software supplied with CSM, but the management server can be configured to use.rhosts files or the secure shell (ssh).

The use of the openssh package to provide ssh authentication is discussed in *CSM for AIX 5L: Software Planning and Installation Guide,* SA22-7919. We set up one of our test environments as described in the installation scenarios and had to perform some additional NIM configuration to successfully install openssh on the managed nodes. There are more details on this in C.5, “A working example of openssh installation” on page 227.

3.2.5 Configure the management server

When the CSM management server software has been installed, a license needs to be registered. The license file is supplied on a separate CD when you order CSM. If the CD is not available, a 60-day trial license can be used; this can be changed to the full license later.

The CSM configuration is the closest equivalent to the site environment information in PSSP. It does not provide the same functionality (no user management or NTP configuration), but it does define the behavior of the

management server. The configuration settings used on our management server are shown in Example 3-2.

Example 3-2 csmconfig data for our fully licensed server

```
AddUnrecognizedNodes = 0 (no)
ClusterSNum =
ClusterTM = 9078-160
ExpDate =
MaxNumNodesInDomain = -1 (unlimited)
RegSyncDelay = 1
RemoteShell = /usr/bin/ssh
SetupRemoteShell = 1 (yes)
```

Note: The ExpDate is blank once the full license has been installed.

The remote shell used on our SP test system is ssh. The other available shell is rsh. We used ssh because it is more secure and also because we wanted to verify the setup of the remote shell during a CSM node installation. The CSM management server was also used as the NIM server for the node installation. This meant that rsh was configured by NIM during the installation so we could not verify the shell setup for rsh.

3.2.6 NTP configuration

The PSSP site environment information holds basic network time protocol (NTP) information that is used to configure NTP on the CWS and nodes. There is no equivalent configuration operation in CSM, so it needs to be configured separately. The NTP daemon is also started by the PSSP code, so to implement NTP in a CSM cluster we have to create the /etc/ntp.conf files and make sure that the daemon is started at reboot.

A basic NTP configuration has the managed nodes using the management server as a time source. The management server uses another time server in your infrastructure. To implement this in CSM, the configuration file management (CFM) utility can be used. CFM will distribute the files to the managed nodes and can make sure NTP is running.

The /etc/ntp.conf configuration file we used on the management server is shown in Example 3-3; it has two time servers configured.

Example 3-3 /etc/ntp.conf on the management server

```
driftfile /etc/ntp.drift
tracefile /etc/ntp.trace
server 9.169.255.93 version 3
```

```
server 9.169.255.92 version 3
```

The configuration file for the managed nodes is shown in Example 3-4; it has the management server configured as the time server. The file is located in /cfmroot/etc/ntp.conf so that it is distributed by CFM.

Example 3-4 /cfmroot/etc/ntp.conf for managed nodes

```
driftfile /etc/ntp.drift
tracefile /etc/ntp.trace
server 192.168.1.254
```

To make sure that NTP recognizes any updates made to the configuration, and that NTP is running after the initial installation, a CFM post exit is used. See Example 3-5.

Example 3-5 /cfmroot/etc/ntp.conf.post CFM exit to refresh NTP

```
#!/bin/ksh
#-----------------------------------------------------------------------------#
#                                                                             #
#   ntp.conf.post : CFM Post exit for /etc/ntp.conf                           #
#                                                                             #
#   Function : Make sure that NTP is restarted after updating ntp.conf        #
#                                                                             #
#-----------------------------------------------------------------------------#

   stopsrc -s xntpd 2>/dev/null  # May be errors if not already running
   startsrc -s xntpd

exit $?
```

Once NTP has been configured, it needs to be started after each reboot. The usual way to do this is to uncomment the NTP line in /etc/rc.tcpip. PSSP did not do this, and for simplicity, we used a different approach also. The System V startup process locates startup scripts in a directory matching the system run level and then executes them. This is how the openssh daemon is started. We can distribute the startup scripts using CFM and no additional configuration is required.

The startup script we used is called /etc/rc.d/rc2.d/S20xntpd and is shown in Example 3-6 on page 40. As each startup script should have a partner shutdown script, this file has also been linked to /etc/rc.d/rc2.d/K20xntpd.

Example 3-6 /etc/rc.d/rc2.d/S20xntpd startup/shutdown script for NTP

```
#!/bin/ksh

################################################
# name: S20xntpd/K20xntpd
# purpose: script that will start or stop the xntpd daemon.
################################################

case "$1" in
start )
        startsrc -s xntpd
        ;;
stop )
        stopsrc -s xntpd
        ;;
* )
        echo "Usage: $0 (start | stop)"
        exit 1
esac
```

The files we need on the managed nodes are identical to those on the management server, so they were created as links in the /cfmroot directory structure, as shown in Example 3-7.

Example 3-7 /cfmroot contents for /etc/rc.d/rc2.d

```
$ ls -l /cfmroot/etc/rc.d/rc2.d
total 0
lrwxrwxrwx   1 root     system           24 Jun 04 10:47 K20xntpd ->
/etc/rc.d/rc2.d/K20xntpd
lrwxrwxrwx   1 root     system           24 Jun 04 10:47 S20xntpd ->
/etc/rc.d/rc2.d/S20xntpd
$
```

In a cluster that contains just AIX-managed nodes, this completes the setup of NTP. If Linux nodes are added to the cluster, then the configuration must be modified so that the files distributed operate correctly in the environment running on the managed node.

It is still possible to do this using CFM node groups. All the files placed in the /cfmroot structure should be defined for the node group AIXNodes only; this variation is shown in Example 3-8.

Example 3-8 NTP configuration limited to AIX nodes

```
$ find /cfmroot/ -name \*ntp\* -exec ls -l {} \;
```

```
lrwxrwxrwx   1 root     system           24 Jun 04 10:47
/cfmroot/etc/rc.d/rc2.d/K20xntpd._AIXNodes -> /etc/rc.d/rc2.d/K20xntpd
lrwxrwxrwx   1 root     system           24 Jun 04 10:47
/cfmroot/etc/rc.d/rc2.d/S20xntpd._AIXNodes -> /etc/rc.d/rc2.d/S20xntpd
-rw-rw-r--   1 root     system         1024 Jun 04 10:51
/cfmroot/etc/ntp.conf._AIXNodes
-rwxr--r--   1 root     system          660 Jun 04 10:45
/cfmroot/etc/ntp.conf.post._AIXNodes
$
```

Refer to 5.2, “CFM versus file collections” on page 83 for more details on CFM.

3.3 Defining nodes to be managed

Once the CSM management server software has been installed, it is ready to manage nodes defined to it. The node definitions can be entered manually or, if the management server has hardware control, they can be discovered and then entered into the CSM database using the data collected during the discovery.

Note: This is a process that is similar to frame discovery in PSSP.

3.3.1 Defining hardware control points

CSM supports HMCs in the same way PSSP does: it uses the openCIMOM package to establish a connection to the HMC. This connection requires a user on the HMC that is known to the management server. This user name and its password must be stored on the CSM management server so that it can manage the hardware.

The hardware control points (HCPs), including the **cspd** code, are controlled by the IBM.HWCTRLRM subsystem and should not be started independently.

3.3.2 Listing candidate nodes

In a PSSP environment, the CWS manages all the nodes it can see. It is the only machine that can control the nodes in the SP, and all LPARs detected on an HMC-managed machine are added to the SDR when they are discovered.

The CSM approach is to use the **lshwinfo** command (refer to the man pages for more information on this command) to list the hardware attached to the different hardware control points. The administrator determines which nodes will be defined to the management server.

The CSP support allows SP frames to be interrogated. It does not distinguish between node types, so it lists MCA nodes even though they are not supported by CSM.

On our test SP system (see Figure 1-1 on page 4) there are 3 node types:

9076-WCNr	Nodes 1, 3, 5 and 7
9076-260	Nodes 9 and 10
MCA	Nodes 11, 12, 13 and 14

The **lshwinfo** output for this frame is shown in Example 3-9.

Example 3-9 lshwinfo for SP frame

```
#lshwinfo -c hmc1 -p hmc -h
Hostname::PowerMethod::HWControlPoint::NodeId::LParId::HWType::HWModel:: . .
no_hostname::csp::/dev/tty1::1::::::::
no_hostname::csp::/dev/tty1::3::::::::
no_hostname::csp::/dev/tty1::5::::::::
no_hostname::csp::/dev/tty1::7::::::::
no_hostname::csp::/dev/tty1::9::::::::
no_hostname::csp::/dev/tty1::10::::::::
no_hostname::csp::/dev/tty1::11::::::::
no_hostname::csp::/dev/tty1::12::::::::
no_hostname::csp::/dev/tty1::13::::::::
no_hostname::csp::/dev/tty1::14::::::::
```

Any nodes that have been previously defined will have the correct hostname displayed. Those that show as no_hostname are not known to CSM.

After customizing the output data shown in Example 3-9, it can be used as input to the **definenode** command to create skeleton entries in the CSM database for the listed nodes.

At the time of writing, the **definenode** command creates the console attributes for HMC-attached nodes only. This information must be entered manually for SP nodes. Alternatively, the script shown in Example 3-10 can be used.

Example 3-10 consoledef sets CSM console data for CSP nodes

```
#!/bin/ksh
#-------------------------------------------------------------------------------#
#                                                                               #
#   consoledef : Define CSM console data for undefined csp nodes                #
#                                                                               #
#-------------------------------------------------------------------------------#

   lsnode -w "ConsoleMethod='' and PowerMethod='csp'"|
      while read node ; do
```

```
    lsnode -x -n ${node} -a HWControlPoint,HWControlNodeId|
       sed "s/, / /g"|
       read HCP HND rest
    chnode -v -n ${node} ConsoleServerName=${HCP} \
                         ConsolePortNum=${HND} \
                         ConsoleMethod=csp
done
```

The script shown in Example 3-10 identifies all CSP nodes that have no console method set, and then uses the hardware information for the nodes to define the console. If the console method is set, then no changes are made to the CSM data.

3.3.3 Acquire the hardware Ethernet address

Like PSSP, CSM is able to acquire the MAC addresses for the network adapters of nodes that are connected to an HCP. This is done by interrogating the firmware on the node to return the adapter information. If required, the **getadapters** command can update the node information in the CSM database with the adapter details.

Notes:

- Prior to CSM 1.3.2, the **getadapters** command power cycles the machines. CSM 1.3.2 avoids the use of the firmware if the node is up issuing a **dsh** command to the node.
- The firmware is accessed using the **rconsole** command, so the node must have its Hardware Control Point (HCP) and console configuration defined in the CSM database before this step can be completed.
- CSM does not provide the bootptab.info functionality that PSSP has.

3.4 Customization of managed nodes

PSSP allows certain adapter types to be entered into the SDR and configures the adapters following the installation of the node. NIM now has limited support for additional adapters so that it can perform the customization.

Using PSSP, the additional adapter details can be stored in the SDR using the **spadaptrs** command. The equivalent NIM operation is the **nimadapters** command, which is documented in *AIX 5L Version 5.2 Commands Reference, Volume 4, N-R,* SC23-4118. The NIM information is held in a directory corresponding to the adapter_def resource that was referenced when using the

nimadapters command. The adapter details can be provided on the command line or in a stanza file.

CSM introduces support for additional adapters in CSM 1.3.2 for AIX. An update to the **getadapters** command provides an option to write the adapter hardware details to a stanza file. The administrator must then enter the IP parameters for the adapters in the stanza file before executing the **nimadapters** command to define the additional adapters to NIM.

The customization takes place if the adapter_def resource is allocated in NIM for a bos_inst or cust operation only. The supported adapter types are shown in Table 3-2.

Table 3-2 Supported secondary adapters

Adapter Type	Where supported		Comments
	PSSP	CSM/NIM	
at	N	N	ATM
css	Y	N	SP switch
en	Y	Y	Ethernet
et	N	Y	IEEE 802.3 ethernet
fi	Y	N	fddi
ml	Y	Y	Aggregate switch address, supported in PSSP 3.5, and in AIX for the HPS
sn	N	Y	Switch network
tr	Y	N	Token ring
vi	N	N	VIPA addressing

PSSP also allows a default gateway to be defined on a node. This is only of use where there is a single network gateway that is reached using an adapter that can be configured during the node installation. If the network gateway is on the SP switch network, or on an adapter type not supported by PSSP as a secondary adapter, the default gateway has to be defined as the CWS for the install. The administrator then has to set the actual default gateway using post installation scripts or standard routing protocols, which require the use of *routed* or *gated*.

In PSSP, the default route is held in the SDR and is configured as part of the PSSP customization of the node. There is no additional action (or exit code) required on the part of the administrator.

3.5 Installing and configuring managed nodes

In order to bring a new node into a PSSP cluster, PSSP requires an overwrite install of the AIX image on this node. This is achieved through a NIM mksysb installation with dedicated PSSP resources allocated to complete the PSSP installation and configuration on the node.

The PSSP **setup_server** command performs the NIM customization and prepares all the other information required to complete the installation of the node. As part of the preparation, it deallocates resources that have previously been allocated so that the resources required can be allocated without error.

The CSM administrator has to make sure that the NIM state of a machine is ready to have new resources allocated and is prepared for a NIM operation. Example 3-11 shows a script that can be used to prepare a machine for a new NIM operation. It is equivalent to the **unallnimres** command that is supplied with PSSP. Note that a client machine is first reset if its NIM Cstate does not allow to deallocate the resources. Note that this is a NIM reset, not a power reset.

Example 3-11 Deallocating NIM resources - unallnimres

```
#!/bin/ksh
#
# usage: unallnimres <managed_node>
NODE=$1

lsnim $NODE >/dev/null
if [ $? -ne 0 ]; then
  echo "`basename $0` - lsnim $NODE failed, exiting."
  exit $?
fi

if [ `lsnim -c resources $NODE | wc -l` -eq 0 ]; then
  echo "`basename $0` - nim machine $NODE has no resources allocated, done."
  exit 0
fi

lsnim -a Cstate -z $NODE | grep ":ready for a NIM operation:" >/dev/null
if [ $? -ne 0 ]; then
  nim -Fo reset $NODE
fi

nim -Fo deallocate -a subclass=all $NODE
```

3.5.1 CSM and PSSP relationship to NIM

The relationship between CSM and NIM, both running on a management server, is shown in Figure 3-1.

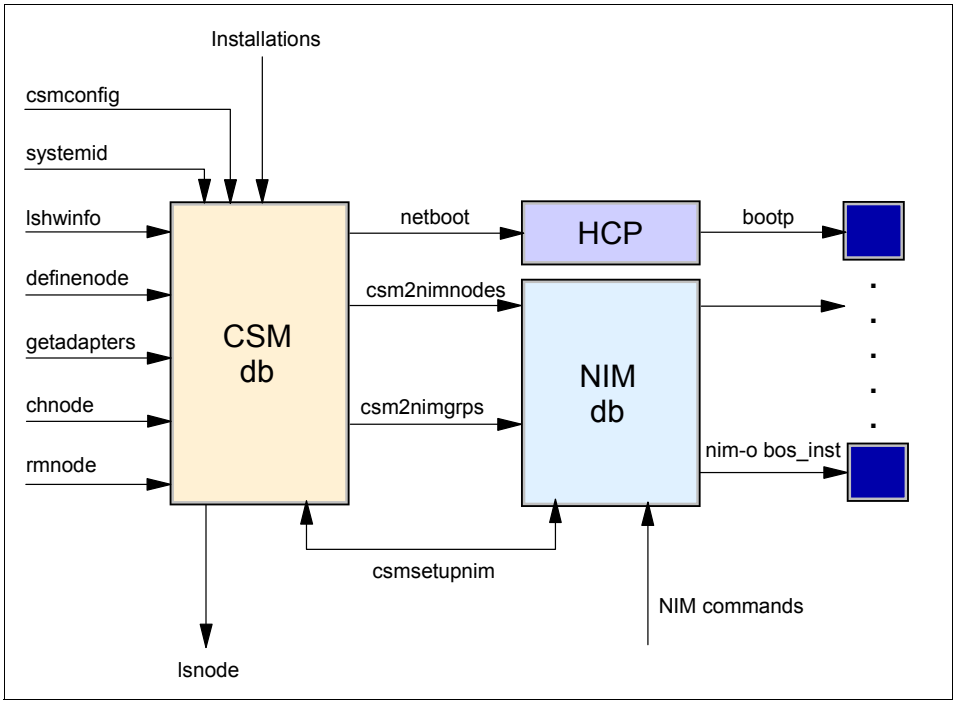


Figure 3-1 CSM and NIM relationship

In contrast, the equivalent relationships for PSSP are shown in Figure 3-2 on page 47. Since there is no PSSP equivalent of the **updatenode** command, the PSSP diagram uses the upper right boxes to represent SP nodes. For CSM, it does not matter where the machines are, since they are all controlled via the hardware control point.

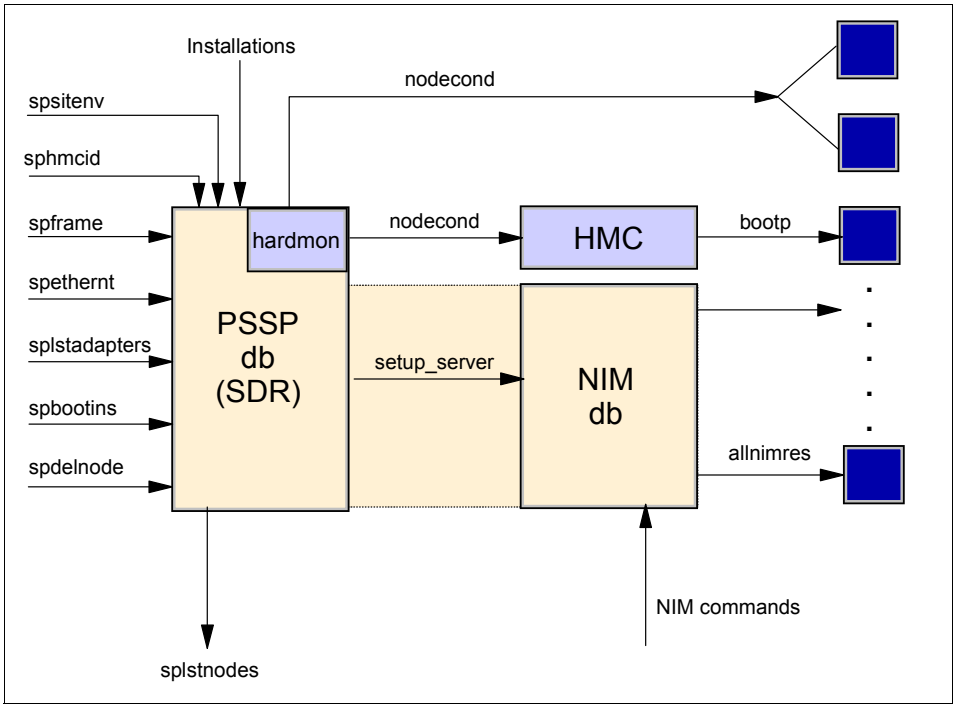


Figure 3-2 PSSP and NIM relationship

3.5.2 Preparing for installation in CSM

CSM is more flexible than PSSP because it does not require hardware control, or does not require that the nodes it manages are installed through CSM. If the node has the CSM client code installed, it can be managed using the **updatenode** command on the management server.

If a node is installed using the CSM management server as the NIM server, the updatenode processing is performed automatically as part of the installation if the **csmsetupnim** command was used to allocate the required resources.

3.5.3 Preparing for installation in NIM

Before a node can be installed using NIM, it must be defined as a NIM client. The **csm2nimnodes** command extracts information about the node from the CSM database and creates a NIM client definition that can be used to install the node.

CSM does not allocate any installation resources in NIM except for the **csmprereboot** script, used to complete the CSM configuration after the installation has completed.

The NIM resources required to install the node must be allocated by the administrator. To simplify this process, we recommend the use of NIM resource groups so that all the commonly used NIM resources can be allocated to a node by allocating the resource group. C.2.1, “NIM resource groups” on page 211 provides more details on the use of resource groups.

The bosinst data file

The examples in Appendix C, “NIM examples” on page 209 use a bosinst_data resource that does not contain any disk location information. This means the installation takes place on the first disk identified after the network boot of the node.

PSSP creates separate bosinst_data resources for each node being processed. The files contain the disk location information that was previously entered into the SDR. We recommend that you use the same approach for all NIM installations, because this prevents accidental overwriting of data on other disks.

There are separate stanzas in the bosinst file for the disk location information. Some sample entries are shown in Example 3-12.

Example 3-12 Sample location information in bosinst file

```
target_disk_data:
    LOCATION = 10-60-00-0,0
    SIZE_MB =
    HDISKNAME =
    CONNECTION =
    PVID =
    SAN_DISKID =

target_disk_data:
    LOCATION = 10-60-00-1,0
    SIZE_MB =
    HDISKNAME =
    CONNECTION =
    PVID =
    SAN_DISKID =
```

rte installations

A NIM rte installation uses the filesets in the lpp_source resource to install a new system. The results are as if the system were installed from a CD. In addition, NIM performs additional processing based on the resources allocated to the machine being installed.

mksysb installations

All PSSP installations are performed using a mksysb installation from a node backup, or the minimal system image supplied with PSSP. There is no minimal image supplied with CSM, but mksysb installations can still be performed.

When performing a mksysb installation, we recommend the use of installp_bundle resources to make sure that some of the key filesets that CSM depends on are updated to the latest level after the mksysb image has been restored. See C.3, “Installing additional filesets” on page 222 for an example installp_bundle containing the CSM and RSCT filesets.

3.6 Post-install customization

There are three customization scripts in PSSP that execute at different points during the installation process. There are no direct replacements in CSM, but they can be implemented as detailed in Table 3-3. In PSSP, they are all distributed to the nodes using **`tftp`** from their boot install server. Our CSM solution distributes the files from the NIM server, which typically resides on the management server.

Table 3-3 Post-install customization comparison

PSSP Command	When called	CSM Implementation
`script.cust`	After PSSP installation and before the node reboots. Also called during a PSSP customize operation.	Allocate a NIM script resource.
`firstboot.cust`	On the first reboot following an install only.	Allocate a NIM script resource that will bootstrap itself into /etc/inittab, and then remove its inittab entry, for a bos_inst operation only. The script should do nothing for a cust operation.

PSSP Command	When called	CSM Implementation
tuning.cust	During a customize boot and from /etc/rc.sp at every server IPL.	Allocate a NIM script resource that will bootstrap itself into /etc/inittab, for execution at every reboot, for a bos_inst operation only. For a cust operation the script should copy itself, so the new code will be executed at each IPL, and then execute the tuning section of the code.
The NIM bootstrap technique is illustrated in C.4, “Using NIM to install and configure LDAP clients” on page 223.		

CSM also provides some post-installation exits:

- CSM post install scripts

  CSM 1.3.2 executes scripts located in specific directories under /csminstall/csm/scripts. The list of scripts to execute before, and after, the NIM reboot is built when **csmsetupnim** is executed. The order of execution is based upon the names of the scripts.

- CFM .post scripts

  The configuration file management facility calls exits before and after a file is updated. We rely on this for our NTP configuration sample (see Example 3-5 on page 39).

While it is possible to use CFM exits to perform post-installation processing, this only happens when **cfmupdatenode** is executed to distribute the file to the node that has been installed. With CSM 1.3.1 this does not happen automatically during the installation of a node. It occurs when **cfmupdatenode** is executed by the administrator or during the scheduled execution from the crontab of the management server. The scheduled execution is set for midnight by default.

CSM 1.3.2 uses a condition and response in event response resource manager (ERRM) to detect when the mode of a node changes from “Installing” to “Managed”. When this is detected, the **cfmupdatenode** command is executed for the node.

Tip: If an administrator does not wish to have **cfmupdatenode** run automatically, they can disable the condition and response pair using the command **stopcondresp “NodeFullInstallComplete” “RunCFMToNode”**.

3.7 The setup_server command

The **setup_server** command in PSSP performs all the preparations for the installation of nodes. The basic steps are shown in Table 3-4 and are compared to the equivalent operation in CSM. A detailed analysis of the processing is shown in D.3, “Comparison with PSSP” on page 242.

Table 3-4 setup_server vs. CSM outline

Function performed	CSM administrator equivalent
Remove NIM clients for nodes that no longer exist or have changed hostname.	None.
Make sure CWS is a NIM master.	Install bos.sysmgt.nim filesets.
Create kerberos tickets for all nodes.	**csmsetupnim** prepares ssh keys.
Make NIM interfaces for the server.	Configure NIM interfaces manually.
Make necessary NIM resources.	Define individual resources and resource groups to simplify allocation. This includes mksysb and bosinst_data resources.
Make NIM clients for all nodes.	**csm2nimnodes**
Create config_info data.	**csmsetupnim** creates customization files for CSM post-installation processes.
Create install_info data.	
Check lppsource contents.	Checked by NIM.
Export the pssplpp directory.	**csmsetupnim** exports directory with customization data.
Allocate NIM resources to clients.	Use resource groups to allocate install resources and bosinst file.

3.8 CSM software maintenance

In the 1.3.1 release, CSM software maintenance on the managed nodes is accomplished with the help of the **updatenode** command. Essentially the same steps as outlined in D.1, “A closer look at CSM’s updatenode script” on page 232 are performed, as long as they are applicable.

Starting in CSM 1.3.2, the CSM software is no longer installed and maintained on AIX servers by the **updatenode** command provided with CSM. Instead, the standard AIX software maintenance tools (**installp**, NIM) must be used by the administrator to ensure that the software is installed and updated. As with all

versions of CSM, the managed nodes must not be at a higher software level than the management server.

When installing nodes using NIM, the CSM software is installed during an rte install. If a mksysb install is used, the software levels in the mksysb will be on the nodes. To ensure that the node always has the current software level installed, we recommend the use of an installp_bundle resource containing the RSCT and CSM filesets. C.3, “Installing additional filesets” on page 222 provides more details and includes a sample bundle file that will update the RSCT and CSM software (Example C-11 on page 223).

3.9 nodecond vs. netboot

The CSM **netboot** command and the PSSP **nodecond** command both initiate network boots of attached servers by accessing the firmware through the hardware control point and initiating the bootp requests from the install adapter.

Since netboot uses the **rconsole** command to control the node being processed, it can force a write session. This means that if someone already has the console open for write, the netboot process won’t fail, which is a common problem with **nodecond**.

Note: This is only true for console sessions opened from the management server. A virtual terminal opened directly on the HMC cannot be forced into read only mode by the **rconsole** command. This is the same for PSSP.

With CSM, **netboot** continues and the previous write session becomes read only. Once **netboot** has completed, the other session is switched to write again. If someone is to use **rconsole** to force a write session while the **netboot** is running, the netboot will fail.

4

PSSP SDR vs. RSCT SR

The System Data Repository (SDR) is one of the key components of PSSP. This repository is the place where all static configuration information about the cluster components is held. During the installation of a PSSP cluster, information about the site environment, frames, nodes, switches, and the installation setup is stored in SDR classes and SDR files. The SDR is also used by some subsystem to store dynamic properties of the cluster. The most prominent examples of SDR classes with dynamic contents are the host_responds and switch_responds classes, which are updated by daemons running on the control workstation.

Whenever the control workstation or a node is rebooted, or a PSSP-specific subsystem is (re)started, configuration information is retrieved from the SDR. If changes in the configuration data require a reconfiguration of a node, this is performed by PSSP scripts during the boot process.

Although the SDR is completely transparent to the administrator in normal operation, many PSSP system administrators have found themselves in situations where a detailed understanding of the SDR and the ability to browse as well as change its content is vital to resolve operational problems.

In this chapter, we give a summary of the SDR's architecture and the data it holds, and compare it to the way cluster configuration information is handled in Cluster Systems Management (CSM) by using the infrastructure that RSCT provides.

4.1 SDR architecture summary

- The SDR is described in detail in Appendix E, “System Data Repository”, of the *PSSP Administration Guide,* SA22-7348. For the purpose of this comparison, we ignore any partition-related issues and assume that the cluster is one partition. We also assume that PSSP security is not set to DCE. (If PSSP security is set to DCE, access control is finer-grained. For details, refer to *PSSP Administration Guide,* SA22-7348.)

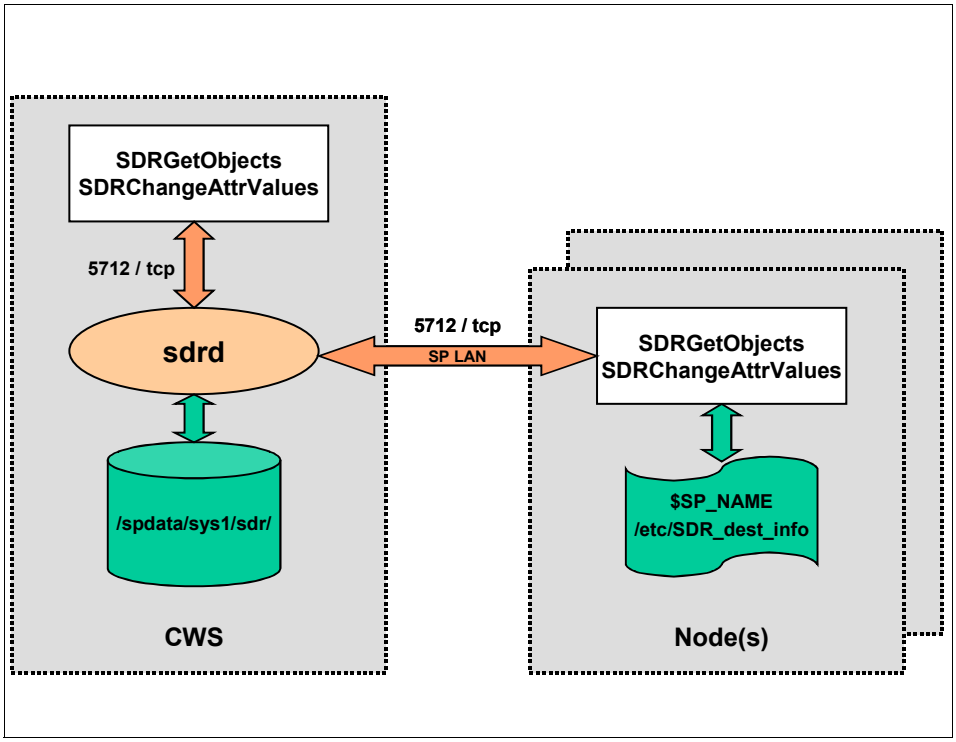


Figure 4-1 PSSP SDR architecture

Figure 4-1 shows the SDR architecture. The following facts are important when comparing the SDR to the CSM data repository:

- The SDR has a client/server architecture. The sdrd daemon is the server component running on the CWS, under SRC control. It listens only on the SP LAN Ethernet, ports sdr=5712/tcp and sdrprot=1712/tcp.
- Clients find their server through $SP_NAME or /etc/SDR_dest_info. Note that a client’s node number is stored in the ODM class CuAt on each node.

- Access control for root on any machine on the SP LAN can read and write the SDR. All other users on a machine that is on the SP LAN can read the SDR, and requests from other networks are ignored.
- The SDR has an object model where SDR classes can contain objects with attribute=value pairs. In addition, the SDR can be used as a cluster-wide repository to store and retrieve files.
- All SDR classes and files are stored on the CWS in the directory tree /spdata/sys1/sdr/*. All SDR class data is stored as plain text files.
- Direct access to the SDR is possible through a set of PSSP commands starting with SDR, for example **SDRGetObjects**. They are documented in the *PSSP Command and Technical Reference (2 Volumes),* SA22-7351. Note that this includes commands to create new SDR classes, objects, attributes, and files.
- sdrd log files can be found in /var/adm/SPlogs/sdr/:
  - SDR_config.log (installation/configuration problems)
  - sdrdlog.syspar_ip_addr.pid (operational problems)

In normal operation, interaction with the SDR does not take place through the SDR commands but through higher-level PSSP commands that read or write to the SDR under the covers. For example, **spsitenv** writes information about the site environment into the SDR class “SP”, and **splstdata -e** lists that data.

4.2 Data in the SDR

To list the SDR classes and files on a particular cluster, use the commands **SDRListClasses** and **SDRListFiles**. The classes provided by IBM are described in Appendix E, “The System Data Repository” of the *PSSP Administration Guide,* SA22-7348. To list the objects in a class, use the **SDRGetObjects** *<classname>* command.

The SDR *classes* can be broadly grouped as follows:

- Classes describing properties of the SP cluster as a whole:

```
SP             #  site environment
SP_Restricted  #  r-cmd setup, login restrictions
SP_ports       #  ports used by PSSP
```

- Classes describing the frames and nodes in the SP, as well as their network interfaces and disks/volume groups. These classes are also vital for node installation:

```
Adapter        #  network adapter configuration (en0, css0, ...)
Frame          #  hardware protocol, tty/s1_tty/HMC definitions
```

```
Node            #  PSSP's main node class
NodeControl     #  directory of which node types support which HW ops
NodeExpansion   #  node expansion units (RIOs)
NodeGroup       #  PSSP node groups
Node_VPD        #  node type, model, and serial number
Volume_Group    #  node rootvg config (PVs, mirroring, install setup)
```

- Classes for the optional SP Switch/SP Switch2 and dependent nodes like the SP Switch Router 9077: CSS_*, Switch*, Aggregate_IP, and Dependent*
- Syspar* (used for SP partitioning)
- TS_* and Network (used by the PSSP version of RSCT topology services)
- GS_* (used by the PSSP version of RSCT group services)
- EM_* and PMAN* and pman* (PSSP event and problem management)
- HSD_* and VSD_* and RVSD_* (used by the PSSP VSD components)
- Gpfs (used by GPFS)
- JM_* (JobManager -- legacy, no longer used by PSSP)
- The host_responds and switch_responds classes, holding dynamic state

In this chapter, we focus on the first two groups of SDR classes because these are directly related to the cluster systems management functions contained in CSM. We compare their contents to the CSM configuration data in 4.4, “CSM data in the system registry” on page 68.

In CSM, the classes for topology services, group services, event management and problem management are all contained in RSCT. They are described in Chapter 6, “Monitoring” on page 111 as well as in the RSCT product documentation and the redbook *A Practical Guide for Resource Monitoring and Control (RMC),* SG24-6615.

The SP Switch/SP_Switch2 is not supported with CSM, and the future IBM high-speed interconnect will be managed on the HMC (“out-of-band”) rather than by the cluster systems management software (“in-band”). VSD and GPFS are not yet available for CSM clusters.

There are only a few SDR *files.* They are typically stored in the SDR through configuration commands issued by the system administrator:

- expected.top.* (the SP Switch/SP Switch2 annotated topology file)
- hats.machines.* (used by the PSSP version of RSCT topology services)
- mmsdr* (used by GPFS)

Nodes can retrieve these files to obtain an up-to-date copy of the configuration data stored in them.

4.3 RSCT system registry overview

Like PSSP, CSM needs some persistent data storage mechanism to hold cluster configuration data. It also needs to keep track of dynamic states. Unlike PSSP, CSM does not implement its own configuration database. CSM relies on the Reliable Scalable Cluster Technology (RSCT) component of AIX 5L to store configuration information. It also uses the RSCT security infrastructure (CtSec) for authentication and access control. For a general discussion of RSCT, refer to *A Practical Guide for Resource Monitoring and Control (RMC),* SG24-6615, chapter 3 of *IBM (e)server Cluster 1600 Managed by PSSP 3.5: What's New,* SG24-6617, and the RSCT product documentation. The security aspects of RSCT are discussed in Chapter 7, "Security and access control" on page 149.

RSCT has a "system registry" (SR) that supports both persistent data (which is kept on disk) and dynamic data (which is only held in memory). The system registry is contained in the RSCT resource management and control (RMC) framework, which is available to other IBM software components but is not externalized through an API.

In this section, we summarize the SR architecture. 4.4, "CSM data in the system registry" on page 68 describes the CSM configuration data stored in the SR and compares it to the equivalent SDR classes. 4.5, "Listing configuration data: splstdata equivalents" on page 76 discusses the high-level commands in CSM that read and write this data and compares them to the PSSP commands that perform similar actions.

4.3.1 The distributed RMC infrastructure

Before we describe how CSM exploits the RSCT infrastructure to store its configuration information and state data, it is useful to take a look at the general RMC architecture. The following conceptual differences between the SDR and the RSCT SR are immediately noticeable:

- The SDR is an SP-specific component that does not exist on every AIX machine. There is one SDR server in an SP cluster, installed and set up on the CWS by explicit PSSP commands. All SP nodes are clients of that SDR. Different SP systems have different SDRs, and there is no interaction between them.
- With AIX 5L, the rsct.core.* filesets are installed by default during the base AIX installation. Therefore, each AIX 5L machine provides the full RSCT Resource Management and Control (RMC) infrastructure as well as the RSCT client commands. Each AIX 5L machine can *serve* resources (including data stored in its local system registry), and can be a *client* of any other machine's resources. This is a truly distributed environment.

Note: The data that CSM stores in the system registry is managed through CSM-specific resource managers and mainly resides on the management server. This is comparable to the PSSP case. However, the RMC/SR infrastructure can be exploited in many different ways (such as on individual machines, in peer domains, etc.) and so is more general than the SDR.

Figure 4-2 shows the architecture of RMC and the various communication paths that exist in this distributed environment.

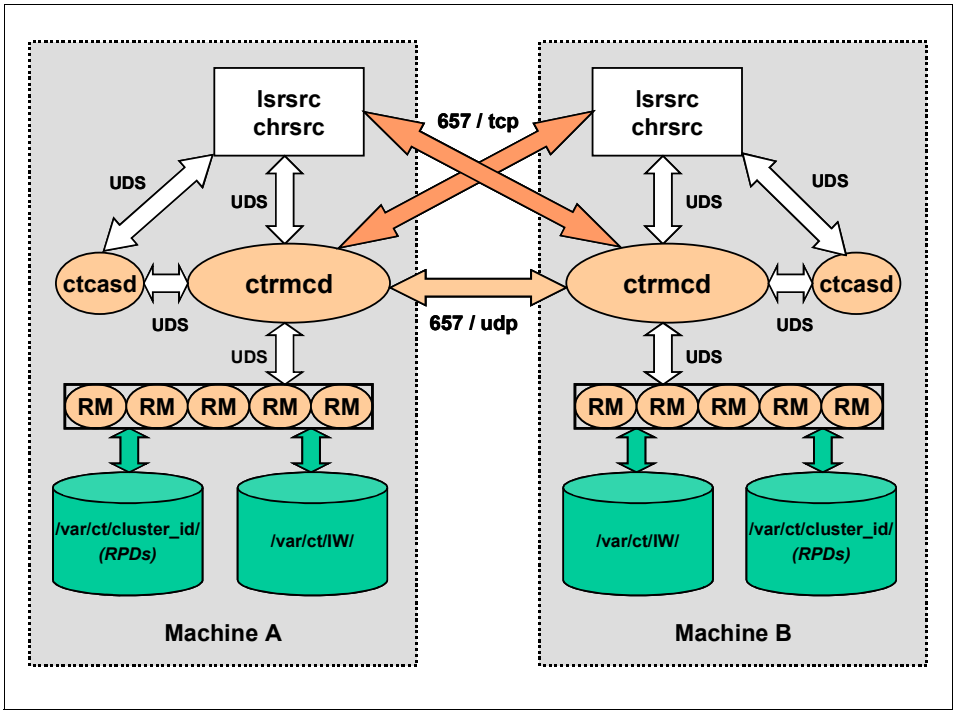


Figure 4-2 RSCT's distributed RMC architecture.

RMC subsystems

The RMC infrastructure includes the following subsystems:

- Each AIX 5L machine runs the RMC daemon rmcd, which is under SRC control. The CtSec library starts the Cluster Authentication Service (CAS) daemon ctcasd on demand when CtSec authentication is required. This daemon is also under SRC control. Both are in the SRC group rsct.

Note: SRC needs root authority to start SRC subsystems. To allow an RSCT command invoked by a non-root user to start ctcasd, the CtSec library uses the SUID-root command **ctstrtcasd** to start ctcasd.

- RMC itself only provides an abstract view of resource classes and resource instances. Different resource managers (RMs) provide access to the actual resources. Some RMs are shipped with RSCT, while other components such as CSM implement their own RMs. All RMs are under SRC control, in the SRC group rsct_rm.

You can use the **lssrc** command to get information about the state of the resource managers.

Example 4-1 Output of the lssrc -ls ctrmc command

```
[c5aix04][/]> lssrc -ls ctrmc
Subsystem         Group            PID          Status
 ctrmc            rsct             18928        active

Trace flags set:
         cci = 1        config = 1       cmdproc = 0          dcfg = 1
        eval = 0          evgn = 1           gsi = 1          init = 1
       insts = 0       iolists = 0           mcp = 1          msgs = 0
        obsv = 0           pci = 1         query = 0           rcv = 0
         rdi = 0           reg = 1        rmctrl = 1         routg = 1
      rstree = 0          sami = 0         sched = 0           sec = 1
         shm = 0           sri = 0

Configuration Data Base version from local copy of CDB:
        140556,515870407,0

Daemon started on Monday 07/28/03 at 15:42:48
Daemon has been running 0 days, 0 hours, 55 minutes and 35 seconds
Daemon is the MCP of a management domain
Management domain node count:  1
Daemon communications are enabled

Logical Connection Information for Local Clients
    LCID          FD           PID     Start Time
       0          12          20642    Monday 07/28/03 15:42:48
       2          23          17294    Monday 07/28/03 15:42:58
       3          24          17294    Monday 07/28/03 15:43:08
       4          25          15714    Monday 07/28/03 15:43:38

Logical Connection Information for Remote Clients
    LCID          FD           PID     Start Time
```

```
Resource Manager Information
         Name           ClassKey      ID         FD       SHMID
IBM.HostRM                   1          0         13       2097156
IBM.ERRM                     1          1         16            -1
IBM.AuditRM                  1          2         17            -1
IBM.DRM                      1          3         -1            -1
IBM.DMSRM                    1          4         18            -1
IBM.FSRM                     1          5         -1            -1
IBM.HWCTRLRM                 1          6         19            -1
IBM.CSMAgentRM               1          7         20            -1
IBM.ConfigRM                 1          8         -1            -1
IBM.SensorRM                 1          9         21            -1
IBM.ServiceRM                1         10         22        524289
IBM.WLMRM                    1         11         -1            -1

Highest file descriptor in use is 25

Internal Daemon Counters
    GS init attempts =          0  GS join attempts =          0
    GS resp callback =         27  writev() stream  =        938
    msgs wrtn stream =        886  maxd wrtn stream =         43
    CCI conn rejects =          0  RMC conn rejects =          0
    Retry req msg    =          0  Retry rsp msg    =          0
    Intervl usr util =          0  Total usr util   =         17
    Intervl sys util =          3  Total sys util   =         44
    Intervl time     =      12001  Total time       =     324014
    lccb's created   =         31  lccb's freed     =         26
    ctb's created    =        322  ctb's freed      =        315
    smb's created    =          5  smb's freed      =          5
    SAMI recs creatd =          6  SAMI recs freed  =          0
    mnpd's created   =          0  mnpd's freed     =          0
    ptb's created    =          0  ptb's freed      =          0
    rsrc inst creatd =          2  rsrc inst freed  =          0
    var inst creatd  =          5  var inst freed   =          0
    Events regstrd   =          7  Events unregstrd =          0
    Insts assigned   =          5  Insts unassigned =          0
    Smem vars obsrv  =          0  State vars obsrv =         16
    Preds evaluated  =         21  Events generated =         22
    PRM msgs to all  =          0  PRM msgs to peer =          4
    PRM resp msgs    =          0  PRM msgs rcvd    =          8
    PRM_NODATA       =          6  PRM_HBRECV       =        169
    PRM_BADMSG errs  =          0  reserved         =          0
    Sched q elements =         16  Free q elements  =         13
    xcb allocated    =       1016  xcb free         =       1008
    xcb free cmdlist =        312  xcb free mcmdlst =          3
    xcb free clntcmd =          0  xcb free clntrsp =        434
    xcb free frmpeer =          8  xcb free allpeer =          0
    xcb free topeer  =          5  xcb free peerrsp =          0
    xcb free rmrsp   =          0  xcb free rmcmd   =        246
```

```
    xcb free unknown =          0  reserved         =          0
    Generic q elems  =         16  Free gq elems    =         16
    Rsearch nodes    =        127  Free RS nodes    =         70
    Rsrc Hndl CBs    =        127  Free RHCBs       =        100
    Peer Msg Anchors =         16  Free PMAs        =         16
    Cached PAttr     =         16  Free CPAs        =          0
    Event Rsrc CBs   =        127  free ERCBs       =        121
    NG RH references =          0  free NRRs        =          0
    Node Name CBs    =         64  free NNCBs       =         62
    rhcl's created   =        126  rhcl's freed     =        126
    ACL's created    =          3  ACL's freed      =          0
    Sec rec methods  =          0  Sec authent      =         29
    Missed sec rsps  =          0  Wake sec thread  =         30
    Wake main thread =         30  Enq sec request  =         30
    Deq sec request  =         30  Enq sec response =         30
    Deq sec response =         30

Daemon Resource Utilization Last Interval
User:                  0.000 seconds    0.000%
System:                0.030 seconds    0.025%
User+System:           0.030 seconds    0.025%

Daemon Resource Utilization Total
User:                  0.170 seconds    0.005%
System:                0.440 seconds    0.014%
User+System:           0.610 seconds    0.019%

Data segment size:  1233K
```

Running the command **`lssrc -ls ctrmc`** produces a detailed status report of the RMC subsystem, as shown in Example 4-1, including the active resource managers.

Note: Although it is possible to use SRC commands such as **`startsrc`** and **`stopsrc`** to start and stop the RSCT subsystems, the RSCT command **`rmcctrl`** should normally be used for this task to avoid conflicts with ongoing RMC operations. **`rmcctrl`** can act on the RMC subsystem alone, or on RMC and all the underlying resource managers.

RSCT subsystem tracing can be controlled through the SRC commands **`traceson`** and **`tracesoff`**, and the (binary) traces in /var/ct/*/log/mc/*<RM>*/trace can be viewed by the **`rpttr`** command (which also has a **`-f`** option similar to **`tail -f`**). This may be useful in diagnosing RM problems.

RMC communication paths

There are several communication paths between these subsystems and the client commands that access the RMC infrastructure. Figure 4-2 on page 58 shows these communication paths:

- RMC to RMC

  All communication between RMC daemons (running on different machines) is done through UDP, using service rmc=657/udp. Such daemon-to-daemon communication takes place in several cases:

  - Machines that are in the same RSCT peer domain require some communication between the peers to keep their common data in sync. In addition, if $CT_MANAGEMENT_SCOPE is set to 2 (peer domain scope), commands will be run on all the peer nodes. This is done through daemon-to-daemon communication from the machine that is the target of the original RSCT command to all other peer RMCs.
  - If $CT_MANGEMENT_SCOPE is set to 3 (management domain scope) and the target of the RSCT command is a management server, the RMC daemon on the management server will connect to the RMC daemons on all the managed nodes through daemon-to-daemon communication.
  - Phoenix Reliable Messaging (PRM) is RSCT's reliable messaging protocol, which is used for "heartbeating" between the nodes in an RSCT management domain.

    Note: This is different from a peer domain where heartbeating is done through RSCT's topology services (TS) subsystem. TS also uses UDP for communication between the TS daemons, but uses a different port and protocol.

- RMC to RMs

  The RMC daemon directly connects only to resource managers (RMs) residing on the same machine. For this connection, it always uses UNIX-Domain Sockets (UDS). If access to remote resources is required, the RMC daemon contacts the remote RMC through UDP, and the remote RMC daemon in turn connects to its local RM to access the resource in question.

- RMC to RSCT client

  RSCT clients use two different communication methods to connect to an RMC daemon, depending on the setting of the $CT_CONTACT environment variable:

  - If $CT_CONTACT is *not* set, the client uses UDS to contact the RMC daemon on the machine on which it is invoked.

– If $CT_CONTACT *is* set, the client uses a TCP connection to contact the RMC daemon on the node designated by the IP name/address that's stored in $CT_CONTACT, using service rmc=657/tcp. This is also true if the $CT_CONTACT variable points to the local machine.

> **Note:** These two communication methods require different RMC ACLs. Refer to 7.3.2, "RMC access control files and identity mapping service" on page 164 for details.

- CAS daemon to RSCT client

  CtSec authentication is always done locally (on the machine on which the client command originates). The connection between the client requiring authentication and ctcasd always uses UDS.

- RMC to GUI

  Web-Based System Management (WebSM) clients always contact the Web-based System Manager server using the TCP service wsmserver=9090/tcp (the port can be configured, but 9090 is the default), and the Web-based System Manager server then contacts the local RMC daemon using UDS.

> **Note:** The RSCT UNIX domain sockets can be found in /var/ct/IW/soc/. You can use:
>
> ```
> lsof -U | grep /var/ct/IW/soc/
> ```
>
> to list all currently open RSCT sockets.
>
> **lsof** is available from ftp://lsof.itap.purdue.edu/pub/tools/unix/lsof/ or in the AIX Linux Application Toolbox CD.

Note that rmcd listens on all network interfaces of the machine. This is different from the SDR case where sdrd only listens on the network interface used for the SP LAN. Because RMC can be used in many different scenarios, such as peer domains using different networking topologies, it is not possible to restrict rmcd to listen only on the CSM management VLAN.

This implies that RSCT client commands on any AIX 5L machine can contact the RMC daemon on any other AIX 5L machine to which a network connection exists. The amount of data it can actually access depends on the RMC ACLs on the target machine, of course. See 7.3.2, "RMC access control files and identity mapping service" on page 164 for details.

Note: You can completely disable remote RMC connections through the **`rmcctrl`** command. However, it may have undesired effects in a CSM management domain.

4.3.2 Where system registry data is stored

The system registry data resides in subdirectories within the /var/ct/ directory. Depending on the environment in which a machine operates, there may actually be one or more directories holding SR data:

- When an AIX 5L machine is installed, RSCT will create a system registry for that individual machine, which in RSCT jargon is called an *Individual Workstation* (IW). RSCT assigns each IW a unique cluster_id. In /var/ct/ you will find a subdirectory with a name that matches the RSCT cluster_id, and a symlink named IW that points to that subdirectory. Different machines will have different IW cluster_id values, but you can always get to the local system registry through the /var/ct/IW symlink. The registry data is in /var/ct/IW/registry/.
- When the machine becomes a member of an RSCT peer domain, RSCT assigns a unique cluster_id for the peer domain and maps this to the cluster_name that is selected when the peer domain is created. On each node in the peer domain, there will be a subdirectory named like the peer domain's cluster_id, and a symlink named like the cluster_name that points to that subdirectory. The registry data for the peer domain's resources resides in a /var/ct/*cluster_id*/registry/ subdirectory, similar to the IW case. The resource management framework (RMF™) will ensure the consistency of these directories among all the peer nodes, using group services.

  Note: If a machine is a member of multiple peer domains, there will be different cluster_id subdirectories and cluster_name symlinks, of course. Only one peer domain can be active at any time.

- A management domain does *not* introduce a new system registry like a peer domain. It actually uses the IW registry on the management server and the IW registries on the managed nodes to store the CSM-specific data. There is also no need for any synchronization of SR data because the management server is the single point of control in the management domain. (The data on the managed nodes is only changed through updates triggered from the management server. These are all RMC actions.)

The data in the system registry is stored in binary form. It can only be accessed through RSCT commands such as **`lsrsrc`** or the higher-level commands provided by some of the resource managers (like **`lscondition`** used by ERRM).

Refer to 4.5, “Listing configuration data: splstdata equivalents” on page 76 and the RSCT product manuals for details.

Note that the setting of the $CT_MANAGEMENT_SCOPE environment variable controls which resources are accessible through the RSCT commands:

- If it is unset or set to 0 or 1, RSCT commands have *local* scope and only the IW data of the target machine is accessible.
- If it is set to 2, RSCT commands have *peer domain* scope and all resources in the system registry of the active peer domain are accessible.
- If it is set to 3, RSCT commands have *management domain* scope. All resources in the IW registries of the managed nodes are available.

Refer to the RSCT product manuals for more details, in particular for the behavior when a machine is in both a management domain and a peer domain.

4.3.3 Querying a machine’s RMC “personality”

In this section, we show some commands you can issue to find out if an AIX 5L machine is just an independent workstation, participates in a peer domain, or is a management server or managed node within a management domain. We also show how to list the available resources and resource managers. This may be useful if the state of a machine is unknown or you want to understand which resources are provided by which software products.

- The first thing you can do is to list the /var/ct/ directory. A typical case may look like Example 4-2 (the user/group information has been deleted for simplicity).

Example 4-2 `List of /var/ct/ directory`

```
c67rp10 root / > cd /var/ct ; ls -al
total 120
drwxrwxr-x   6   4096 May 23 13:30 .
drwxr-xr-x  22   4096 Apr  2 14:21 ..
drwxr-xr-x   8    256 May 23 16:31 2TcqY1PXV9t0wGSvexwr
drwxrwxr-x   8    256 Apr  2 14:21 675588684
-rwxr-xr-x   1      0 May 23 16:31 IBM.AuditRM.stderr
-rwxr-xr-x   1      0 May 23 16:31 IBM.CSMAgentRM.stderr
-rwxr-xr-x   1      0 May 23 16:31 IBM.ConfigRM.stderr
-rwxr-xr-x   1      0 May 23 16:31 IBM.DRM.stderr
-rwxr-xr-x   1      0 May 23 16:31 IBM.ERRM.stderr
-rwxr-xr-x   1      0 May 21 15:32 IBM.FSRM.stderr
-rwxr-xr-x   1      0 May 23 16:31 IBM.HostRM.stderr
-rwxr-xr-x   1      0 May 23 16:31 IBM.SensorRM.stderr
-rwxr-xr-x   1      0 May 23 16:31 IBM.ServiceRM.stderr
lrwxrwxrwx   1     17 Apr  2 13:12 IW -> /var/ct/675588684
```

```
-rw-r--r--   1  44356 May 23 16:31 RMstart.log
drwxrwxr-x   2   4096 May 27 11:56 cfg
-rw-r--r--   1    536 Apr  2 13:27 first.log
drwxrwxr-x   2    256 Apr  2 13:27 lck
lrwxrwxrwx   1     30 Apr  4 09:14 rpdc67rp91012 ->
                                   /var/ct/2TcqY1PXV9t0wGSvexwr
```

By looking at the symbolic links, you can see that this machine's IW data resides in ./675588684/, and that it is also a member of a peer domain named rpdc67rp91012, which has its data in ./2TcqY1PXV9t0wGSvexwr/. The *.stderr files give some indication about startup problems of RMs.

- To find out about a machine's role in a management domain, you can call the **/opt/csm/bin/mgmtsvr** command without arguments:
  - If the machine is a managed node and the management server value is set, the command will output the name of the management server and a return code of zero.
  - If all other cases, a non-zero return code will be returned together with an explanatory text for the state of the machine. The **mgmtsvr** manpage contains an explanation of the possible return codes.
- If the machine is a management server, the **lsnode** command (which is contained in the csm.server fileset) can be used to list the managed nodes. Setting the CT_MANAGEMENT_SCOPE variable to management domain scope (3) allows the management server to access resources on the managed nodes.
- The managed nodes in a management domain have no knowledge of their siblings, and they are not allowed to run commands with management domain scope (3).
- In a peer domain, the **lsclcfg** command will show the active cluster_name and cluster_id, which should match the contents of the /var/ct/ directory shown above (for an IW, **lsclcfg** would show a cluster_name of IW). You can also use the **lsrpdomain**, **lsrpnode** and **lscomg** commands to obtain more information about the RSCT Peer Domain (RPD). Every peer node can run commands with CT_MANAGEMENT_SCOPE set to peer domain scope (2).

The following commands can be run to find out about the resource managers, resource classes and resource instances:

- To list the resource managers on a machine, you can use **lssrc** to list the subsystems in SRC group rsct_rm. A typical output is shown in Example 4-3 on page 67.

Example 4-3 SRC group rsct_rm listing

```
c5aix07 root / > lssrc -g rsct_rm
Subsystem         Group            PID          Status
 IBM.ERRM         rsct_rm          17544        active
 IBM.CSMAgentRM   rsct_rm          18068        active
 IBM.DMSRM        rsct_rm          5010         active
 IBM.ServiceRM    rsct_rm          14202        active
 IBM.AuditRM      rsct_rm          16274        active
 IBM.FSRM         rsct_rm          16534        active
 IBM.HostRM       rsct_rm          19892        active
 IBM.HWCTRLRM     rsct_rm          4522         active
 IBM.ConfigRM     rsct_rm                       inoperative
 IBM.SensorRM     rsct_rm                       inoperative
 IBM.WLMRM        rsct_rm                       inoperative
```

Note: If you use **`lslpp -w`** or **`lslpp -f`** to search for RMs, beware that some daemons do not follow the usual naming convention "IBM.*<Name>*RM". In particular, the IBM.ERRM subsystem is implemented by a daemon named IBM.ERrmd not IBM.ERRMd.

- The **`lsrsrc`** and **`lsrsrcdef`** commands list all known resources. Passing them a resource name lists detailed attributes of that resource. Beware that you have to use the **`-Ad`** (or **`-Ab`**) option to list non-persistent attributes, and **`-p`** with a suitable parameter to list attributes that are not public. The **`-c`** option lists properties of the resource class rather than individual resource instances.
- Note that RSCT provides no command to list all resource classes that are owned by a given resource manager. The lsrm script in Example 4-4 does this by finding all resource classes with a matching *mgr_name* attribute.

Example 4-4 lsrm - List all classes owned by a resource manager.

```
#!/bin/ksh
#
# usage: lsrm <RM> - List all classes owned by RM <RM>
#
lssrc -s $1 >/dev/null
if [ $? -ne 0 ]; then
  echo "‘basename $0‘ - resource manager $1 is not known."
  exit $?
fi
echo "Resource classes for Resource Manager $1:"
lsrsrcdef -cx | sed 's/\"//g' | while read R
do
  lsrsrcdef -cx $R | grep "mgr_name" | grep "\"$1\"" > /dev/null
  if [ $? -eq 0 ]; then
    echo $R
```

```
    fi
  done
```

The resource managers and resource classes that are provided by RSCT are described in detail in *RSCT for AIX 5L: Technical Reference,* SA22-7890. The following section covers the CSM resource managers and resource classes.

4.4 CSM data in the system registry

In this section, we discuss the resource managers that are provided by CSM, and in particular the data that they hold in the system registry. As mentioned in 4.3.2, “Where system registry data is stored” on page 64, all CSM management domain data is stored in the IW system registry, and almost all data resides on the management server.

4.4.1 Persistent data on the CSM clients

The csm.client fileset is installed by default during base AIX install. It adds one CSM resource manager to the RMC infrastructure: IBM.CSMAgentRM. This resource manager implements a single resource class, IBM.ManagementServer. This class holds the management server’s Name, which is equivalent to the $SP_NAME variable or the /etc/SDR_dest_info file in PSSP. It also holds the node’s LocalHostname, which is roughly equivalent to the PSSP node_number:

- IBM.ManagementServer has the following persistent attributes:

```
$ dsh -w node5 lsrsrc IBM.ManagementServer
node5: Resource Persistent Attributes for: IBM.ManagementServer
node5: resource 1:
node5:  Name          = "sp6cws-en0"
node5:  Hostname      = "sp6cws-en0"
node5:  ManagerType   = "CSM"
node5:  LocalHostname = "node5"
node5:  ClusterTM     = "9078-160"
node5:  ClusterSNum   = ""
node5:  NodeNameList  = {"node5"}
```

The IBM.ManagementServer class is populated by the **mgmtsrv** command, which is called when a node is added to the CSM cluster. Unless the management server is also configured to be a managed node, this resource class is not populated on the management server. You can call **mgmtsrv** without arguments to list the current status of a machine.

Note: PSSP stores the node_number in the CuAt ODM class (the CWS is node 0):

```
[c179s][/]> odmget -q attribute="node_number" CuAt   #   or -q name="sp"

CuAt:
        name = "sp"
        attribute = "node_number"
        value = "0"
        type = "R"
        generic = "DU"
        rep = "s"
        nls_index = 24
```

In a CSM cluster, the nodes do not store any CSM-specific data in the ODM. All configuration data is held in RMC resource classes. The LocalHostname in the IBM.ManagementServer resources roughly corresponds to PSSP's node_number.

4.4.2 Persistent data on the CSM management server

The csm.server fileset, which is only installed on the management server, contains two resource managers: IBM.DMSRM and IBM.HWCTRLRM.

Domain management server resource manager

The domain management server resource manager, IBM.DMSRM, implements the resource classes that are used to store CSM information about nodes and node groups, as well as some cluster-wide setup data:

- Class IBM.ManagedNode
  - This class corresponds to the SDR class Node in PSSP, but it also contains information that in PSSP resides in the Adapter, Frame, Node_VPD, and Volume_Group classes.
  - It is populated by the **definenode** command. The **lshwinfo** and **getadapters** commands also provide input to this class.
  - It includes attributes for hardware control, which are then propagated to the hardware control resource classes managed by the IBM.HWCTRLRM resource manager (see "Hardware control resource manager" on page 71 for details). You can use:

    ```
    lsrsrc -l IBM.ManagedNode
    ```

    to list all public and persistent attributes. Most of them are described in Chapter 2, "Planning for CSM" of *CSM for AIX 5L: Software Planning and*

Installation Guide, SA22-7919. You can also refer to the *manpage* of the **definenode** command.

- Class IBM.NodeGroup
  - This class corresponds to the SDR class NodeGroup (with the addition of dynamic node groups, which do not exist in PSSP).
  - It is populated and changed with the **nodegrp** command.
  - It holds static as well as dynamic node groups. In Example 4-5, NIMNodes is a dynamic group with a SelectStr, whereas regatta_ler_grp is a static group with a MemberList.

Example 4-5 Listing IBM.NodeGroup

```
$ lsrsrc IBM.NodeGroup
Resource Persistent Attributes for: IBM.NodeGroup
resource 13:
        Name          = "NIMNodes"
        ValidateNodes = 1
        MemberList    = {}
        SelectStr     = "InstallMethod=='nim'"
        ExpMemberList =
          "c67rp10.ppd.pok.ibm.com","c66_ler1_p01.ppd.pok.ibm.com"}
        NodeNameList  = {"c5aix07"}
resource 18:
        Name          = "regatta_ler_grp"
        ValidateNodes = 1
        MemberList    = {"c66_ler1_p01.ppd.pok.ibm.com"}
        SelectStr     = ""
        ExpMemberList = {"c66_ler1_p01.ppd.pok.ibm.com"}
        NodeNameList  = {"c5aix07"}
```

- Class IBM.PreManagedNode

  This is a deprecated class, not used by CSM 1.3.

- Class IBM.NodeAuthenticate

  This class does not have any resources (it only supports actions, which are invoked when an unauthenticated node joins the cluster. You can list them using the **lsactdef** and **lsactdef -c** commands).

- Class IBM.DmsCtrl
  - Although it has much fewer attributes, this class roughly corresponds to the SDR class SP since it stores some settings that apply to the whole management domain. It also contains r-command setup information that is the equivalent of the SP_Restricted SDR class.
  - It is populated with the **csmconfig** command. Most of the attributes in this class have a corresponding **csmconfig** flag to set them, as shown in

Example 4-6. The MaxNumNodesInDomain and ExpDate attributes are set when a CSM license is installed through **csmconfig -L**.

Example 4-6 Listing the IBM.DsmCtrl attributes

```
$ lsrsrc IBM.DmsCtrl
Resource Persistent Attributes for: IBM.DmsCtrl
resource 1:
        AddUnrecognizedNodes = 0
        MaxNumNodesInDomain  = -1
        RemoteShell          = "/usr/bin/rsh"
        ClusterTM            = "9078-160"
        ClusterSNum          = ""
        SetupRemoteShell     = 1
        ExpDate              = ""
        RegSyncDelay         = 1
        Frequency            = 12
        Sensitivity          = 8
        NodeNameList         = {"c5aix07"}
```

Hardware control resource manager

The IBM.HWCTRLRM resource manager implements two resource classes which hold the information that is needed for hardware control of the nodes through the **rpower** and **rconsole** commands, as well as the nodes' MAC addresses, which are needed for NIM installation.

Note: All the hardware control information is actually stored in the resource class IBM.ManagedNode. This data is propagated to IBM.NodeHwCtrl under the covers: IBM.HWCTRLRM registers with RMC to get notified of changes in the IBM.ManagedNode class. It monitors the dynamic attributes ConfigChanged and ChangedAttributes of IBM.ManagedNode. It then updates the appropriate attributes in its resources whenever a persistent attribute is changed in IBM.ManagedNode. Refer to 4.4.3, "Dynamic CSM data" on page 74 for a description of the dynamic attributes.

- Class IBM.NodeHwCtrl
  - This class contains all hardware-related data for the managed nodes. In PSSP, this data resides in the SDR classes Node (MAC address), Node_VPD (type, model, and serial number), and Frame (hardware control information for power and consoles).

    Sample output of the resource persistent attribute for IBM.NodeHwCtrl for an SP node:

    ```
    $ lsrsrc IBM.NodeHwCtrl
    Resource Persistent Attributes for: IBM.NodeHwCtrl
    ```

```
resource 5:
   Hostname              = "node9"
   HWControlPoint        = "/dev/tty1"
   HWControlNodeId       = "9"
   PowerMethod           = "csp"
   HWType                = "010005312"
   HWModel               = "9076-WCN"
   HWSerialNum           = "0005312"
   LParID                = "000"
   ConsoleServerName     = "/dev/tty1"
   ConsoleServerNumber   = ""
   ConsoleMethod         = "csp"
   ConsolePortNum        = "9"
   InstallAdapterMacaddr = "0004AC4947E9"
   NodeNameList          = {"sp6cws"}
```

Sample output of the resource persistent attributes for IBM.NodeHwCtrl for an HMC-connected node/LPAR:

```
$ lsrsrc IBM.NodeHwCtrl
Resource Persistent Attributes for: IBM.NodeHwCtrl
resource 2:
    Hostname              = "c66_ler1_p01.ppd.pok.ibm.com"
    HWControlPoint        = "c66hmc.ppd.pok.ibm.com"
    HWControlNodeId       = "c66_ler1_p01"
    PowerMethod           = "hmc"
    HWType                = "01100B90A"
    HWModel               = "7028-6C4"
    HWSerialNum           = "100B90A"
    LParID                = "001"
    ConsoleServerName     = "c66hmc.ppd.pok.ibm.com"
    ConsoleServerNumber   = ""
    ConsoleMethod         = "hmc"
    ConsolePortNum        = ""
    InstallAdapterMacaddr = "0002554F009C"
    NodeNameList          = {"c5aix07"}
```

- Class IBM.HwCtrlPoint
  - This class has no direct correspondence in PSSP. It has a similar role to the Frame class in PSSP, but without the implication that the frame concept in PSSP brings with it. It also does not include any information about the consoles, which resides only in IBM.NodeHwCtrl.

    Sample output of the resource persistent attributes for IBM.HwCtrlPoint for an SP HCP (a TTY to the SP frame):

```
$lsrsrc -l IBM.HwCtrlPoint
Resource Persistent Attributes for: IBM.HwCtrlPoint
resource 1:
   Name            = "/dev/tty1"
```

```
    PowerMethod     = "csp"
    PollingInterval = 300
    NodeNameList    = {"sp6cws"}
```

Sample output of the resource persistent attributes for IBM.HwCtrlPoint for an HCP for HMC-connected nodes:

```
Resource Persistent Attributes for: IBM.HwCtrlPoint
resource 1:
    Name            = "c66hmc.ppd.pok.ibm.com"
    PowerMethod     = "hmc"
    PollingInterval = 300
    NodeNameList    = {"c5aix07"}
```

As mentioned above, these classes are populated from the IBM.ManagedNode class and not directly through CSM commands.

VSD resource manager

PSSP includes the Virtual Shared Disk (VSD) component, which stores its configuration data in the SDR. VSD is not yet available on CSM 1.3. A “Cluster ready VSD” release will be shipped as part of RSCT, as a PTF against AIX 5.2 ML01. The rsct.vsd.vsdrm fileset contains the VSD resource manager, which implements the following VSD resource classes to store the data that was previously stored in the SDR:

```
IBM.vsdnode:
    NodeNameList
    VSD_nodenum
    VSD_adapter
    VSD_min_buddy_buffer_size
    VSD_max_buddy_buffer_size
    VSD_max_buddy_buffers
    VSD_maxIPmsgsz
    RVSD_version
    CVSD_cluster_name
    CVSD_node_number
    cvgs_defined

IBM. vsdgvg:
    global_group_name
    local_group_name
    primary_node
    secondary_node
    eio_recovery
    primary_ts
    secondary_ts
    server_list
    vsd_type
```

```
IBM.vsdtable:
   VSD_name
   global_group_name
   logical_volume_name
   minor_number
   size_in_MB
   lv_blk0_pdev
   lv_blk0_pbn


IBM.rvsdrestrictlevel:
   level
```

These VSD resources map easily to the VSD* SDR classes of PSSP.

4.4.3 Dynamic CSM data

In addition to the persistent attributes discussed in the previous sections, the CSM-specific resource classes also contain a few dynamic attributes. You can list them by using the **-A d** option of the **lsrsrc** command.

- The resource *classes* typically have two dynamic attributes that are set when resource instances are created or destroyed. These are typically used to trigger callback actions. You can run:

  ```
  lsrsrc -c -Ad ResourceClass
  ```

  to list the dynamic attributes of a resource *class*.

- For resource *instances*, ConfigChanged is one of RSCT's standard dynamic attributes, which is also supported by all CSM resources. It is implicitly set to true whenever a persistent attribute of a resource instance changes. By monitoring this attribute, automatic response actions can be triggered whenever a persistent configuration change is made.

- The IBM.ManagedNode resources contain several dynamic attributes as shown in Example 4-7.

Example 4-7 IBM.ManagedNode resource attributes

```
$ lsrsrc -Ad IBM.ManagedNode
Resource Dynamic Attributes for: IBM.ManagedNode
resource 1:
   ConfigChanged     = 1
   Status            = 127
   PowerStatus       = 1
   ChangedAttributes = {"HWControlPoint"}
   InstallStatus     = "PreManaged"
resource 2:
```

```
ConfigChanged     = 1
Status            = 1
PowerStatus       = 1
ChangedAttributes = {"Mode"}
InstallStatus     = "Managed"
```

Example 4-7 shows the following:

- Both nodes have undergone a change of a persistent attribute as shown by ConfigChange=1.
- The ChangedAttributes attribute is another standard dynamic attribute. Whenever ConfigChanged indicates a change of a persistent attribute, this attribute contains a list of the names of the affected persistent attributes. In this case, one node had just been attached to another HMC and the ChangedAttributes field reflects the update to the HCP field. The second node had just finished installation so its Mode attribute has changed.
- Status displays information about the reachability of the node through RMC heartbeating. This corresponds to PSSP's host_responds status.
- PowerStatus shows the power status of the node as reported by the HCP. This corresponds to PSSP's nodePower or powerLED status.
- InstallStatus: A dynamic resource attribute that mirrors the Mode attribute (the mirroring is done automatically by DMSRM). ERRM events can register for changes to this dynamic attribute, which allows CSM (or the user) to run responses when the Mode of the node changes. The CSM tool monitorinstall has similar functionality to PSSP's SDR attributes that provide the install status of a node.

> **Note:** `lsnode` does include some of the dynamic attributes in its output, notably the Status and PowerStatus attributes.

- The IBM.NodeHwCtrl resources also have a PowerStatus dynamic attribute, which is kept in sync with the IBM.ManagedNode data through the same event notification mechanism that's used to synchronize the persistent attributes in those two classes.
- IBM.NodeGroup has a MembershipChanged dynamic attribute that contains details about a nodegroup membership change (which is flagged by a nonzero ConfigChanged attribute).

In summary, many of the dynamic attributes in the CSM resource classes are used by CSM internally. The two attributes that are particularly useful for the system administrator are the PowerStatus and Status attributes, which correspond to PSSP's nodePower and host_responds. See Appendix B, "Cluster startup and shutdown" on page 193 for an example of how to make use of these

attributes in your own scripts. Both attributes are also returned by the **lsnode** command.

4.5 Listing configuration data: splstdata equivalents

While PSSP includes low-level **SDR*** commands that can be used to directly interact with the SDR, in most cases a higher-level PSSP command is more convenient to list and/or change SDR data. The same is true for RSCT and the system registry:

- There are some low-level RSCT commands which interact directly with resource classes, notably the **chrsrc**, **lsactdef**, **lsrsrc**, **lsrsrcdef**, **mkrsrc**, **refrsrc**, and **rmrsrc**. They are described in detail in chapter 2, "Resource monitoring and control (RMC): commands," of *RSCT for AIX 5L: Technical Reference,* SA22-7890.
- Each resource manager normally includes some high-level commands that are more convenient to use with the specific resources it provides.

Figure 3-1 on page 46 and Figure 3-2 on page 47 show which PSSP and CSM commands interact with the configuration databases in PSSP and CSM, and 4.4, "CSM data in the system registry" on page 68 discusses some of the CSM details.

In this section, we discuss how you can list the cluster configuration data that the PSSP command **splstdata** provides in the CSM. This command has more than a dozen options that select different data sets to be reported. We discuss them in three groups: Options that have no equivalent in CSM, those which list data that is not stored in the configuration database but retrieved locally from the nodes, and those for which there is an equivalent in CSM.

4.5.1 splstdata options that have no equivalent in CSM

Some **splstdata** options list SDR objects for features that are present in PSSP, but are not supported in CSM:

- **-A** for SP accounting
- **-g** for aggregate IP on SP Switch2
- **-p** for SP partitions
- **-s** for SP Switch/SP Switch2
- **-u** for SP user management

Consequently, there is no equivalent for these options in CSM.

4.5.2 Configuration data not stored in the database

Some **splstdata** options do not report SDR data, but collect configuration data from the nodes that are “up” at the time the command runs:

- **-d** collects **df** output.
- **-h** collects **lscfg** output.
- **-i** collects **netstat -in | grep -v ':'** output.

Under CSM, you have to run a **dsh** command yourself to collect this data.

4.5.3 splstdata options that do have a CSM equivalent

The remaining **splstdata** options do SDR queries for features that *do* have a correspondence in CSM. They mainly deal with the general cluster setup or *site environment*, and the specific frames, nodes, adapters, and network installation information.

splstdata -f lists frame database information

CSM does not have the rigid frame concept that PSSP uses. From a hardware control perspective, the hardware control point (HCP) replaces the frame concept. You can run:

```
lsrsrc -l IBM.HwCtrlPoint
```

to list all your HCPs (but without any s1_tty information), or you can use:

```
lsrsrc -l IBM.NodeHwCtrl
```

to list all of the nodes’ hardware control data including both the HCP and console information. See Example 5-12 on page 107 for a sample script that cycles through all HCPs. The **lsnode** command also provides all of this information.

splstdata -n lists node configuration information

This data is provided by the CSM command **lsnode**. You may also want to run **lsrsrc** directly against the IBM.ManagedNode class, as discussed in “Domain management server resource manager” on page 69.

splstdata -b lists node boot/install information

There is no full equivalent to this command in CSM. Some of the data that relates to the PSSP boot/install information resides in the IBM.ManagedNode class and can be listed with **lsnode**. However, many of the specific installation attributes are missing: Information such as PSSP’s install_disk attribute is not held in CSM’s IBM.ManagedNode, and there is also no VG concept such as the selected_vg in CSM. In CSM, you have to resort to NIM to define these details.

splstdata -v lists volume group information

While the IBM.ManagedNode class stores some information about the desired software levels, CSM has no support for rootvg mirroring or specifying the physical volumes on which to install. Similar to the boot/install information, the system administrator has to handle these issues directly in NIM or through a post-installation step.

splstdata -a lists LAN database information

The adapter information returned by the **`splstdata -a`** command is covered in CSM as follows:

- The install adapter is stored in the IBM.ManagedNode class and can be retrieved by **`lsnode`**, including the adapter's MAC address.
- Since SP Switch/SP Switch2 is not supported in CSM, there is no switch adapter information in CSM.
- Automatic configuration of secondary adapters can be done by NIM using the adapter_def resource. For more information, refer to 3.4, "Customization of managed nodes" on page 43.

splstdata -e lists site environment database information

The SP *site environment data* consists of the contents of the SP and the SP_Restricted SDR classes. Most of the data in the SP SDR class does not apply to CSM: PSSP features such as SP accounting, SP user management, NTP setup, and file collections do not exist in CSM.

PSSP stores the default values for the AIX and PSSP versions, as well as a default mksysb image name for node installation, in the SP class. CSM has a different approach. No default is stored in the SR, rather the versions that are installed on the management server are used when a node definition does not explicitly set the desired software versions.

The resource class that holds the remaining site environment data for CSM is IBM.DmsCtrl. The **`csmconfig`** command provides the equivalent output to **`splstdata -e`**, showing the cluster's machine type and model, as well as the more important remote shell setup settings. (All of these values are also *set* through the **`csmconfig`** command.)

Note: Our test environments did not include extension nodes because there is no support for extension nodes. Therefore, we did not investigate how the **`splstdata -x`** and **`splstdata -X`** commands map to CSM.

4.5.4 Node groups

Note that PSSP does not provide a means to list node groups through **splstdata**, although they are also stored in the SDR. You have to use commands such as **nglist** to list PSSP node groups. In CSM, you can use the **nodegrp** command to list (and change) CSM node groups.

5

Administration

This chapter discusses the tools and techniques that a PSSP administrator typically uses for daily maintenance and monitoring. We introduce some CSM tools that provide similar functionality, and discuss alternate methods of performing common tasks that have no direct equivalent in CSM.

PSSP is a tightly integrated set of tools that provides a single interface for all cluster operations. CSM is less tightly integrated and gives the administrator greater flexibility in choosing and implementing the underlying tools. In this chapter, we review some of the common tasks a PSSP administrator is familiar with and explore ways of accomplishing those tasks with CSM.

5.1 Node groupings

PSSP allows administrators to refer to a number of nodes with a single name through the concept of node groups.This provides a convenient way for administrators to perform tasks on subsets of nodes without specifying each individual host name. CSM also provides this capability, although the management and overall function of node groups is different.

Node groups in CSM are created and maintained with the **nodegrp** command. Nodes are added to node groups by specifying the **-a** flag on the **nodegrp** command. If the node group does not exist, it is created automatically. Example 5-1 shows the creation of a new static node group called p630s with three nodes (node9, node10, and node11). The second command adds node12 to the node group, and the third command lists the attributes and resulting members of the group.

Example 5-1 Creating a static node group

```
csmms> nodegrp -a node9, node10, node11 p630s
csmms> nodegrp -a node12 p630s
csmms> nodegrp -L p630s
p630s: static,validated,node9,node10,node11,node12
csmms>
```

The node group created in Example 5-1 illustrates the creation of a static node group. This type of node group contains only those members that are explicitly added through the command line. You can also define dynamic node groups where the members are determined at the time the node groups are referenced. To create a dynamic node group, specify attributes of the nodes to be included in the group using the **-w** flag instead of the **-a** flag on the **nodegrp** command. Example 5-2 shows the creation of a dynamic node group that consists of all machines with a hardware model of 9076.

Note that when you list the group with the **-L** flag, you only get the selection criteria. You can see the group members by running **nodegrp** with just the group name, which evaluates the condition at command execution time and lists all matching nodes, as shown in the third command.

Note: Dynamic node groups can only be defined with attributes that are persistent. For example, you cannot define a dynamic node group that includes *PowerStatus*, since that attribute is dynamic.

Example 5-2 Creating a dynamic node group

```
csmms> nodegrp -w "HWModel like '9076%'" Thin
csmms> nodegrp -L Thin
Thin: dynamic,(validated),HWModel like '9076%'
csmms> nodegrp Thin
node1
node3
node5
node7
node9
node10
```

5.2 CFM versus file collections

Configuration File Manager (CFM) provides functionality similar to that provided by PSSP File Collections to simplify the task of maintaining common files across multiple nodes. Both subsystems provide a means of collecting files on a central server and periodically distributing changed files to the managed nodes. Although PSSP File Collections and CFM provide similar functions, there are several important differences that you should be aware of.

The system architecture and philosophy of CFM are fundamentally different from PSSP File Collections. PSSP uses a number of configuration files to define which files and directories were contained within a file collection and how each file within the collection should be handled. In contrast, CFM uses a much simpler model to manage common files and uses no configuration files. PSSP File Collections use a client *pull* technology that relies on the clients to request updated files from the server. Files are only updated when requested by the client, and the client has the ability to refuse specific files. CFM uses a server *push* technology that allows the server to update all cluster nodes whenever necessary. Clients cannot refuse file updates.

Common files are maintained under the /cfmroot directory and copied to the nodes if the version on the management server is different from the copy on the node. To add a new file to the distribution, you simply copy it into the proper location under /cfmroot. For example, to update /etc/environment on all nodes in your cluster, you would copy it to /cfmroot/etc/environment. The file will be updated the next time CFM runs, which defaults to midnight, or you can force an immediate update to all nodes by typing **cfmupdatenode -a**. Updated files will be copied to the cluster nodes, preserving user and group ownership and file modification time.

Note: The owning UID and GID for a file must exist in /etc/passwd and /etc/group on the cluster nodes, or CFM will refuse to distribute the file.

You can control which files are distributed to particular nodes by specifying CSM node groups or RSCT node groups as part of the filename extension. For example, to distribute /etc/resolv.conf to your cluster nodes, you would copy the file to /cfmroot/etc/resolv.conf. If you want a different version of resolv.conf on the host firewall, you would copy your customized file to your CFM repository as resolv.conf._firewall, which will only get distributed to the specified node. You can distribute a different version of the file to all your Web servers by creating a CSM node group called webservers and creating resolv.conf._webservers in your CFM repository.

Note: More restrictive host specifications will take precedence over more general host specifications, so the local resolv.conf file on one of the webserver hosts will receive /cfmroot/resolv.conf._webservers instead of the more general /cfmroot/etc/resolv.conf.

The results of this configuration are shown in Example 5-3.

Example 5-3 CFM file segregation

```
resolv.conf -- distributed to all nodes except firewall and webservers
resolv.conf._firewall -- distributed only to node firewall
resolv.conf._webservers -- distributed to all nodes in RSCT group webservers
```

PSSP classified file collections as being *resident* or *available* to differentiate files that were installed on the server from files that were simply available from the server. You can obtain similar behavior with CFM by using symbolic links. When CFM encounters a symbolic link to a file, it copies the contents of the target file to the node instead of the link itself. You can use this method to ensure that your cluster nodes have the same files as your management server. To maintain one version of a file on your management server and distribute a different version to your cluster nodes, you can place the modified copy of the actual file under /cfmroot instead of a symbolic link.

CFM also provides the ability to perform actions before and after a file is updated through the use of .pre and .post files. If a file distributed by CFM has a corresponding executable named filename.pre. This file will be executed before the file is distributed to the node. Similarly, if filename.post exists, it will be executed after the file is distributed to the node. These files can be any executable file, including shell scripts or compiled code.

There are some additional tasks you can perform with CFM that are not easily done with PSSP File Collections. You can perform a *dry run* of CFM to obtain a list of files that need to be updated by using the **-q** flag with **cfmupdatenode**. You can get a summary of just the node names that have out-of-date files by using the **-s** flag in addition to the **-q** flag.

For assistance in debugging CFM operations, errors are logged to /var/log/csm/cfmerror.log. The operations performed by CFM are stored in **/**tmp/cfm_distfile, and a log of changed files and dates is stored in /var/log/csm/cfmchange.log.

Table 5-1 summarizes some of the differences between PSSP File Collections and CFM.

Table 5-1 File Collections vs. CFM (default settings)

	PSSP File Collections	**CSM CFM**
Configuration commands	PSSP commands	Standard AIX commands
Node control	Can refuse files	No ability to refuse files
Default update interval	Hourly	Daily
Predefined common files	4 file collections	None
Owner	Privileged user supman	root
Repository location	Varies	/cfmroot
Configuration files	/var/sysman/sup	None
Log files	/var/adm/SPlogs/filec/*	/var/log/csm/cfm*

5.3 User management

By default, management of user accounts and passwords in a PSSP environment is handled by PSSP File Collections. The user.admin file collection defines all the files necessary to share user authentication information among all nodes in a PSSP installation. PSSP is also NIS-aware. For PSSP environments using NIS, the PSSP software references the maps available from the NIS master.

In this section, we discuss using CFM to perform the same functions provided by the user.admin file collection in PSSP. We also discuss alternate methods of user management and accessibility of home directories across the cluster.

5.3.1 User management with CFM

CFM can be used to manage user accounts in the same manner that the PSSP user.admin file collection manages user accounts. By using symbolic links to add the password and group files to the CFM repository, user accounts can be replicated to all nodes in the cluster. The steps required to set this up are shown in Example 5-4. If you use indexed password files, you can distribute the .idx files using the same method of symbolic links, or you could write a .post script to have CFM run **/usr/sbin/mkpasswd** after the initial files have been distributed.

*Example 5-4 Using CFM to duplicate **user.admin** behavior*

```
ln -s /etc/passwd /cfmroot/etc/passwd
ln -s /etc/group /cfmroot/etc/group
ln -s /etc/security/passwd /cfmroot/etc/security/passwd
ln -s /etc/security/group /cfm/etc/security/group
```

You can also keep user accounts on your cluster nodes separate from the accounts defined on your management server. To accomplish this, you would copy the actual password and group files into the CFM repository instead of creating symbolic links. This method would require additional work to define where the master files reside, and copy them to the management server before running CFM.

Either of these approaches has the same problems associated with the user.admin file collection under PSSP, namely that users can only change their password on a single server that contains the master copy of the password files, and changes will not be visible to all nodes until the next distributed file update occurs.

5.3.2 User management with NIS

Network Information Services (NIS) is an alternative to copying password files to all the cluster nodes. Using NIS as a password management scheme for your cluster has several advantages: Users can change their password on any node, password updates are available immediately on all cluster nodes, and user information can be shared with machines outside your cluster or machines running different architectures.

There are also several disadvantages that should be considered. Using NIS requires that you set up and maintain an NIS master to manage NIS maps. Network connectivity problems between your NIS master and a client may keep users from authenticating to a host. Encrypted passwords are easily obtainable, and accounts can be compromised through brute force password cracking.

For a more complete discussion on the issues associated with implementing and maintaining NIS, see one of the many available NIS references.

5.3.3 User management with LDAP

Lightweight Directory Access Protocol (LDAP) is another mechanism for maintaining user authentication information from a central server, which is gaining widespread acceptance and popularity. LDAP provides the same advantages as using NIS and offers additional functionality. LDAP also suffers from some of the disadvantages of using NIS. AIX 5.2 ships with a standards-compliant LDAP implementation that can be used to provide user account management.

For more complete details on implementing and managing user accounts with LDAP, see *Secure Network Implementations on AIX - VPN, LDAP, PAM,* SG24-6066, *AIX Security Guide,* SC23-4850*,* or one of the many other LDAP references available. We also provide an example of LDAP configuration using NIM in C.4, “Using NIM to install and configure LDAP clients” on page 223.

5.3.4 Home directory access

Methods of providing access to users’ home directories vary greatly depending on the needs of your particular cluster. The PSSP user management commands (**spmkuser, spchuser**) were integrated with automount and would automatically update the automount maps with the changed user information.

CSM does not provide any integrated support for shared directory access. If you need to provide user access to home directories across multiple cluster nodes, you will need to manage the method yourself.

AIX offers several methods for providing shared access to user directories. Among these are Network File System (NFS), which is available for many different platforms with or without automount as used by PSSP, and General Parallel File System (GPFS), which is only available within the cluster. Each of these methods has distinct advantages and disadvantages that must be taken into consideration. The requirements of your particular environment and capabilities offered by each method will determine the strategy you select.

5.4 Cluster startup and shutdown

PSSP provided scripts to manage the startup and shutdown of the cluster in a controlled manner, allowing dependencies to be defined so that *nodex* would not start until *nodey* was already running. If no dependencies were defined by the

administrator, the boot install servers would be started before the other nodes. The default for shutdown was to have all nodes shut down at the same time.

When the shutdown process was complete, the nodes were powered off; the startup process powered the nodes on to restart them.

No equivalent function is supplied with CSM, so if there is a need to sequence the startup and shutdown of the managed nodes, it must be handled by the administrator. See Appendix B, “Cluster startup and shutdown” on page 193 for a sample startup/shutdown script.

There is a chapter dedicated to cshutdown and cstartup in *PSSP Administration Guide,* SA22-7348, which describes all the control files, and features, of the commands.

Table 5-2 and Table 5-3 on page 89 outline the basic operations performed by the **cshutdown** and **cstartup** commands with suggestions of how these tasks could be performed in a CSM environment.

Table 5-2 cshutdown processing phases vs. CSM/AIX

PSSP cshutdown phase	Suggested CSM/AIX approach
Notifying users and terminating nonroot processes	Code **/etc/rc.shutdown** to perform this processing.
Subsystem shutdown	
Node shutdown	Issue **shutdown -F** command on node, this can be scheduled using the **at** command on a **dsh** call. **dsh -n nodex** **‘echo “shutdown -F”|at now’**
Restarting nodes, if requested, after detecting power off	Power on the node using **rpower -n nodex on** command, this assumes hardware control is available. Power off can be detected when PowerStatus attribute in the IBM.ManagedNode RSCT class for the node is set to 0 (off).

Table 5-3 cstartup processing phases vs. CSM/AIX

PSSP cstartup phase	Suggested CSM/AIX approach
Start the node	Power on the node using **rpower**.
Wait until host_responds is set	Wait until the Status attribute in the IBM.ManagedNode RSCT class for the node is set to 1 (Alive).
Start the defined subsystems	Have entries in /etc/inittab or definitions in /etc/rc.d/rc2.d to start the subsystems.

We did some scripting based around ERRM that allows a script to monitor the CSM equivalents of host_responds (Status) and nodePower (PowerStatus) to determine when a node has shut down and when it has restarted. The scripts are in Appendix B, “Cluster startup and shutdown” on page 193.

5.5 Software maintenance

In this section, we discuss software maintenance in a CSM environment. We look specifically at installing and updating software products on your cluster nodes.

There are two common methods for installing and updating software in a CSM cluster. NIM can be used to do this by creating or updating an lpp_source and using NIM operations to update your nodes. You can also use the .pre and .post file scripting capabilities of CFM to update software. This method is preferable when your software packages are not in a format compatible with NIM.

Note: Whichever method you use, you should ensure that any prerequisite software is already installed, or available to be installed as part of the install/update process.

5.5.1 Maintaining software with NIM

The preferred method for installing or updating software is to use the capabilities provided by NIM. We recommend that you keep your lpp_source up to date with the current software and fixes installed on your cluster. This will enable you to install a node at the same software level as the rest of your cluster nodes, or to add a previously installed node and update all the installed software using a single source. There are times when it is more convenient to create a separate installp_bundle resource, however.

As an example, consider an existing cluster that does not have the LDAP software installed. You can install LDAP on all your nodes by creating an

installp_bundle NIM resource with the required software and running a NIM *cust* operation on all of the cluster nodes. You can also allocate script resources to the target nodes to configure the software as part of the installation process. The details of this procedure are documented as an example in C.4, “Using NIM to install and configure LDAP clients” on page 223.

NIM can also be used to perform operations such as committing, rejecting, or deinstalling software packages using the maint operation. For complete details on using NIM, see *Network Installation Guide and Reference,* SC23-4385 or the redbook *NIM: A to Z in AIX 4.3,* SG24-5524.

5.5.2 Maintaining software with CFM

CFM can be used to perform software package installation and upgrades by using the .pre and .post script capabilities. The general procedure for installing a new software package using CFM would be to write a .post script that installs the software, and place it in the /cfmroot directory along with the software package itself. When CFM updates the node, the new package will be copied to the node and the .post installation script will complete the software installation. Software packages can be upgraded in a similar manner. You can use the .pre script capabilities to perform any necessary subsystem shutdown before upgrading, and use a .post script to upgrade and restart the subsystem.

As an example, consider a situation where you need to upgrade the SSH server on your software nodes. You could create a .pre script that shuts down the sshd subsystem by calling **`stopsrc -s sshd`** before performing the upgrade. The file distributed to the node would contain the updated SSH packages, and the .post script would install the software and restart the subsystem with **`startsrc -s sshd`**.

5.5.3 Using dsh

You can also use **`dsh`** to mount a directory with the installp filesets and/or RPM packages (from the management server or elsewhere), and then to run installp or rpm locally on the nodes. This is often more convenient than using NIM or CFM, but also is more error prone because it is a more manual process.

5.6 Backups

Backups are a crucial part of system administration. The general topic of backups is beyond the scope of this book, so this section focuses on backing up CSM-specific data.

There are two common methods that come to mind for backing up your management server. The first is making a full mksysb of the management server, and the second is only backing up the CSM data. Each of these methods has unique concerns that you should be aware of.

Important: The systemid files might not work properly after a restore. The permissions of the restored system_config directory must be rw by owner only, or the systemid command won't write out new files. Also, if the .key file was restored, it may or may not be valid on the restored system.

5.6.1 mksysb backups

When you create a mksysb backup of your management server, you are creating a bootable copy of your root volume group. If you use this method, you must ensure that all your CSM data is contained in your root volume group, or make additional backups to capture that data. Note that /csminstall may be in a different volume group.

When restoring your management server from an mksysb, there is an unique concern if you restore the image on a different host. RSCT verifies the CPU identifier of the host it is running on, and will create a new cluster identifier if the CPU identifier has changed. This means that you cannot use an mksysb to simply move your management server to a new machine. If you restore to the same host, your mksysb should work correctly.

5.6.2 CSM data backups

You should periodically back up your CSM data independent of any mksysb images you make. This will provide the ability to quickly restore your CSM data without restoring your entire system's image. This is particularly important before major upgrades, or before performing operations that may potentially corrupt your CSM data. There are two methods for backing up your CSM data.

The first method is to back up all CSM-related data and directories with a standard archive command such as **tar** or **cpio**. You should shut down RMC before starting your backup. Example 5-5 shows the steps required to back up your data, as well as the directories that contain CSM data.

Example 5-5 Script to back up CSM data

```
#!/bin/ksh
#Make backups relative to root directory
cd /
#Shut down RMC
/usr/sbin/rsct/bin/rmcctrl -z
```

```
#Back up all configuration and data directories
/usr/bin/tar -cf /backups/csmdata.tar var/ct var/opt/csm etc/opt/csm \
etc/opt/conserver csminstall cfmroot opt/csm
#Restart RMC
/usr/sbin/rsct/bin/rmcctrl -s
```

The script in Example 5-5 produces a tar file with your CSM configuration information that can be restored relative to the root directory. To restore your CSM configuration, you need to shut down RMC. The shell script in Example 5-6 on page 92 shows how to restore the tar file created with the script in Example 5-5.

Example 5-6 Script to restore your CSM data

```
#!/bin/ksh
#Restore data relative to the root directory
cd /
#Shut down RMC
/usr/sbin/rsct/bin/rmcctrl -z
#Restore CSM data
/usr/bin/tar -xpf /backups/csmdata.tar
#Restart RMC
/usr/sbin/rsct/bin/rmcctrl -s
```

Note that this method only works if the data is restored to the same machine from which it was originally saved. If you try to restore this data on a different management server, you will encounter the same problems that show up when restoring an mksysb to a different machine.

The second method for backing up CSM-related data is to dump all your configuration information into ASCII files. The advantage of this method is that the data can be restored on a different management server. However, you will lose some information, such as historical event data.

Backing up data in ASCII format requires dumping out all node definitions, group definitions, events, responses, and ERRM associations. Since events and responses can be defined using other resources, we recommend that you back up all resource definitions. The shell script in Example 5-7 outlines the procedure for backing up your configuration using this method.

Example 5-7 Script to back up CSM data in ASCII format

```
#!/bin/ksh
#Back up CSM configuration data in /backups/csmdata
#
#Back up node definitions
/opt/csm/bin/lsnode -F > /backups/csmdata/nodedefs.bak
```

```
#Back up node group definitions
/opt/csm/bin/nodegrp -L > /backups/csmdata/nodegroups.bak
#Back up all ERRM resources
for rsrc in ‘/usr/bin/lsrsrc | tr -d ‘””‘ ; do
   /usr/bin/lsrsrc -i $rsrc > /backups/csmdata/$rsrc.bak
done
#Back up conditions/response associations
/usr/bin/lscondresp -lx > /backups/csmdata/condresp.bak
```

Restoring your CSM configuration is a little more complicated. Node definitions and node groups can be restored directly from the backup files as shown in Example 5-8 on page 93.

Example 5-8 Restoring CSM configuration files

```
#Restore node definitions
/opt/csm/bin/definenode -f > /backups/csmdata/nodedefs.bak
#Restore node groups
/opt/csm/bin/nodegrp -f > /backups/csmdata/nodegroups.bak
```

Note that you may have nodes in a PreManaged state. You can see this by running **/opt/csm/bin/lsnode -l <nodename>** and looking for InstallStatus. If you have nodes in this state, you can run **/opt/csm/bin/updatenode -P** to get them back to a Manage state. Depending on your particular situation, you may also need to update the security keys for the nodes. If you cannot run remote commands on your nodes with dsh, you can refresh the RSCT public keys with **/opt/csm/bin/updatenode -k <nodename>** for each of your nodes.

To restore conditions and responses, you must first remove the existing resources, and then add your backed-up resources in their place. For each resource, you can remove the resource using /usr/bin/rmrsrc and restore the definition with /usr/bin/mkrsrc. An example using the resource IBM.HwCtrlPoint is shown in Example 5-9.

Example 5-9 IBM.HwCtrlPoint resource

```
#Remove the existing resource
/usr/bin/rmrsrc -s “Name like ‘&’” IBM.HwCtrlPoint
#Restore the backed up definition
/usr/bin/mkrsrc -f /backups/csmdata/IBM.HwCtrlPoint.bak
```

There is no direct way to restore the associations between conditions and their responses. If you have a large number of associations it may be worthwhile to write a small script that parses the backup file and automatically recreates them. The backup file lists one association on each line with four fields. The first field is the name of the condition, followed by the response to the condition, and then the

node where the condition is defined. The fourth field shows whether the condition is being monitored.

The procedure is the same whether you restore the associations by hand or use a script. For each association in your backup file, you recreate the association with the **/usr/bin/mkcondresp** command. For associations that have an Active state (the fourth field in the backup file is “Active”), you restart the monitoring with the **/usr/bin/startcondresp** command.

5.6.3 Recovery tips

Even with the best intentions for backups, we are sometimes faced with catastrophic failures for which no solution seems to work. You can reset your RMC configuration with the **/usr/sbin/rsct/install/bin/recfgct** command. Note that this removes all your node data and starts you with a clean configuration. If you have backed up your CSM data into ASCII files, you can rebuild your cluster with the steps outlined in 5.6.2, “CSM data backups” on page 91.

When rebuilding a management server, you may get into a state where you cannot bring a node into the cluster. We have seen this problem when the node has AllowedManageRequest set to zero (0). You can use the **chnode** command to reset this attribute with **/opt/csm/bin/chnode -n <nodename> AllowManageRequest=1**.

Depending on the security method you use for your cluster, you may encounter problems with public keys getting out of synchronization. This may happen when a node gets reinstalled. You can update the public keys for a node by running **/opt/csm/bin/updatenode -k <nodename>**. Note that you should have a trusted network before running this command to prevent another machine from inserting a public key in place of the real node.

5.7 Accounting

PSSP provided scripts to enhance the basic AIX accounting and allow accounting on all nodes to be collected on a central accounting master. Further scripts were provided that consolidated the accounting information from all the nodes and produced global accounting reports.

CSM does not provide support for cluster-wide accounting. You can generate the same style of reports under CSM with a few commands to enable accounting and scripts to process the collected data. Generating cluster-wide accounting reports involves three steps:

1. Enabling accounting on all nodes
2. Collecting accounting data from cluster nodes on a single host
3. Processing the accounting data and generate a consolidated report

The first step only has to be performed once and can easily be accomplished with the dsh commands shown in Example 5-10 on page 95. Note that the accounting files have to exist before accounting can be enabled, so we run the **nulladm** command, which creates the required files with the proper permissions.

Note: There are a number of files and directories that must be set up before you use the standard AIX accounting scripts. For full details on setting up accounting, see the System Accounting section in *Systems Management Guide: Operating System and Devices,* SC23-4126.

Example 5-10 Enabling accounting in your cluster

```
dsh -N AIXNodes /usr/sbin/acct/nulladm /var/adm/pacct /var/adm/wtmp
dsh -N AIXNodes su - adm -c mkdir -m 0700 /var/adm/acct/nite \
/var/adm/acct/fiscal
dsh -N AIXNodes su - adm -c mkdir -m 0700 /var/adm/acct/sum
dsh -N AIXNodes /usr/sbin/acct/accton /var/adm/pacct
```

Once accounting has been enabled on your nodes, you need to set up cron jobs to process the accounting data for each node. For our testing, we added the standard **/usr/sbin/acct/runacct** script to the root crontab on each node.

After you have accounting enabled on your nodes, and the nightly scripts processing the data, you need to collect all your accounting reports on a central machine for processing. For our testing, we used **rcp** to collect the accounting files, but you could also use **scp** for secure remote copy, or the publicly available **dcp**, which works more like the PSSP **pcp** command.

Tip: The **dcp** script is available from http://www.alphaworks.ibm.com/ as part of the *ECT for Linux* package. If you convert the package to **cpio** format with **rpm2cpio**, you can use **cpio** to extract the **dcp** script.

The small script in Example 5-11 shows a simple way of collecting all your accounting records on your management server. This will copy accounting reports for the current day from each machine to /tmp/acct/MMdd.

Example 5-11 Collecting accounting records

```
#!/bin/ksh
repdate=‘date +%m%d‘
```

```
mkdir -f -m 0700 /tmp/acct/$repdate
cd /tmp/acct/$repdate
for node in ‘lsnode -N AIXNodes‘
do
   rcp $node:/var/adm/acct/sum/rprt$repdate rprt$repdate.$node
done
```

After your accounting records are collected in a common location, you can process the records from each node to produce an overall report, extract only certain accounting information, or store the information in a database.

5.8 Remote command execution

In this section, we discuss running parallel commands across your cluster or subsets of nodes in your cluster. We look at two tools provided with CSM, **dsh** and **dcem**.

5.8.1 dsh

The **dsh** command has been enhanced with additional features in CSM. The functionality of **dsh** is mostly backwards compatible with the **dsh** command from PSSP, so scripts and techniques that you are used to using should work without change. Some of the differences and enhancements of **dsh** that you should be aware of are as follows:

- **dsh** now uses DSH_LIST instead of DSHLIST to define the working collective of nodes, in addition to WCOLL. Both environment variables can be set to a file name that lists the nodes where the command should be run. If both DSH_LIST and WCOLL are set, DSH_LIST will take priority over WCOLL. Command line node specifications will override the environment variables. Node specification priorities, from highest to lowest, are the **-n** flag, the **-w** flag, the DSH_LIST environment variable, and finally the WCOLL environment variable.
- The path that is used to resolve remote commands can be specified through the environment variable DSH_PATH. If DSH_PATH is not specified, the default PATH on the remote system is used.

The enhanced **dsh** also provides some new options to further refine how remote commands are executed. Some of the command line options to **dsh** have also changed. A summary of the changes follows:

- The **-m** command line option has been added to print a message for each node when the remote command has started and when the command has ended.

- The **-r** command line option has been added to specify the full path of the remote shell used to connect to the remote system. If not specified, the shell specified in the **csmconfig** RemoteShell attribute is used.
- The **-s** command line option has been added to allow the output from each node to be printed as soon as possible. This may cause the output from different nodes to be mixed together, but may increase performance and decrease memory utilization.
- The **-S** command line option has been added to specify the remote shell syntax to use for setting environment variables. This can be set to `csh` or `ksh`, which will use `setenv` or `export`, respectively, to set remote environment variables.
- The **-t** command line option has been added to specify the number of seconds to wait before a remote command times out.
- The **-z** command line option has been added to print the return code of the last command that was run remotely. Check the documentation for caveats with OpenSSH.

The scheduling of remote shells to the managed nodes has also improved significantly. In PSSP, the **dsh** command stalls until all **rsh** invocations in one fanout batch have completed (or timed out), and only then starts the next fanout batch. The CSM version of **dsh** continuously spawns new **rsh** invocations as soon as a previous **rsh** returns. This is very valuable for large systems.

5.8.2 DCEM

The Distributed Command Execution Manager (DCEM) provides a graphical interface for running a command or script on multiple distributed machines in parallel. DCEM allows the administrator to save commonly used command options and recall them when needed. Saved command specifications are saved as Perl scripts in $HOME/dcem/scripts/myscripts and can be run from the command line. DCEM keeps a log of all distributed command activity and command output under $HOME/dcem/logs and $HOME/dcem/reports, respectively.

Figure 5-1 on page 98 shows the initial DCEM screen with a sample command to check file system space across all AIX nodes.

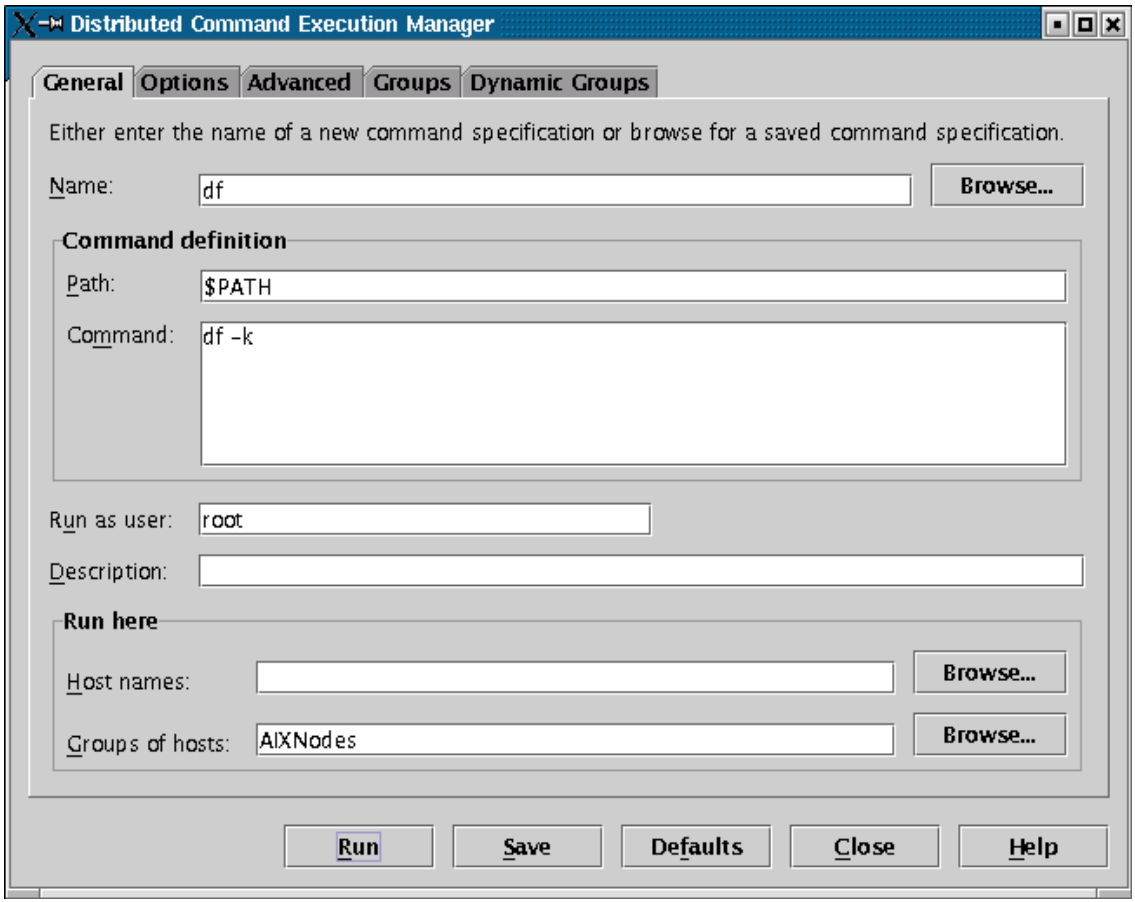


Figure 5-1 Initial DCEM screen

When you click **Run**, DCEM begins running the command on all specified nodes in parallel. In our example, there are three nodes in the AIXNodes group that we have entered, but only one node is running. The output from DCEM in this configuration is shown in Figure 5-2 on page 99.

The top half of the screen provides a real-time display of the command execution; as the command is run on each node, the node name moves from the Waiting column to the Working column, and finally to either the Successful or Failed column when the command is complete. In our example you can see that one node completed successfully and each of the nodes that is not running generated an error. By clicking the name of the node in the Successful or Failed columns, you can view the output from the command, or the error message, in the Results window by clicking the **Output** tab.

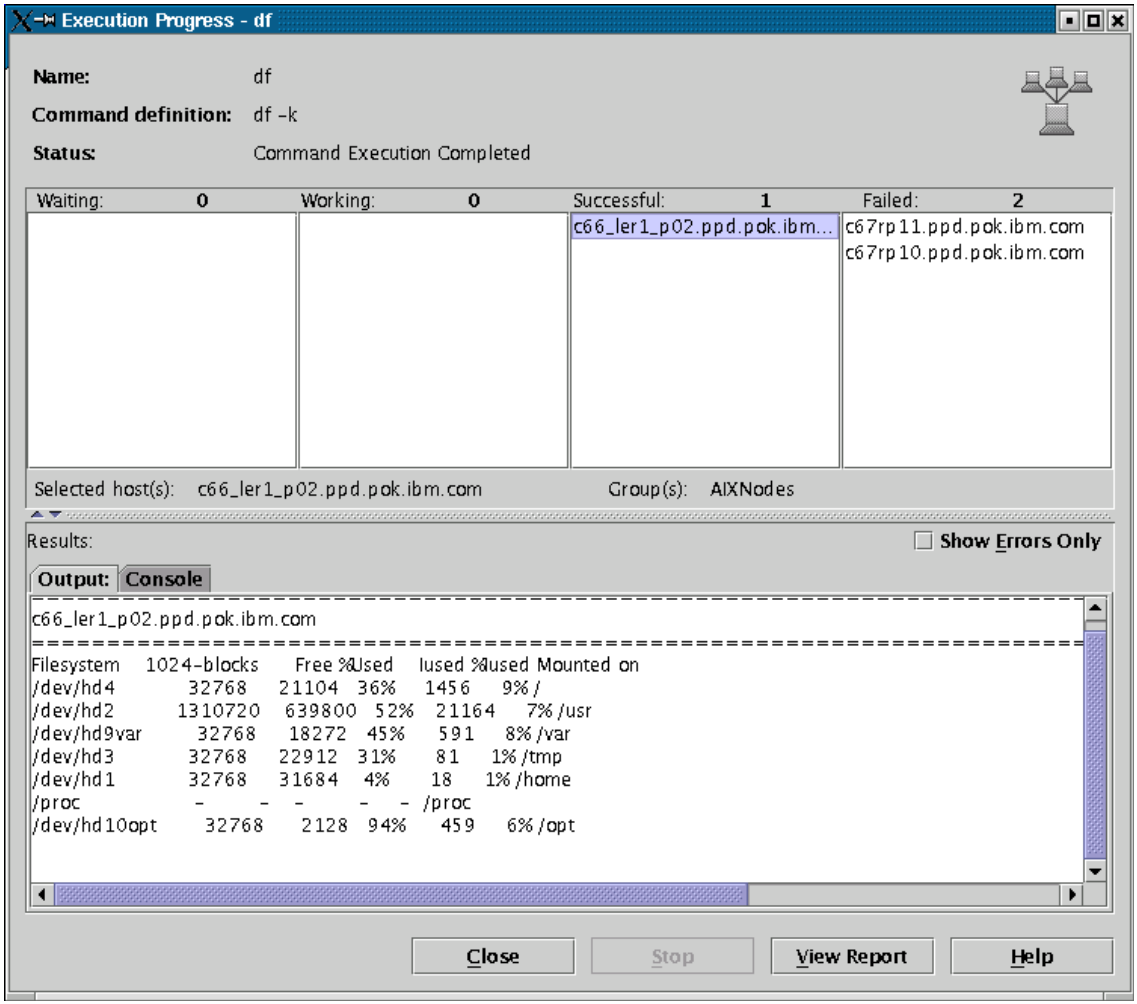


Figure 5-2 DCEM output

Note: DCEM stores information in a user's home directory under $HOME/dcem/. This can have serious consequences for root. Make sure you frequently check the amount of free space available in your root file system to avoid running out of space. Alternatively, you may want to create a symbolic link, /dcem, that points to a directory that is not in the root file system.

5.9 Hardware control

In this section, we summarize the hardware control features of PSSP and compare them with CSM hardware control.

5.9.1 Hardware control in PSSP

Hardware control in PSSP is strongly influenced by the original SP model: The control workstation is connected to a frame supervisor card through a serial RS232 connection, and the frame supervisor is connected to a node supervisor card in each node in the frame as well as to the (optional) SP switch board. The connectors on the cable that daisy-chains the nodes to the frame supervisor are hard-coded with the slot numbers of the SP frame. This allows the frame supervisor to detect which of the 17 slots are populated, and to target commands to specific slots. This serial connection is used to access the consoles of the nodes as well as for hardware monitoring and control. For each SP frame there is a separate serial connection with a separate TTY device configured on it (requiring multiport adapters on the CWS for large systems).

On the control workstation, the hardmon daemon controls the TTY devices that correspond to the serial connections to the frames, and communicates with the frame supervisors using the hardmon protocol (an SLIP protocol). For monitoring the hardware state, the daemon polls the frames every 5 seconds. It accepts client connections through TCP on a port defined in the SDR. Access control is implemented by integrating hardmon with the SP Security Services for authentication, and a hardmon-specific access control list file for authorization. These ACLs allow separate access control for the console, monitoring, and control actions. See figure Figure 5-3 on page 101 for a pictorial description of the SP hardware control in PSSP.

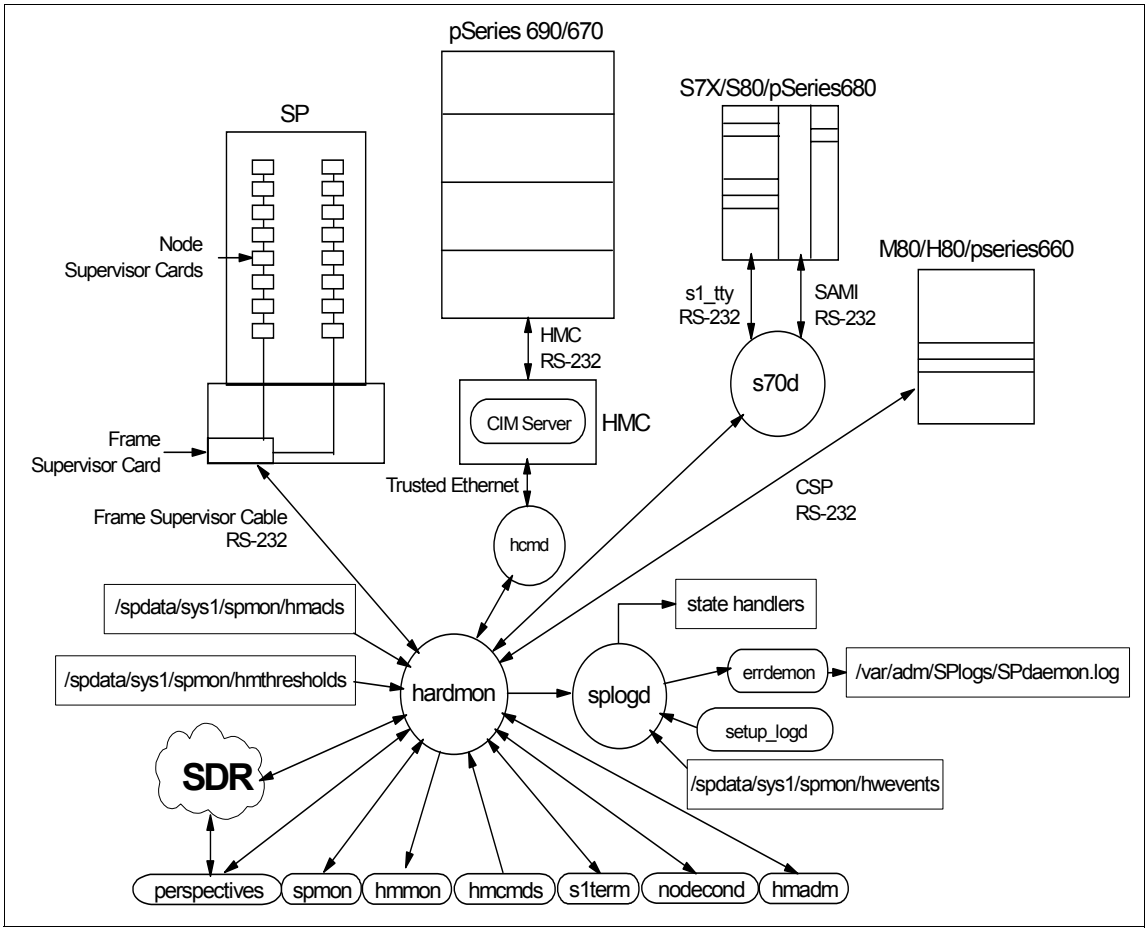


Figure 5-3 SP hardware control in PSSP

The following PSSP commands are the main hardmon clients:

- **`hmadm`**: Controls hardmon debugging and ACL list (re)loading
- **`hmmon`**: Monitors environmental variables and power status
- **`hmcmds`**: Hardware control such as power on/off
- **`spmon`**: An older SP command providing both monitoring and control. The

  ```
  spmon -d -G
  ```

  command is still very popular with PSSP system administrators to get a concise summary of the whole cluster's hardware state.
- **`s1term`**: Console access (one read/write session, plus read-only sessions)

- **nodecond**: Acquire adapter MAC addresses and initiate network boots
- **perspectives**: The GUI to perform hardware monitoring and control actions

Finally, the splogd daemon on the CWS records hardware events (such as loss of power on a node) that the frame supervisors report to the hardmon daemon.

For clustered enterprise servers (CES), the original SP hardware control model is slightly modified. These servers do not have supervisor cards. To treat them as SP nodes (and frames):

- Different serial connections for console access and for the other hardware control functions are required, which connect to the console and service processor serial connectors of the server.
- hardmon starts a server-specific daemon that translates the hardmon protocol to the specific service processors of the attached systems.

For SP-attached servers like the p660, M80, and H80, hardmon uses a different protocol Converged Service Processor (CSP) to communicate with the adapter cards in those servers, using a serial connection.

For POWER4-based systems that are managed by a Hardware Management Console (HMC), hardmon starts a separate daemon that communicates with the HMC through an IP connection. Since access to the HMC requires a username and password, the PSSP commands **sphmcid** and **spdelhmcid** are provided to store these encrypted on the CWS.

For details about these various attachment methods, refer to section 6.4 of *IBM (e)server Cluster 1600 Managed by PSSP 3.5: What's New,* SG24-6617.

5.9.2 CSM hardware control architecture

CSM has hardware control functionality that is similar to that of PSSP, but the architecture of the hardware control subsystem differs in the following ways:

- The first conceptual difference is that hardware control in CSM is optional. While most system administrators will probably view centralized hardware control as one of the key features of a cluster management system, it is possible to add nodes to a CSM cluster without having hardware control (still using the other CSM features such as monitoring and distributed command execution through **dsh**). This is also very convenient for testing purposes.
- The second difference is that CSM uses the concept of a *hardware control point* (HCP). This is in line with the introduction of a hardware management console (HMC) for hardware management of pSeries machines. Rather than requiring that all cluster hardware be directly attached to the management server, CSM records the IP name of an HCP with each node definition.

Different HCPs can use different protocols, allowing for the support of HMC-controlled nodes as well as SP hardware, and even xSeries machines. This makes the architecture more modular, but also has its drawbacks because it adds more components to the picture, and is potentially vulnerable to attacks against the connection from the management server to the HCP.

The main point of control (comparable to the hardmon daemon in the PSSP world) is the RMC subsystem of RSCT with the IBM.HWCTRLRM resource manager added to it by CSM. Access control to all hardware resources (except the console, which is handled by the conserver) is done through RSCT's CtSec infrastructure and ACLs.

Note: In addition to conserver, console access control is provided by RMC for HMC and CSP nodes.

The hardware control resource manager starts hardware control daemons depending on the PowerMethod that the nodes have set up.

For each required *PowerMethod*, the hardware control resource manager IBM.HWCTRLRM loads a shared library that provides the hardware-specific functions.

To list the power_method libraries available on your management server, you can run the following command:

```
sp6cws root / > lslpp -w "*_power.so"
  File                                    Fileset              Type
-----------------------------------------------------------------------------
  /opt/csm/lib/libcsp_power.so            csm.server           File
  /opt/csm/lib/libhmc_power.so            csm.server           File
  /opt/csm/lib/libapc_power.so            csm.server           File
  /opt/csm/lib/libnetfinity_power.so      csm.server           File
  /opt/csm/lib/libxseries_power.so        csm.server           File
```

Note: The libapc_power.so library does not start any daemons.

As part of its initialization, the power_method library starts a hardware control daemon, which then communicates with the actual hardware.

- For HMC-connected nodes, the *hcdaemon* is started, which is a Java™ application communicating with the HMCs:

```
c5aix07 root / > pstree -s HWCTRL
-+- 00001 root /etc/init
 \-+= 06474 root /usr/sbin/srcmstr
   \-+= 04522 root [13]/usr/sbin/rsct/bin/IBM.HWCTRLRMd
     \--- 18596 root [22]/usr/java131/jre/bin/java -Dhcdaemon ...
```

- For support of SP hardware, the *cspd* daemon is started. It provides a hardmon-like functionality to the SP hardware while providing a CSM-style power_method interface to the hardware control resource manager.

```
sp6cws root / > pstree -s HWCTRL
-+- 00001 root /etc/init
 \-+= 14530 root /usr/sbin/srcmstr
   \-+= 38898 root [14]/usr/sbin/rsct/bin/IBM.HWCTRLRMd
     \--= 39072 root [3]/opt/csm/csmbin/cspd -r 5
```

The hardware control client commands invoke the RMC action PowerAction of the IBM.NodeHwCtrl resource class to perform the hardware control actions. Figure 5-4 shows the relationship between the components of the hardware control subsystem.

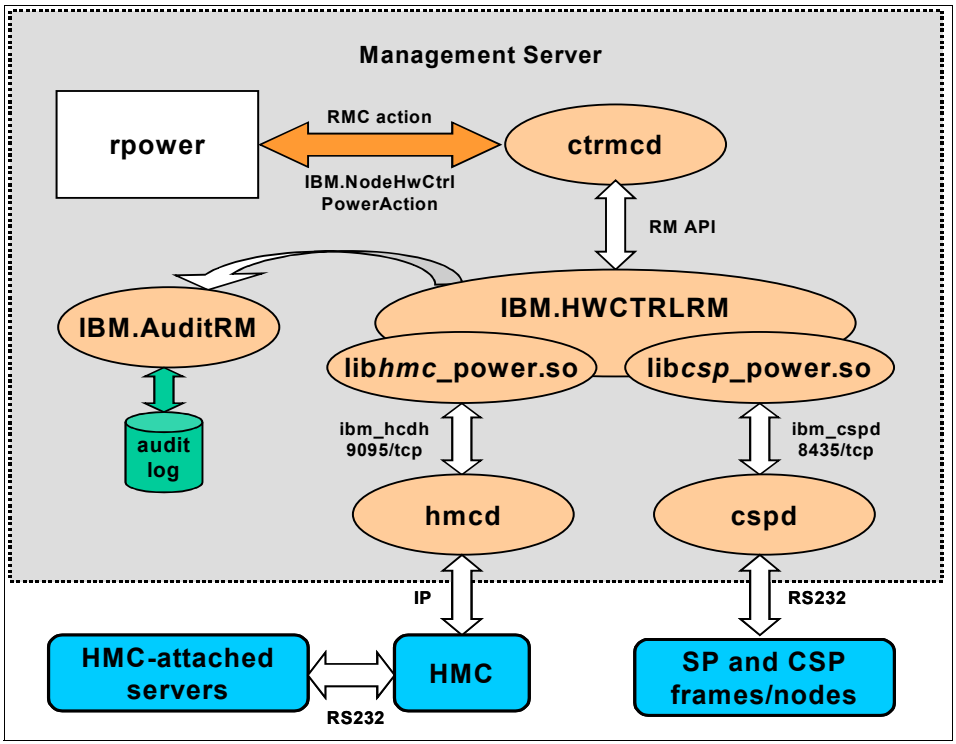


Figure 5-4 CSM hardware control subsystem

Finally, since CSM does not have PSSP's notion of frame numbers and node numbers, the CSM hardware control commands use nodenames (or nodegroup names) rather than a frame or slot specifier to designate the target nodes.

In the remainder of this section, we present the commands that exploit the CSM hardware control subsystem.

5.9.3 Remote console access in CSM (rconsole)

The equivalent to PSSP's **s1term** command is the **rconsole** command:

```
rconsole [-h] [-t] [-c | [-r|-f]] \
         { -a | -n node_list | -N node_groups }
```

The **rconsole** command uses the conserver package to provide multiple read-only console sessions in addition to one read/write session. It can also be asked to bypass conserver by specifying the **-c** option, but then it only provides a single read/write session.

There are some small differences in the default behavior of **rconsole** and **s1term** that you need to remember:

- Target nodes are denoted by **-a** for all nodes, or through a list of nodes or node groups using the **-n** or **-N** options, which are common across many CSM commands.

  Note: The **rconsole** window size is influenced by the number of consoles you open at the same time.

- **rconsole** by default starts a new terminal session, whereas **s1term** runs in the same terminal it was invoked in. Use the **-t** option to force **rconsole** to run in the invoking terminal session.
- By default, **rconsole** attempts to open a read/write session, and falls back to read-only if that fails. **s1term** will only start a read-write session when the **-w** flag is specified, and has no fallback to read-only mode. To start a read-only session, use the **-r** option.

Tip: The **rconsole** command uses different escape sequences than **s1term**. In particular, the sequence **^Ec**. (<ctrl><E><lowercase-c><period>) terminates the session. Use **^Ec?** to display the help screen:

```
[help]
 .    disconnect                          a    attach read/write
 b    send broadcast message              c    toggle flow control
 d    down a console                      e    change escape sequence
 f    force attach read/write             g    group info
 i    information dump                    L    toggle logging on/off
 l?   break sequence list                 l0   send break per config file
 l1-9 send specific break sequence        o    (re)open the tty and logfile
 p    replay the last 60 lines            r    replay the last 20 lines
 s    spy read only                       u    show host status
 v    show version info                   w    who is on this console
 x    show console baud info              z    suspend the connection
 <cr> ignore/abort command                ?    print this message
 ^R   replay the last line                \ooo send character by octal code
```

Note that **rconsole** will not be able to open a console if a virtual console (vterm) has been opened directly on an HMC. This behavior is similar to PSSP. That vterm cannot be overridden by **rconsole**'s **-f** option, which only forces consoles allocated through **rconsole** itself into read-only mode.

5.9.4 Remote power control in CSM (rpower)

CSM's **rpower** command only supports power on/off and various reset actions. It lacks the broad monitoring capabilities of hmmon.

```
rpower [-v] { -a | -n node_list | -N node_groups } \
       { on | off | reboot | query | resetsp_hcp | resetsp_host }
```

The actions that can be performed depend on the power_method of the node:

- On and off powers the node on and off, respectively. This should work for all power methods.
- **reboot** does a soft reset of the node if it is detected to be "on", and does a power-on if it is detected to be "off" (HMC and CSP nodes).
- resetsp_host sends the **reboot** command to the node supervisor (CSP nodes), or does a soft reset of the node (HMC).
- resetsp_hcp sends a **reboot** command to the frame supervisor on CSP nodes. This is not supported on the HMC nodes.

Also refer to the **cspmon** command below, which provides additional monitoring options (for CSP nodes only).

5.9.5 Additional CSP hardware control commands

APAR IY42379 adds some commands that parallel the PSSP hardware control commands. These include:

- **cspadm**
- **cspmon**
- **cspcmds** (does only power on/off of the frame, and microcode).

Please refer to the manpages of these commands for more details.

5.9.6 Network booting and the netboot command

The CSM **netboot** command provides the same functionality as the PSSP **nodecond** command.

Similar to PSSP, **netboot** is also called internally by commands that query a machine's network adapter MAC addresses by cycling through the service menus.

5.9.7 Getting whole-cluster summaries similar to spmon

In PSSP, the **spmon -d** command is a very useful tool to get a report of the cluster's hardware state. CSM does not offer a similar command. However, it is possible to get the information returned by **spmon** using a few CSM commands. We created the small *csmmon* script in Example 5-12 to create a summary report of a CSM cluster that is similar to the **spmon -d** command.

Example 5-12 csmmon script

```
#!/bin/ksh
#
# csmmon - poor man's "spmon -d"
#

echo "\n1. Checking server processes\n"
lssrc -s ctrmc        | tail +2
lssrc -s IBM.DMSRM    | tail +2
lssrc -s IBM.HWCTRLRM | tail +2

echo "\n2. Checking hardware control point(s)\n"
lsrsrc -t IBM.HwCtrlPoint Name PowerMethod \
 | tail +2
echo "\n`lsrsrc -tx IBM.HwCtrlPoint Name|wc -l` hardware control points"

echo "\n3. Checking nodes\n"
lsrsrc -t IBM.ManagedNode Name HWControlNodeId HWModel PowerStatus Status \
```

```
 | tail +2

echo
```

Example 5-13 is a sample output of the *csmmon* script when run on a CSM cluster with two nodes attached through two different HMCs.

Example 5-13 Sample output of the csmmon script

```
c5aix07 root / > csmmon

1. Checking server processes

 ctrmc            rsct             16770        active
 IBM.DMSRM        rsct_rm          5010         active
 IBM.HWCTRLRM     rsct_rm          4522         active

2. Checking hardware control point(s)

Name                      PowerMethod
"c66hmc.ppd.pok.ibm.com"  "hmc"
"c66hmc2.ppd.pok.ibm.com" "hmc"

       2 hardware control points

3. Checking nodes

Name                            HWControlNodeId HWModel    PowerStatus
Status
"c67rp10.ppd.pok.ibm.com"       "c67rp08"       "7040-681" 127         1
"c66_ler1_p01.ppd.pok.ibm.com" "c66_ler1_p01"  "7028-6C4" 1           1
```

On an SP with one frame, *csmmon* returns the output shown in Example 5-14.

Example 5-14 .csmmon script output after being run on an SP frame

```
sp6cws root / > csmmon

1. Checking server processes

 ctrmc            rsct             48486        active
 IBM.DMSRM        rsct_rm          15130        active
 IBM.HWCTRLRM     rsct_rm          55038        active

2. Checking hardware control point(s)

Name        PowerMethod
"/dev/tty1" "csp"
```

```
        1 hardware control points

   3. Checking nodes

   Name     HWControlNodeId HWModel    PowerStatus Status
   "node1"  "1"             "9076-WCN" 1           1
   "node3"  "3"             "9076-WCN" 1           1
   "node5"  "5"             "9076-260" 1           1
   "node7"  "7"             "9076-260" 1           1
   "node9"  "9"             "9076-WCN" 1           1
   "node10" "10"            "9076-WCN" 1           1
```

Although the output in Example 5-14 does not provide all the information that PSSP administrators are used to (in particular, no LED/LCD values), we found this useful for getting a quick overview of the cluster's status.

5.10 Microcode updates

Since the frame and node supervisors in an SP are a unique feature of the SP and have microcode loaded onto them, the PSSP software is also used to maintain that microcode:

- Frame, node, and SP Switch/SP Switch2 microcode is shipped as software PTFs, in the fileset ssp.ucode.
- The PSSP command **spsvrmgr** can be used to query the installed microcode level and compare it against the available levels installed on the control workstation.
- The PSSP command **hmcmds** is used to load microcode onto the cards. There are three command modes: basecode, microcode and boot_supervisor, which perform the actions to load the microcode and reboot the supervisors. Refer to *PSSP Command and Technical Reference (2 Volumes),* SA22-7351 for details.

With the introduction of support for a selected set of SP nodes in SP frames through APAR IY42379, CSM also provides a means to maintain the microcode in that SP hardware:

- Microcode for SP hardware is shipped as software PTFs and is included in the csm.server fileset. It will be installed into /csminstall/AIX/ucode/.
- You can use the following command to query the current microcode level:

```
cspmon -Qsv codeVersion { -C tty | -n nodename }
```

Note: The command gets a list of the microcode versions for all the hardware in the given list of frames, including frame and node supervisors. This command is not limited to just a single frame. Also, when the command is given in this fashion (without the -s flag), the information is shown in the same format that the ucode file names use. The `tty` argument is the full path of the device name of the frame serial connection. Also note that the returned data is different from the **spsvrmgr** output.

- Microcode uploads can be performed by the **cspcmds** command. It supports the basecode, microcode and boot_supervisor command modes, which match the corresponding **hmcmds** command modes. Refer to *CSM for AIX 5L: Command and Technical Reference,* SA22-7934 for details.

Note: Note that CSM does not support the SP Switch/SP Switch2, so there is neither a need nor support for switch microcode.

5.11 Additional administration tools (snap commands)

In addition to the **/usr/sbin/snap** command (from bos.rte.serviceaid), which gathers basic AIX system configuration information, PSSP provides some snap commands to aid problem determination for some of its subsystems. In particular:

- **/usr/lpp/ssp/css/css.snap** and **/usr/lpp/ssp/css/css.snap.corsair** (from ssp.css) for SP Switch and SP Switch2 data collection
- **/usr/lpp/ssp/bin/spd/spd.snap** (from ssp.css) for gathering switch logfiles from all the nodes to the CWS
- **/usr/sbin/rsct/bin/phoenix.snap** (from rsct.basic.rte) and **/usr/sbin/rsct/bin/ctsnap** (from rsct.core.utils) for RSCT data collection
- **/usr/lpp/csd/bin/vsd.snap** for VSD data collection

CSM does not support the SP Switch or SP Switch2, so the CSS-related **snap** commands no longer exist.

The RSCT **snap** commands can be used with CSM. In particular the **ctsnap** command can be useful for gathering RSCT data for problem determination.

6

Monitoring

This chapter discusses monitoring concepts and tools that Cluster Systems Management (CSM) provides in a distributed environment. Our goal is to help PSSP administrators who are experienced in event monitoring and problem management with PSSP understand the similarities and differences between monitoring under PSSP and monitoring under CSM.

We discuss various situations where monitoring and response are done under PSSP. We describe CSM commands that provide similar functionality to PSSP and provide scripts if there is no equivalent CSM command for a PSSP function.

Relevant references include:

- *RSCT for AIX 5L: Guide and Reference,* SA22-7889
- *RSCT for AIX 5L: Technical Reference,* SA22-7890
- *A Practical Guide for Resource Monitoring and Control (RMC),* SG24-6615
- *CSM for AIX 5L: Administration Guide,* SA22-7918

6.1 Architecture difference

The PSSP Event Management (EM) subsystem and the Problem Management (pman) subsystem provide an infrastructure for recognizing and acting on problem events within a domain. Each SP system partition is such a domain. The CWS is a member of each system partition and therefore, a member of each EM domain.

On each node or CWS, there is a copy of the EM daemon running for each domain the host belongs to. An EM client registers with its local EM daemon the interested events from a set of resources. The EM daemon can communicate with EM daemons on other nodes in order to provide events to its local client. Therefore, an EM client may register for and receive events about any resource in the PSSP cluster.

A resource monitor observes the state of the resources and transforms the state into resource variables to pass onto the EM daemon. The EM daemon compares the resource variable state with registered interests. If they match, an event is generated and sent to the appropriate EM client, as shown in Figure 6-1 on page 113.

Resource variables, resource monitors, and other related information are stored in the System Data Repository (SDR).

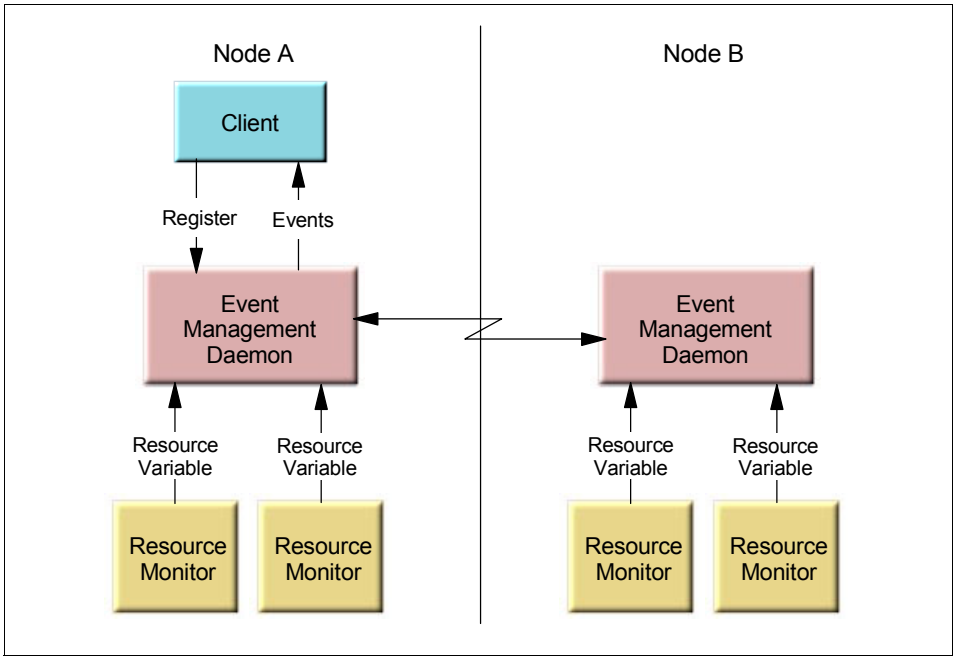


Figure 6-1 PSSP event management information flow

Resource Monitoring and Control (RMC) subsystem, part of Reliable Scalable Cluster Technology (RSCT), is responsible for event monitoring and problem management under CSM. There is a core set of resource classes and resource managers provided by RMC that are installed with AIX 5L. Additional resource classes and resource managers are provided by CSM for AIX.

RMC also works within its own domain. There are three different types of RMC domains: local domain, peer domain, and management domain. A host can be in different domains at the same time and only one copy of the RMC daemon is needed.

Under the CSM management domain, all managed nodes are considered in the management domain but the management server is not considered a member of the management domain. Events in the management domain can only be managed on the management server since there is no trust or knowledge among the managed nodes. Thus, a node cannot learn that an event happened on another node. A node only knows the, and the management server knows all managed nodes in its cluster.

The RMC subsystem acts as a broker between RMC clients that are interested in events and the resource managers that monitor the resources; see Figure 6-2 on page 114. A resource manager (RM) is a stand-alone daemon that monitors

specific types of resources and maps resource and resource class abstraction for the RMC daemon. Event response resource manager (IBM.ERRM) provides the ability to take actions in response to conditions occurring in the cluster.

Information related to resource class, attributes, event condition, and response is stored in the RSCT registry. See Chapter 4, “PSSP SDR vs. RSCT SR” on page 53 for a comparison between the SDR and the RSCT registry.

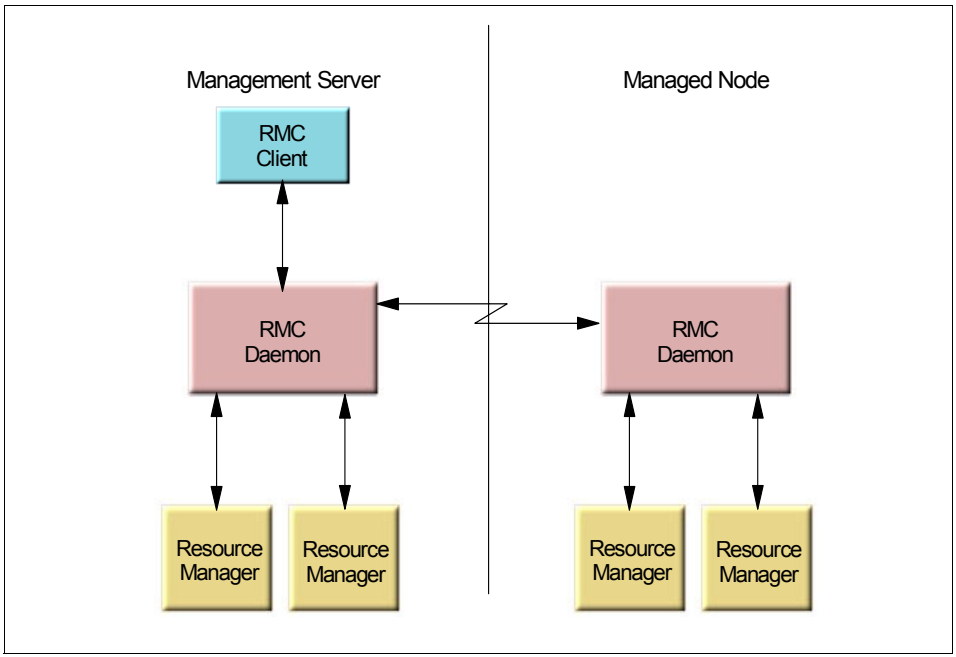


Figure 6-2 RMC event management information flow

6.2 RMC resource classes and manager

To list RMC resource classes, use the command **lsrsrc** as shown in Example 6-1.

Example 6-1 List RMC resource classes

```
$ lsrsrc -x | sort
"IBM.ATMDevice"
"IBM.Association"
"IBM.AuditLog"
"IBM.AuditLogTemplate"
"IBM.Condition"
"IBM.DRM"
```

```
   "IBM.DmsCtrl"
   "IBM.EthernetDevice"
   "IBM.EventResponse"
   "IBM.FDDIDevice"
   "IBM.FileSystem"
   "IBM.Host"
   "IBM.HostPublic"
   "IBM.HwCtrlPoint"
   "IBM.ManagedNode"
   "IBM.ManagementServer"
   "IBM.NetworkInterface"
   "IBM.NodeAuthenticate"
   "IBM.NodeGroup"
   "IBM.NodeHwCtrl"
   "IBM.PagingDevice"
   "IBM.PhysicalVolume"
   "IBM.PreManagedNode"
   "IBM.Processor"
   "IBM.Program"
   "IBM.Sensor"
   "IBM.ServiceEvent"
   "IBM.Sfp"
   "IBM.TokenRingDevice"
   "IBM.WLM"
```

Required classes for CSM include:

- IBM.ManagedNode
- IBM.NodeGroup
- IBM.NodeHWCtrl
- IBM.HwCtrlPoint
- IBM.Condition
- IBM.EventResponse
- IBM.Association

Example 6-2 shows a list of resource managers by querying the System Resource Controller (SRC).

Example 6-2 List resource managers known by the SRC manager

```
$ lssrc -g rsct_rm
   Subsystem         Group            PID          Status
    IBM.ERRM         rsct_rm          20722        active
    IBM.DMSRM        rsct_rm          25110        active
    IBM.ServiceRM    rsct_rm          15684        active
    IBM.HWCTRLRM     rsct_rm          19640        active
    IBM.CSMAgentRM   rsct_rm          10484        active
    IBM.AuditRM      rsct_rm          5492         active
    IBM.FSRM         rsct_rm          11726        active
```

```
    IBM.HostRM       rsct_rm          22720        active
    IBM.DRM          rsct_rm                       inoperative
    IBM.SensorRM     rsct_rm                       inoperative
    IBM.WLMRM        rsct_rm                       inoperative
    IBM.ConfigRM     rsct_rm                       inoperative
```

Example 6-3 lists resource managers known by the RMC daemon by querying the subsystem ctrmc.

Example 6-3 List resource managers known by the RMC daemon

```
$ lssrc -ls ctrmc | grep -p "Resource Manager Information"
   Resource Manager Information
            Name           ClassKey      ID         FD       SHMID
   IBM.HostRM                   1         0         29       1179651
   IBM.ERRM                     1         1         20            -1
   IBM.AuditRM                  1         2         23            -1
   IBM.DRM                      1         3         -1            -1
   IBM.DMSRM                    1         4         16            -1
   IBM.FSRM                     1         5         26       1572866
   IBM.HWCTRLRM                 1         6         21            -1
   IBM.CSMAgentRM               1         7         19            -1
   IBM.ConfigRM                 1         8         -1            -1
   IBM.SensorRM                 1         9         -1            -1
   IBM.ServiceRM                1        10         18        131076
   IBM.WLMRM                    1        11         -1            -1
```

Tip: You need the *rsct.core.sensorrm* fileset installed on your server to have the resource manager “IBM.SensorRM”, which you will need if you want to define your own monitors.

You can find out which manager is responsible for a particular resource class as shown in Example 6-4. Table 6-1 on page 117 lists all resource classes with their corresponding resource managers.

Example 6-4 List resource manager for a resource class

```
$ lsrsrcdef -c "IBM.FileSystem" | grep mgr_name

mgr_name = "IBM.FSRM”
```

Table 6-1 RMC resource class and corresponding manager

RMC resource class	RMC resource manager	Obsv Interval	Reporting interval
IBM.ATMDevice	IBM.HostRM	40	10
IBM.Association	IBM.ERRM		
IBM.AuditLog	IBM.AuditRM		
IBM.AuditLogTemplate	IBM.AuditRM		
IBM.Condition	IBM.ERRM		
IBM.DRM	IBM.DRM		
IBM.DmsCtrl [a]	IBM.DMSRM		
IBM.EthernetDevice	IBM.HostRM	40	10
IBM.EventResponse	IBM.ERRM		
IBM.FDDIDevice	IBM.HostRM	40	10
IBM.FileSystem	IBM.FSRM	60	60
IBM.Host	IBM.HostRM	[b]	[b]
IBM.HostPublic	IBM.HostRM		
IBM.HwCtrlPoint [a]	IBM.HWCTRLRM		
IBM.ManagedNode [a]	IBM.DMSRM		
IBM.ManagementServer [c]	IBM.CSMAgentRM		
IBM.NetworkInterface	IBM.ConfigRM		
IBM.NodeAuthenticate [a]	IBM.DMSRM		
IBM.NodeGroup [a]	IBM.DMSRM		
IBM.NodeHwCtrl [a]	IBM.HWCTRLRM		
IBM.PagingDevice	IBM.HostRM	30	10
IBM.PhysicalVolume	IBM.HostRM	30	10
IBM.PreManagedNode [a]	IBM.DMSRM		
IBM.Processor	IBM.HostRM	15	5
IBM.Program	IBM.HostRM		

Table 6-1 RMC resource class and corresponding manager

RMC resource class	RMC resource manager	Obsv Interval	Reporting interval
IBM.Sensor	IBM.SensorRM		
IBM.ServiceEvent	IBM.ServiceRM		
IBM.Sfp	IBM.ServiceRM		
IBM.TokenRingDevice	IBM.HostRM	40	10
IBM.WLM	IBM.WLMRM	10	10

a. IBM.DMSRM and IBM.HWCTRLRM are on the management server only.
b. Intervals vary depending on attributes: KMem* 15/5, *RealMem* 5/5, PctTotal Time* 5/5, VM* 5/5, *TotalPgSp* 10/10, Proc* 20/20, LoadAverage and NumUsers 15/15 and Uptime 60/60.
c. IBM.CSMAgentRM is on the managed nodes only.

Table 7 in *RSCT for AIX 5L: Guide and Reference,* SA22-7889 provides details on resource managers provided with RSCT.

Each resource class has a number of persistent and dynamic attributes associated with it, which can be listed by **lsrsrcdef -ap** *resource_class* and **lsrsrcdef -ad** *resource_class*, respectively, as shown in Example 6-5.

Example 6-5 Sample output of lsrsrcdef -t

```
$ lsrsrcdef -ad -t IBM.FileSystem | cut -c1-79

Resource Dynamic Attribute Definitions for: IBM.FileSystem
program_name       display_name group_name properties description attribute_id
"OpState"          ""           ""         {"public"} ""          1
"PercentTotUsed"   ""           ""         {"public"} ""          2
"PercentINodeUsed" ""           ""         {"public"} ""          3

$ lsrsrcdef -ap -t IBM.FileSystem | cut -c1-79

Resource Persistent Attribute Definitions for: IBM.FileSystem
program_name     display_name group_name properties
"Name"           ""           "" {"read_only","reqd_for_define","select
"MountPoint"     ""           "" {"read_only","inval_for_define","select
"MountDir"       ""           "" {"read_only","inval_for_define","select
"Dev"            ""           "" {"read_only","inval_for_define","select
"Vfs"            ""           "" {"read_only","inval_for_define","select
"Log"            ""           "" {"read_only","inval_for_define","select
"Mount"          ""           "" {"read_only","inval_for_define","select
"Permissions"    ""           "" {"read_only","inval_for_define","select
"Account"        ""           "" {"read_only","inval_for_define","select
```

```
"Type"           ""          "" {"read_only","inval_for_define","select
"Size"           ""          "" {"read_only","inval_for_define","select
"Frag"           ""          "" {"read_only","inval_for_define","select
"Nbpi"           ""          "" {"read_only","inval_for_define","select
"Compress"       ""          "" {"read_only","inval_for_define","select
"Bf"             ""          "" {"read_only","inval_for_define","select
"Ag"             ""          "" {"read_only","inval_for_define","select
"AgBlkSize"      ""          "" {"read_only","inval_for_define","select
"FsState"        ""          "" {"read_only","inval_for_define","select
"ErrorsBehavior" ""          "" {"read_only","inval_for_define","select
"ReservedBlock"  ""          "" {"read_only","inval_for_define","select
"MountCount"     ""          "" {"read_only","inval_for_define","select
"MaxMntCount"    ""          "" {"read_only","inval_for_define","select
"NoDev"          ""          "" {"read_only","inval_for_define","select
"NoSuid"         ""          "" {"read_only","inval_for_define","select
"DumpInterval"   ""          "" {"read_only","inval_for_define","select
"FsckPass"       ""          "" {"read_only","inval_for_define","select
"BlockSize"      ""          "" {"read_only","inval_for_define","select
"ManualMode"     ""          "" {"read_only","inval_for_define","public
"NodeNameList"   ""          "" {"read_only","option_for_define","select
```

Persistent attributes define the configuration characteristics of the resource and resource class. Examples of persistent attributes are Device Name, IP Address, and so on. For resource classes, persistent attributes describe or control the operations of the class. For the RMC client, persistent resources are mainly used to specify which resources will be accessed. They are used as filters over the set of managed resources to select those on which the client wants to act (**-s** parameter of the **mkcondition** command).

Dynamic attributes reflect internal states or performance variables of resources and resource classes. For example, all the file system resources have dynamic attributes, such as operational state, % total used, % inode used, and so on. You generally refer to dynamic attributes to define the monitoring condition of the resource you want to monitor (**-e** or **-E** parameter of the **mkcondition** command).

To see a detailed description of each attribute, use the command **lsrsrcdef -e resource_class attribute** (if the attribute is dynamic, you need to specify **-Ad**) as shown in Example 6-6.

Example 6-6 Sample output of lsrsrcdef -e

```
$ lsrsrcdef -e IBM.FileSystem ErrorsBehavior

Resource Persistent Attribute Definitions for: IBM.FileSystem
attribute 1:
        program_name  = "ErrorsBehavior"
        display_name  = "Behavior When Detecting Errors."
```

```
        group_name    = "Advanced"
        properties    = {"read_only","inval_for_define","selectable","public"}
        description   = "Behavior when errors are detected."
        attribute_id  = 22
        group_id      = 1
        data_type     = "char_ptr"
        variety_list  = {[3,3],[6,6]}
        variety_count = 2
        default_value = ""
```

$ lsrsrcdef -Ad -e IBM.FileSystem PercentINodeUsed

```
Resource Dynamic Attribute Definitions for: IBM.FileSystem
attribute 1:
        program_name           = "PercentINodeUsed"
        display_name           = "Percent I-Node Used"
        group_name             = "General"
        properties             = {"public"}
        description            = "This dynamic attribute gives the percentage
total I-Node used in a resource."
        attribute_id           = 3
        group_id               = 0
        data_type              = "int32"
        variable_type          = "quantity"
        variety_list           = {[1,6]}
        variety_count          = 1
        init_value             = 0
        min_value              = 0
        max_value              = 100
        expression             = "PercentINodeUsed > 90"
        expression_description = "An event will be generated when more than 90
percent of the total inodes in
the filesystem is in use."
        rearm_expression       = "PercentINodeUsed < 85"
        rearm_description      = "The event will be rearmed when the percent of
the inodes used in the
filesystem falls below 85 percent."
        PTX_name               = ""
```

Building monitors under RMC can then be summarized as:

1. Finding the right resource class with the right attributes as shown in Example 6-7. Table on page 123 shows the CSM resource class and its attributes.

Example 6-7 How to list all resources and their attributes

```
for rc in $(lsrsrc -x | sort) ; do
    for a in p d; do
   echo $rc:$a $(lsrsrcdef -A $a -t -x $rc | awk '{print $1}')
```

```
    done
done | tr -d '"'
```

2. Learning the details of the resource attribute:

   ```
   lsrsrcdef -A d|p -e Resource_class Attribute
   ```

3. Creating the monitor definition as show in 6.3, “Defining a monitor with predefined resource class” on page 130.

Table on page 123 is quite extensive. How can you find the right resource class and attribute that suits your needs? We created a little script, *rmchlp,* shown in Example 6-8, to help you find the right resource class and attributes.

Example 6-8 script rmchlp

```
#!/bin/ksh
function usage {
  echo "Usage: `basename $1`: find|use resource..."
  exit
}
function fnd {
  f=$(echo $* | /usr/bin/tr -s ' ' '|')
  for rc in $(/usr/bin/lsrsrc -x | /usr/bin/sort | /usr/bin/tr -d '"'); do
     /usr/bin/lsrsrcdef -Ad -t -x $rc |
        /usr/bin/awk -v rc=$rc '{print rc,$1}' |
        /usr/bin/grep -i -E "$f"
  done
}
function use {
  rc=$1;at=$2
  echo "##" $rc:$at
  /usr/bin/lsrsrcdef -Ad -e -p0 $rc $at
  echo "## selectable fields for $rc:"
  /usr/bin/lsrsrcdef -x -Ap -d -p0 $rc |
    awk -F':' '$4 ~ "selectable" {print $1}' |
    xargs -n 1 /usr/bin/lsrsrcdef -x -Ap -e -p0 $rc |
      /usr/bin/tail +2
}
[ $# -lt 2 ] && usage $0
case $1 in
  find) shift; fnd $*;;
  use) shift; use $*;;
  *) usage $0;;
esac
```

Example 6-9 on page 122 shows how rmchlp can be used to find out what resource class attribute may be available for monitoring “CPU wait”.

Example 6-9 Use rmchlp to find relevant resource class and attributes

```
$ rmchlp find cpu wait
IBM.Host "PctTotalTimeWait"
IBM.Processor "PctTimeWait"
IBM.WLM "CPUPct"
IBM.WLM "CPUAtSoftMax"
IBM.WLM "CPUAtHardMax"
```

If we think *IBM.Processor "PctTimeWait"* might be what we need, Example 6-10 shows how rmchlp can be used again to get detailed information on how the resource class attribute can be used.

Example 6-10 Use rmchlp to find detailed info on a resource class attribute

```
$ rmchlp use IBM.Processor "PctTimeWait"
## IBM.Processor:PctTimeWait
Resource Dynamic Attribute Definitions for: IBM.Processor
attribute 1:
        program_name           = "PctTimeWait"
        display_name           = "% Time Waiting"
        group_name             = "Advanced"
        properties             = {"public"}
        description            = "This attribute reflects the percentage of
time the processor is waiting.

<text deleted>

        attribute_id           = 3
        group_id               = 1
        data_type              = "float64"
        variable_type          = "quantity"
        variety_list           = {[1,1]}
        variety_count          = 1
        init_value             = 0
        min_value              = 0
        max_value              = 100
        expression             = "(PctTimeWait >= 50) && (PctTimeWait@P >= 50)"
        expression_description = "The example event expression causes the
generation of an event when
the average time the processor is in wait state is at least 50% for two
consecutive observations."
        rearm_expression       = "(PctTimeWait < 30) && (PctTimeWait@P < 30)"
        rearm_description      = "The example rearm expression reenables events
after an event has
been generated and the time in wait mode decreases below 30% for two
consecutive observations."
        PTX_name               = ""
## selectable fields for IBM.Processor:
```

```
        program_name  = "Name"
        display_name  = "Name"
        group_name    = "General"
        properties    = {"read_only","reqd_for_define","selectable","public"}
        description   = "Identifies the name of the processor such as 'proc1'."
        attribute_id  = 0
        group_id      = 0
        data_type     = "char_ptr"
        variety_list  = {[1,1]}
        variety_count = 1
        default_value = ""
attribute 1:
        program_name  = "ResourceHandle"
        display_name  = "Resource Handle"
        group_name    = "Internal"

<text deleted>
```

Table 6-2 shows the CSM resource class and its attributes.

Table 6-2 CSM resource class and attributes

Resource class	p/d [a]	Attributes
IBM.ATMDevice	p	Name NodeNameList
IBM.ATMDevice	d	RecErrorRate RecDropRate XmitDropRate XmitErrorRate XmitOverflowRate RecByteRate RecPacketRate XmitByteRate XmitPacketRate RecErrors RecDrops XmitDrops XmitErrors XmitOverflows RecBytes RecPackets XmitBytes XmitPackets
IBM.Association	p	Name ActiveFlag ConditionHandle EventResponseHandle NodeNameList
IBM.Association	d	ConfigChanged
IBM.AuditLog	p	Name MessageCatalog MessageSet DescriptionId DescriptionText RetentionPeriod MaxSize SubsystemId NodeNameList
IBM.AuditLog	d	
IBM.AuditLogTemplate	p	TemplateId Subsystem FieldDefinitions FormatMessageId NodeNameList
IBM.AuditLogTemplate	d	

Table 6-2 CSM resource class and attributes

Resource class	p/d [a]	Attributes
IBM.Condition	p	Name ResourceClass DynamicAttribute EventExpression EventDescription RearmExpression RearmDescription SelectionString ImmediateEvaluate Severity NodeNames ManagementScope NodeNameList
IBM.Condition	d	ConfigChanged MonitorStatus EventOccurred
IBM.DRM	p	
IBM.DRM	d	
IBM.DmsCtrl	p	AddUnrecognizedNodes MaxNumNodesInDomain RemoteShell ClusterTM ClusterSNum SetupRemoteShell ExpDate RegSyncDelay Frequency Sensitivity NodeNameList
IBM.DmsCtrl	d	ConfigChanged
IBM.EthernetDevice	p	Name NodeNameList
IBM.EthernetDevice	d	RecErrorRate RecDropRate XmitDropRate XmitErrorRate XmitOverflowRate RecByteRate RecPacketRate XmitByteRate XmitPacketRate RecErrors RecDrops XmitDrops XmitErrors XmitOverflows RecBytes RecPackets XmitBytes XmitPackets
IBM.EventResponse	p	Name Actions NodeNameList
IBM.EventResponse	d	ConfigChanged
IBM.FDDIDevice	p	Name NodeNameList
IBM.FDDIDevice	d	RecErrorRate RecDropRate XmitDropRate XmitErrorRate XmitOverflowRate RecByteRate RecPacketRate XmitByteRate XmitPacketRate RecErrors RecDrops XmitDrops XmitErrors XmitOverflows RecBytes RecPackets XmitBytes XmitPackets
IBM.FileSystem	p	Name MountPoint MountDir Dev Vfs Log Mount Permissions Account Type Size Frag Nbpi Compress Bf Ag AgBlkSize FsState ErrorsBehavior ReservedBlock MountCount MaxMntCount NoDev NoSuid DumpInterval FsckPass BlockSize ManualMode NodeNameList
IBM.FileSystem	d	OpState PercentTotUsed PercentINodeUsed
IBM.Host	p	Name NumProcessors RealMemSize OSName KernelVersion DistributionName DistributionVersion Architecture NodeNameList

Table 6-2 CSM resource class and attributes

Resource class	p/d [a]	Attributes
IBM.Host	d	ProcRunQueue ProcSwapQueue TotalPgSpSize TotalPgSpFree PctTotalPgSpUsed PctTotalPgSpFree PctTotalTimeIdle PctTotalTimeWait PctTotalTimeUser PctTotalTimeKernel PctRealMemFree PctRealMemPinned RealMemFramesFree VMPgInRate VMPgOutRate VMPgFaultRate VMPgSpInRate VMPgSpOutRate KMemReqMbufRate KMemReqSockRate KMemReqProtcbRate KMemReqOtherIPRate KMemReqMblkRate KMemReqStreamsRate KMemReqOtherRate KMemFailMbufRate KMemFailSockRate KMemFailProtcbRate KMemFailOtherIPRate KMemFailMblkRate KMemFailStreamsRate KMemFailOtherRate KMemNumMbuf KMemNumSock KMemNumProtcb KMemNumOtherIP KMemNumMblk KMemNumStreams KMemNumOther KMemSizeMbuf KMemSizeSock KMemSizeProtcb KMemSizeOtherIP KMemSizeMblk KMemSizeStreams KMemSizeOther VMActivePageCount PctRealMemActive LoadAverage NumUsers UpTime ActiveMgtScopes
IBM.HostPublic	p	PublicKey PublicKeyBinary NodeNameList
IBM.HostPublic	d	
IBM.HwCtrlPoint	p	Name PowerMethod PollingInterval NodeNameList
IBM.HwCtrlPoint	d	ConfigChanged
IBM.ManagedNode	p	Hostname InstallAdapterMacaddr HWType HWModel HWSerialNum LParID Name ConsolePortNum ConsoleServerName HWControlPoint HWControlNodeId InstallOSName InstallDistributionVersion InstallDistributionName InstallKernelVersion InstallDiskType InstallDisk InstallMethod ConsoleMethod ConsoleServerNumber PowerMethod LastCFMUpdateTime AllowManageRequest ManagementServer InstallPkgArchitecture InstallCSMVersion InstallAdapterType InstallAdapterSpeed InstallAdapterDuplex Mode CSMVersion UpdatenodeFailed UserComment ConsoleSerialDevice InstallServer InstallTemplate InstallAdapterGateway InstallAdapterNetmask NodeNameList
IBM.ManagedNode	d	ConfigChanged Status PowerStatus ChangedAttributes InstallStatus
IBM.ManagementServer	p	Name Hostname ManagerType LocalHostname ClusterTM ClusterSNum NodeNameList
IBM.ManagementServer	d	ConfigChanged

Table 6-2 CSM resource class and attributes

Resource class	p/d [a]	Attributes
IBM.NetworkInterface	p	Name DeviceName IPAddress SubnetMask Subnet CommGroup HeartbeatActive Aliases DstAddress DeviceSubType LogicalID NetworkID NodeNameList
IBM.NetworkInterface	d	OpState ConfigChanged
IBM.NodeAuthenticate	p	
IBM.NodeAuthenticate	d	
IBM.NodeGroup	p	Name ValidateNodes MemberList SelectStr ExpMemberList NodeNameList
IBM.NodeGroup	d	ConfigChanged MembershipChanged
IBM.NodeHwCtrl	p	Hostname HWControlPoint HWControlNodeId PowerMethod HWType HWModel HWSerialNum LParID ConsoleServerName ConsoleServerNumber ConsoleMethod ConsolePortNum InstallAdapterMacaddr NodeNameList
IBM.NodeHwCtrl	d	ConfigChanged PowerStatus
IBM.PagingDevice	p	Name Size NodeNameList
IBM.PagingDevice	d	OpState PctFree
IBM.PhysicalVolume	p	Name PVId NodeNameList
IBM.PhysicalVolume	d	PctBusy RdBlkRate WrBlkRate XferRate
IBM.PreManagedNode	p	Hostname InstallAdapterMacaddr HWType HWModel HWSerialNum LParID Name ConsolePortNum ConsoleServerName HWControlPoint HWControlNodeId InstallOSName InstallDistributionVersion InstallDistributionName InstallKernelVersion InstallDiskType InstallDisk InstallMethod ConsoleMethod ConsoleServerNumber PowerMethod InstallCSMVersion ManagementServer InstallAdapterType InstallAdapterSpeed InstallAdapterDuplex InstallPkgArchitecture NodeNameList
IBM.PreManagedNode	d	ConfigChanged
IBM.Processor	p	Name ProcessorType NodeNameList
IBM.Processor	d	OpState PctTimeIdle PctTimeWait PctTimeKernel PctTimeUser
IBM.Program	p	Name ProgramName Filter Origin NodeNameList
IBM.Program	d	Processes

Table 6-2 CSM resource class and attributes

Resource class	p/d [a]	Attributes
IBM.Sensor	p	Name Command UserName RefreshInterval Description ErrorExitValue NodeNameList
IBM.Sensor	d	ConfigChanged ExitValue String Int32 Uint32 Int64 Uint64 Float32 Float64 Quantum
IBM.ServiceEvent	p	ChangeCounter HSCId HSCName Status ErrLogLabel CallHomeCandidate CalledHome CEComments CECommentsNew FRUList FRUListAdditional FRUListNew PtrHscEED Nodes NodeNameList
IBM.ServiceEvent	d	OpState ConfigChanged Counter Quantity ConfigChangedName
IBM.Sfp	p	
IBM.Sfp	d	
IBM.TokenRingDevice	p	Name NodeNameList
IBM.TokenRingDevice	d	RecErrorRate RecDropRate XmitDropRate XmitErrorRate XmitOverflowRate RecByteRate RecPacketRate XmitByteRate XmitPacketRate RecErrors RecDrops XmitDrops XmitErrors XmitOverflows RecBytes RecPackets XmitBytes XmitPackets
IBM.WLM	p	Name Description NodeNameList
IBM.WLM	d	CPUPct CPUAtSoftMax CPUAtHardMax MemPct MemAtSoftMax MemAtHardMax DiskIOPct DiskIOAtSoftMax DiskIOAtHardMax NumProcs NumProcsAtMax NumThreads NumThreadsAtMax NumLogins NumLoginsAtMax

a. *p* for persistent attributes and *d* for dynamic attributes

Table 6-3 lists the corresponding RMC class and attributes for some of the PSSP resource variables.

Table 6-3 Corresponding RMC class attributes for PSSP variables

PSSP variable	RMC class	RMC class attribute
IBM.PSSP.Membership.LANAdapter.state	IBM.NetworkInterface	HeartbeatActive
IBM.PSSP.Membership.Node.state	IBM.NetworkInterface	HeartbeatActive
IBM.PSSP.PRCRS.procs_online	IBM.Host	NumProcessors [a]
IBM.PSSP.Prog.pcount	IBM.Program	Processes

Table 6-3 Corresponding RMC class attributes for PSSP variables

PSSP variable	RMC class	RMC class attribute
IBM.PSSP.Prog.xpcount	IBM.Program	Processes [b]
IBM.PSSP.Response.Host.state	IBM.NetworkInterface	HeartbeatActive
IBM.PSSP.aixos.CPU.glidle	IBM.Host	PctTotalTimeIdle
IBM.PSSP.aixos.CPU.glkern	IBM.Host	PctTotalTimeKernel
IBM.PSSP.aixos.CPU.gluser	IBM.Host	PctTotalTimeUser
IBM.PSSP.aixos.CPU.glwait	IBM.Host	PctTotalTimeWait
IBM.PSSP.aixos.cpu.idle	IBM.Processor	PctTimeIdle
IBM.PSSP.aixos.cpu.kern	IBM.Processor	PctTimeKernel
IBM.PSSP.aixos.cpu.user	IBM.Processor	PctTimeUser
IBM.PSSP.aixos.cpu.wait	IBM.Processor	PctTimeWait
IBM.PSSP.aixos.Disk.busy	IBM.PhysicalVolume	PctBusy
IBM.PSSP.aixos.Disk.rblk	IBM.PhysicalVolume	RdBlkRate
IBM.PSSP.aixos.Disk.wblk	IBM.PhysicalVolume	WrBlkRate
IBM.PSSP.aixos.Disk.xfer	IBM.PhysicalVolume	XferRate
IBM.PSSP.aixos.FS.%nodesused	IBM.FileSystem	PercentINodeUsed
IBM.PSSP.aixos.FS.%totused	IBM.FileSystem	PercentTotUsed
IBM.PSSP.aixos.VG.free	[c]	
IBM.PSSP.aixos.LAN.rcverrors	IBM.ATMDevice / IBM.EthernetDevice / IBM.FDDIDevice / IBM.TokenRingDevice /	RecErrorRate
IBM.PSSP.aixos.LAN.recvdrops	IBM.ATMDevice / IBM.EthernetDevice / IBM.FDDIDevice / IBM.TokenRingDevice /	RecDropRate
IBM.PSSP.aixos.LAN.xmitdrops	IBM.ATMDevice / IBM.EthernetDevice / IBM.FDDIDevice / IBM.TokenRingDevice /	XmitDropRate

Table 6-3 Corresponding RMC class attributes for PSSP variables

PSSP variable	RMC class	RMC class attribute
IBM.PSSP.aixos.LAN.xmiterrors	IBM.ATMDevice / IBM.EthernetDevice / IBM.FDDIDevice / IBM.TokenRingDevice /	XmitErrorRate
IBM.PSSP.aixos.LAN.xmitovfl	IBM.ATMDevice / IBM.EthernetDevice / IBM.FDDIDevice / IBM.TokenRingDevice /	XmitOverflowRate
IBM.PSSP.aixos.Mem.Kmem.calls	IBM.Host	KMemReqMbufRate / KMemReqSockRate / KMemReqProtcbRate / KMemReqOtherIPRate / KMemReqMblkRate / KMemReqStreamsRate / KMemReqOtherRate
IBM.PSSP.aixos.Mem.Kmem.failures	IBM.Host	KMemFailMbufRate / KMemFailSockRate / KMemFailProtcbRate / KMemFailOtherIPRate / KMemFailMblkRate / KMemFailStreamsRate / KMemFailOtherRate
IBM.PSSP.aixos.Mem.Kmem.inuse	IBM.Host	KMemNumMbuf / KMemNumSock / KMemNumProtcb / KMemNumOtherIP / KMemNumMblk / KMemNumStreams / KMemNumOther
IBM.PSSP.aixos.Mem.Kmem.memuse	IBM.Host	KMemSizeMbuf / KMemSizeSock / KMemSizeProtcb / KMemSizeOtherIP / KMemSizeMblk / KMemSizeStreams / KMemSizeOther
IBM.PSSP.aixos.Mem.Real.%free	IBM.Host	PctRealMemFree
IBM.PSSP.aixos.Mem.Real.%pinned	IBM.Host	PctRealMemPinned
IBM.PSSP.aixos.Mem.Real.numfrb	IBM.Host	RealMemFramesFree

Table 6-3 Corresponding RMC class attributes for PSSP variables

PSSP variable	RMC class	RMC class attribute
IBM.PSSP.aixos.Mem.Real.size	IBM.Host	RealMemSize
IBM.PSSP.aixos.Mem.Virt.pagein	IBM.Host	VMPgInRate
IBM.PSSP.aixos.Mem.Virt.pageout	IBM.Host	VMPgOutRate
IBM.PSSP.aixos.Mem.Virt.pagexct	IBM.Host	VMPgFaultRate
IBM.PSSP.aixos.Mem.Virt.pgspgin	IBM.Host	VMPgSpInRate
IBM.PSSP.aixos.Mem.Virt.pgspgout	IBM.Host	VMPgSpOutRate
IBM.PSSP.aixos.PagSp.%totalfree	IBM.Host	PctTotalPgSpFree
IBM.PSSP.aixos.PagSp.%totalused	IBM.Host	PctTotalPgSpUsed
IBM.PSSP.aixos.PagSp.totalfree	IBM.Host	TotalPgSpFree
IBM.PSSP.aixos.PagSp.totalsize	IBM.Host	TotalPgSpSize
IBM.PSSP.aixos.pagsp.%free	IBM.PagingDevice	PctFree
IBM.PSSP.aixos.pagsp.size	IBM.PagingDevice	Size
IBM.PSSP.aixos.Proc.runque	IBM.Host	ProcRunQueue
IBM.PSSP.aixos.Proc.swpque	IBM.Host	ProcSwapQueue
IBM.PSSP.pm.Errlog	[d]	
IBM.PSSP.pm.User_state1-16	IBM.Sensor	

a. PSSP variable reports active processors while RMC reports installed processors.
b. PSSP provides two Prog.count variable, pcount for all processes while xpcount for processes as a result of exec(), RMC does not have xpcount equivalent.
c. There is no resource class in CSM corresponding to PSSP resource class IBM.PSSP.aixos.VG.free, we show you how you can implement it with IBM.Sensor in Example 6-29 on page 142.
d. RMC/CSM does not provide resource monitor for errlog, we show you how you could implement this function in “Monitoring errlog” on page 138.

6.3 Defining a monitor with predefined resource class

Under PSSP, a monitor can be defined using the **pmandef** command, which specifies a condition by an expression and an action associated with that condition. The expression specifies the resource variable, the associated resource identifier, and a value under which an event will be generated. The action can be executing a command, generating an SNMP trap, or writing an

entry to the AIX *errlog* or *syslog*. Optionally, **pmandef** can specify a rearm condition and its associated action.

Running the **pmandef** command shown in Example 6-11 on a server causes the command **echo program has stopped >/tmp/myevent.out** to run on the server whenever the number of processes named *mycmd* and owned by user *bob* on node 12 becomes 0. When this number increases back to 1, the command **echo program has restarted >/tmp/myrearm.out** runs on the server.

Example 6-11 pmandef sample 1

```
pmandef -s Program_Monitor \
    -e 'IBM.PSSP.Prog.pcount:NodeNum=12;ProgName=mycmd;UserName=bob:X@0==0' \
    -r "X@0==1" \
    -c "echo program has stopped >/tmp/myevent.out" \
    -C "echo program has restarted >/tmp/myrearm.out"
```

Example 6-12 shows a scenario that causes recovery commands to run on both k21n01 and k21n02 nodes whenever bob's program dies or gets restarted on either nodes 12 or 13.

Example 6-12 pmandef sample 2

```
pmandef -s Program_Monitor \
    -e 'IBM.PSSP.Prog.pcount:NodeNum=12-13;ProgName=mycmd;UserName=bob:X@0==0'\
    -r "X@0==1" \
    -c /usr/local/bin/start_recovery \
    -C /usr/local/bin/stop_recovery \
    -h k21n01,k21n02
```

A single **pmandef** command defines the condition, the response, the association between the condition and the response, and starts the monitor as well.

Under CSM, these are separate steps. You first define the condition with **mkcondition**, then define the response with **mkresponse**, then define the association with **mkcondresp** (optional), then start the monitor with **startcondresp**.

The command **pmandef** can specify the list of servers where the conditions apply using “NodeNum=”. It can specify the list of servers where the responses apply in a number of different ways, using **-N NodeGroup**, **-n NodeRange** or **-h ListOfServerNames**.

Under CSM, RMC commands work in a local domain, a management domain, or a peer domain; you can specify the domain with option **-m** and value of *l*, *m,* or *d,* respectively.

You have the option to define where the event is monitored and where the condition or response is created. You can use **-n** to provide a list of servers that should be monitored on that domain. Use **-p** to specify the server on which the condition should be created—it has to be the management server if the condition has a management domain scope.

CSM provides many predefined conditions and responses. The command **lscondition** gives a list of defined conditions, and **lscondition -C** shows the command that would create the *condition*. Likewise, **lsresponse** gives a list of defined responses, and **lsresponse -C** shows the command that would create the *response*.

6.3.1 mkcondition

The command **mkcondition** is used to create a new condition definition that can be monitored. It defines an event expression on a resource class attribute under which the event occurs.

An event expression consists of a dynamic attribute name, a mathematical comparison symbol, and a constant. A selection string is similar but works on a persistent attribute. The expression in the definition can be either similar to a C language statement or to the WHERE clause of an SQL query. Refer to Table 20, “Operators that can be used in expressions” in *RSCT for AIX 5L: Guide and Reference,* SA22-7889 for a complete list of operators you may use.

Example 6-13 defines a condition similar to what the **pmandef** command does in Example 6-11 without defining a response. Also, since it is defined in a management domain, it must be created on the management server. The pman monitor can be created on any host within the SP partition.

Example 6-13 mkcondition sample 1

```
mkcondition -r IBM.Program \
        -s 'ProgramName=="mycmd" && Filter=="user==\"bob\""' \
        -e Processes.CurPidCount==0 \
        -E Processes.CurPidCount!=0 \
        -m m \
        -n node12 \
        Program_Monitor
```

Similar to pman in PSSP, RMC keeps track of the previously observed value of the dynamic attribute, so the event expression can compare the currently observed value with the previously observed value. Like pman, the previously observed value is represented by adding suffix “@P” to the dynamic attribute name. Example 6-14 on page 133 creates an event on node5 whenever the number of users changes.

Example 6-14 mkcondition sample 2

```
mkcondition -r IBM.Host \
        -e NumUsers!=NumUsers@P \
        -m m \
        -n node5 \
        User_Monitor
```

6.3.2 mkresponse

The **mkresponse** command creates a new response definition. Unlike **pmandef**, now you have the option of defining the days of the week and the time of day for which the response can be run. Once a response is defined, you can use it for any defined condition simply by referring to the name of the response. You can also add additional actions to the response using the **chresponse** command.

Example 6-15 shows a response definition similar to what **pmandef** does in Example 6-12 on page 131. You can specify whether the response is for event only (**-e a**), rearm event only (**-e r**), or both (**-e b**).

Example 6-15 mkresponse sample

```
mkresponse -n start_script -s /usr/local/bin/start_recovery \
        -e a \
        start_recovery
mkresponse -n stop_script -s /usr/local/bin/stop_recovery \
        -e r \
        stop_recovery
```

A number of environment variables are available for use in your response script. A comparison of pman and ERRM environment variables is found in Table 6-4. The new additions in ERRM include:

- ERRM_TYPE indicates whether this is an event or a rearm event.
- ERRM_RSRC_TYPE indicates whether the resource is an existing one, a new one, or a deleted one.
- ERRM_COND_SEVERITY indicates whether the event is informational, warning, or critical.

For a complete list of ERRM environment variables, see Table 16 of *RSCT for AIX 5L: Guide and Reference,* SA22-7889.

Table 6-4 Corresponding pman and ERRM environment variables

PMAN	ERRM
PMAN_HANDLE	ERRM_COND_NAME

PMAN	ERRM
PMAN_PRINCIPAL	
PMAN_DCEPRIN	
PMAN_RVNAME	ERRM_RSRC_CLASS_PNAME (ERRM_RSRC_CLASS_NAME), ERRM_ATTR_PNAME (ERRM_ATTR_NAME)
PMAN_IVECTOR	ERRM_RSRC_NAME
PMAN_PRED	ERRM_EXPR
PMAN_TIME	ERRM_TIME
PMAN_LOCATION	ERRM_NODE_NAME
PMAN_RVTYPE	ERRM_DATA_TYPE
PMAN_RVCOUNT/PMAN_RVFIELD#	ERRM_SD_DATA_TYPE
PMAN_RVVALUE	ERRM_VALUE

6.3.3 mkcondresp and startcondresp

You can use **mkcondresp** to define the association between defined conditions and responses, or use **startcondresp** to link and activate the association at the same time, as shown in Example 6-16. Note that you can define multiple responses for a single condition.

Example 6-16 startcondresp sample

```
startcondresp Program_Monitor start_recovery stop_recovery
```

6.3.4 Event and response on the same server

Under PSSP, this is defined using the **-h local** option with the **pmandef** command, as shown in Example 6-17.

Example 6-17 pmandef event and response on the same server

```
pmandef -s Filesystem_Monitor_Tmp \
    -e 'IBM.PSSP.aixos.FS.%totused:NodeNum=*;VG=rootvg;LV=hd3:X>90' \
    -r 'X<75' \
    -c /usr/local/bin/pman_notify \
    -C "/usr/local/bin/pman_notify -r" \
    -h local
```

This is considered a local domain event under CSM, and a monitor can be defined using **mkcondition** with the option **-m l -p hostname** on the management server, or the server itself, as shown in Example 6-18. If you run the command on the server itself, the **-p hostname** and the **:hostname** options can be omitted.

Example 6-18 mkcondition event and response on the same server

```
mkcondition -r IBM.FileSystem \
        -e 'PercentTotUsed>90' \
        -E 'PercentTotUsed<75' \
        -s 'Name=="/tmp"' \
        -m l \
        -p node5 \
        Filesystem_Tmp_Monitor
mkresponse -n "logEvent" \
    -s "/usr/local/bin/myresponse" \
    -e b \
    -p node5 \
    myresponse
startcondresp Filesystem_Tmp_Monitor:node5 myresponse
```

6.3.5 Event on managed nodes, response on management server

Under PSSP, this is defined using the **-n 0** option with the **pmandef** command as shown in Example 6-19.

Example 6-19 pmandef event on managed node(s), response on management server

```
pmandef -s Filesystem_Monitor_Tmp \
    -e 'IBM.PSSP.aixos.FS.%totused:NodeNum=*;VG=rootvg;LV=hd3:X>90' \
    -r 'X<75' \
    -c /usr/local/bin/pman_notify \
    -C "/usr/local/bin/pman_notify -r" \
    -n 0
```

This is considered as a management domain event under CSM, and a monitor can be defined using **mkcondition** with the option **-m m** on the management server, as shown in Example 6-20.

Example 6-20 mkcondition event on managed nodes, response on management server

```
mkcondition -r IBM.FileSystem \
        -e 'PercentTotUsed>90' \
        -E 'PercentTotUsed<75' \
        -s 'Name=="/tmp"' \
        -m m \
```

```
        Filesystem_Tmp_Monitor
mkresponse -n "logEvent" -s "/usr/local/bin/myresponse" -e b myresponse
startcondresp Filesystem_Tmp_Monitor myresponse
```

6.3.6 Event on all managed nodes and the management server

Under PSSP, this is simply accomplished by using **NodeNum=***, which includes the CWS and all the nodes in the cluster.

Under CSM, you need to define a monitor on the management server as a separate local domain event with the option **-m l**. For all the managed nodes, you can also define a local domain event on each managed node. The better way is to define a management domain event for the whole cluster with the option **-m m** and without the **-n** option.

Note: The management server is not considered a host in the management domain.

6.3.7 Event on some managed nodes

Under PSSP, this is done by specifying a list of node numbers with the **NodeNum=** option on the **pmandef** command.

Under CSM, this is done by specifying a list of node names with the **-m m -n** options on the **mkcondition** command.

6.3.8 Event and response on the management server

Under PSSP, a monitor for a CWS only event is defined using **NodeNum=0** and **-n 0** with the **pmandef** command. Under CSM, a condition is defined on the management server as a local domain event with the option **-m l**.

6.3.9 Event and response on different managed nodes

Under PSSP, this is easily accomplished by using **NodeNum=** to include the node numbers where the event is monitored, and **-n** to list the node numbers where the responses will happen, as shown in Example 6-21.

Example 6-21 pmandef event and response on different managed nodes

```
pmandef -s Filesystem_Tmp_Monitor \
    -e 'IBM.PSSP.aixos.FS.%totused:NodeNum=7-9;VG=rootvg;LV=hd3:X>90' \
```

```
    -r 'X<75' \
    -c /usr/local/bin/pman_notify \
    -C "/usr/local/bin/pman_notify -r" \
    -n 5,6
```

Under CSM, the managed nodes only know the management server but not each other. Therefore, RMC on one managed node cannot communicate an event to another managed node for response action. One workaround is to have the management server responding to the event. During the response, the management server can open a remote shell to each of the managed nodes that should respond to the events to start the appropriate response actions.

However, some care needs to be taken. For example, the remote shell needs to be passed the necessary environment information so that it knows the detailed conditions under which the event happened.

Here is an example of how to implement Example 6-21 under CSM. First, add **-m m -n hostlist** to **mkcondition** to limit the condition to only the managed nodes to be monitored; see Example 6-22.

Example 6-22 mkcondition event and response on different managed nodes

```
mkcondition -r IBM.FileSystem \
        -e 'PercentTotUsed>90' \
        -E 'PercentTotUsed<75' \
        -s 'Name=="/tmp"' \
        -m m \
        -n node8,node9 \
        Filesystem_Tmp_Monitor
```

Second, when defining the response on the management server, we pass an environment variable to hold the managed node names on which the actual response should happen, as shown in Example 6-23.

Example 6-23 mkresponse event and response on different managed nodes

```
mkresponse -n "logEvent" \
        -s "/usr/local/bin/myresponse" \
        -e b \
        -E DSH_OPT="-vw node3 -vw node5"
        myresponse
```

Third, we modify the response script on the management server to examine the environment variable and to take appropriate action according to the values of the environment variable, as shown in Example 6-24 on page 138.

Example 6-24 Response script event and response on different managed nodes

```
if [ -n "$DSH_OPT" ]; then
   ERRM_ENV=$(env | grep "^ERRM_" | sed 's@=\(.*\)@="\1" @' | tr -d '\n')
   exec /opt/csm/bin/dsh $DSH_OPT $ERRM_ENV $0 $*
fi
```

These lines are inserted in the beginning of the script. If the environment DSH_OPT is not set, the script proceeds as normal. Thus the same script can be copied to all the managed nodes and the management server. If DSH_OPT is set, then it simply runs the same script with the same options on remote managed nodes as defined by the environment variable DSH_OPT.

Note that we chose to implement it as a **dsh** option so it could be very flexible; for example, a dynamic node group name can be used here. Also note that in the example, we used **-vw node3 -vw node5**, not **-vw node3, node5** since the **mkresponse** command considers the comma (,) as an environment variable separator.

6.3.10 Monitoring errlog

Monitoring the AIX error log entry under PSSP is done through the resource variable IBM.PSSP.pm.Errlog, which provides summary data from an entry written to the AIX error log. The value of IBM.PSSP.pm.Errlog is a structured byte string that consists of nine fields, each corresponding to a field in the AIX error log entry, as shown in Table 6-5.

Table 6-5 IBM.PSSP.pm.Errlog field information

IBM.PSSP.pm.Errlog field number	AIX error log field	Sample value
0	Sequence number	2333
1	Error ID	0x825849bf
2	Error class	H
3	Error type	TEMP
4	Alert flags value	FALSE
5	Resource name	fcs1
6	Resource type	df1000f9
7	Resource class	adapter
8	Error label	FCS_ERR2

Tip: Use **errpt -t** to find errors you would like to monitor, **errpt -atJ** ErrLogLabel to get detailed information on ErrLogLabel.

Since all the information from the AIX errorlog is available, monitoring is very flexible. Example 6-25 monitors allFibre Channel adapter errors.

Example 6-25 pmandef FCS adapter error

```
pmandef -s FCS_Adaptor_Monitor \
        -e 'IBM.PSSP.pm.Errlog:NodeNum=*:X@6=="df1000f7"||X@6=="df1000f9"' \
        -c /usr/local/bin/recover_fcs \
        -h local
```

Example 6-26, on the other hand, monitors only GXENT_LINK_DOWN errors.

Example 6-26 pmandef GXENT_LINK_DOWN error

```
pmandef -s GXENT_LINK_DOWN_Monitor \
        -e 'IBM.PSSP.pm.Errlog:NodeNum=*:X@8=="GXENT_LINK_DOWN"' \
        -c /usr/local/bin/recover_gxe \
        -h local
```

RMC or CSM do not provide a resource manager for the AIX errorlog. Here is a workaround to offer some ideas on how error entries can be monitored using the AIX errorlog notification facility.

Example 6-27 is similar to what PSSP put into the ODM for errorlog monitoring. You can save it to a file and add it into ODM with the command **odmadd** <*filename*>. You need do this on every server where you would like to monitor error entries. Whenever an entry is written into the errorlog, a script defined by the en_method will be executed. The parameters passed to the script are the fields described in Table 6-5 on page 138.

Example 6-27 Create AIX error log notification ODM entry

```
errnotify:
        en_pid = 0
        en_name = "ErrLogMgmt"
        en_persistenceflg = 1
        en_label = ""
        en_crcid = 0
        en_class = ""
        en_type = ""
        en_alertflg = ""
        en_resource = ""
        en_rtype = ""
        en_rclass = ""
```

```
        en_symptom = ""
        en_err64 = ""
        en_dup = ""
        en_method = "/usr/local/bin/errlog_rm \" $1\" \" $2\" \" $3\" \" $4\"
\" $5\" \" $6\" \" $7\" \" $8\" \" $9\""
```

Example 6-28 is a simplified example of the errorlog response script. It mails the error entry to root at the management server and responds to error entries as **pmandef** does in Example 6-25 on page 139 and Example 6-26 on page 139.

Example 6-28 Sample script to monitor AIX errlog

```
#!/bin/ksh
/usr/bin/errpt -a -l $1 |
  mail -s "ErrLogLabel $9 on `hostname`" root@management_server
if [ "$7" == "df1000f7" -o "$7" == "df100f9" ]; then
  /usr/local/bin/recover_fcs
elif [ "$9" == "GXENT_LINK_DOWN" ]; then
  /usr/local/bin/recover_gxe
fi
```

6.3.11 Generating AIX error log or BSD syslog entries

Under PSSP, this is done using the **-l EventLogText -L RearmLogText** option on the **pmandef** command.

RMC provides the sample script /usr/sbin/rsct/bin/logevent, which logs messages to a user-specified file using **alog**. You can also write your own script to log messages to the AIX error log using the **errlogger** command.

See *RSCT for AIX 5L: Technical Reference,* SA22-7890 for more information on the **logevent** command.

6.3.12 Sending SNMP traps

Under PSSP, this is done using the **-t EventTrapid -T RearmTrapid** option on the **pmandef** command.

Under RMC, this is done by running an **snmptrap** command. A sample script for calling the **snmptrap** command is provided by CSM and can be found in /usr/sbin/rsct/bin/snmpevent. See *RSCT for AIX 5L: Technical Reference,* SA22-7890 for more information on the **snmpevent** command.

6.4 Defining custom monitors

Under PSSP, up to 16 custom monitors can be defined by the user using the resource variables IBM.PSSP.pm.User_state1~16. The steps are:

1. Create the custom monitor script

```
$ cat /usr/local/bin/ck_dft_route
   #!/bin/ksh
   A=$(/usr/bin/netstat -rn | /usr/bin/awk '$1 == "default" {print $2}')
   echo ${A:-"NodefRoute"}
```

2. Create the pmanrmd configuration file

```
$ cat /tmp/tmpfile
   TargetType=NODE_RANGE
   Target=ALLPMAN
   Rvar=IBM.PSSP.pm.User_state2
   SampInt=60
   Command="/usr/local/bin/ck_dft_route”
```

3. Load the monitor into the SDR

```
$ pmanrmdloadSDR /tmp/tmpfile
```

4. Create the monitor definition

```
$ pmandef -s deRoute_Monitor \
   -e 'IBM.PSSP.pm.User_state2:NodeNum=*:X@0=="NoDefRoute"' \
   -r 'X@0!="NoDefRoute"' \
   -c /usr/local/bin/recover_route \
   -C /usr/local/bin/stop_recovery \
   -h local
```

5. Refresh the daemon

```
$ dsh -a stopsrc -s pmanrm; stopsrc -s pmanrm.partition_name
$ dsh -a startsrc -s pmanrm; startsrc -s pmanrm.partition_name
```

Creating a custom monitor under RMC is done using the IBM.Sensor resource class. The steps are:

1. Create the custom monitor script

```
$ cat /usr/local/bin/ck_dft_route
   #!/bin/ksh
   A=$(/usr/bin/netstat -rn | /usr/bin/awk '$1 == "default" {print $2}')
   echo “String=”${A:-"NodefRoute"}
   exit 0
```

Tip:

- By default, the sensor script must have a return value of 0 or the conditions defined on the sensor will stop working after the first occurrence of a nonzero return. You can change the default behavior by using the **-e 0** option with the **mksensor** command.
- lssensor *sensor_name* will execute the command defined by **mksensor** *sensor_name.* To list the definition of a sensor without executing the command defined by the sensor, use **lsrsrc -s 'Name="sensor_name"' IBM.Sensor**.

2. Create the sensor

```
$ dsh -av mksensor -e 0 deRoute_Monitor /usr/local/bin/ck_dft_route
```

Or

```
$ lsnode -s |
xargs -I{} mksensor -e 0 -n {} deRoute_Monitor /usr/local/bin/ck_dft_route
```

Attention: mksensor does not support node groups, a list of node names, or domains. You need to create a sensor one node at a time. **rmsensor** does not check if a condition is defined using the sensor.

3. Create the condition and the response, and link the condition to the response

```
$ mkcondition -r IBM.Sensor \
   -e 'String=”NoDefRoute”' \
   -E ‘String!=”NoDefRoute”’\
   -s 'Name=="deRoute_Monitor"' \
   -m m deRoute_Monitor
$ mkresponse -n “logEvent” -s /usr/local/bin/recover_route -e a recovery
$ mkresponse -n “logEvent” -s /usr/local/bin/stop_recovery -e r
stop_recovery
$ startcondresp deRoute_Monitor recovery stop_recovery
```

Example 6-29 shows how to create a monitor that is similar to the PSSP variable IBM.PSSP.aixos.VG.free.

Example 6-29 CSM sample implementation of PSSP variable IBM.PSSP.aixos.VG.free

```
$ ls -l /usr/local/bin/VGfree.ksh
-rwxr--r--   1 root     system       235 May 21 16:10 /usr/local/bin/VGfree.ksh
$ cat /usr/local/bin/VGfree.ksh
#!/bin/ksh
vg=${1:-rootvg}
String=$(/usr/sbin/lsvg $vg 2>&1)
Uint32=$(echo "$String"|/bin/awk '$0 ~ "FREE PPs:" {print substr($7,2)}')
```

```
if [ -z "$Uint32" ]; then
   echo String=$String
else
   echo Uint32=$Uint32
fi
exit 0
$ mksensor -i 86400 rootVGfree '/usr/local/bin/VGfree.ksh rootvg'
$ mkcondition -r IBM.Sensor -e 'Uint32<10' -s 'Name=="rootVGfree"' rootVGfull
```

6.4.1 CFMRootModTime

If you have rsct.core.sensorrm installed, the **/opt/csm/bin/predefined-condresp** command creates the predefined sensor CFMRootModTime, a predefined condition CFMRootModTimeChanged, and a predefined response CFMModResp. When you associate and start them, modifications of the CFM repository files are automatically pushed out to the managed nodes. If CSM did not automatically create them for you, you can follow Example 6-30 to create and activate them.

Example 6-30 CFM predefined sensor, condition and response

```
$ mksensor CFMRootModTime "/opt/csm/csmbin/mtime /cfmroot"
$ mkcondition -r "IBM.Sensor" \
    -e "String!=String@P" \
    -d "An event will be generated whenever a file under /cfmroot is added or
modified." \
    -s "Name=\"CFMRootModTime\"" \
    CFMRootModTimeChanged
$ mkresponse -n "CFMModResp" \
    -s "/opt/csm/csmbin/CFMmodresp" \
    -e b \
    "CFMModResp"
$ startcondresp "CFMRootModTimeChanged" "CFMModResp"
```

6.5 ACL, logs, and relevant commands

The ACL for pman is found in /etc/sysctl.pman.acl. The ACL for RMC is found in /var/ct/cfg/ctrmc.acls, and a sample is found in /usr/sbin/rsct/cfg/ctrmc.acls.

For non-root users to create their own sensor, the userids need to have write permission to the IBM.Sensor class. The sensor script will be run under the userid that ran the **mksensor** command.

For a non-root user to run the **mkcondition**, **mkresponse**, and **mkcondresp** (or **startcondresp**) commands, the userid needs to have write permission to the

IBM.Condition, IBM.Response, and IBM.Association resource classes, respectively. The response script always runs under the root ID regardless of which ID ran the **mkresponse** command. This could lead to users creating a response running their own script under root authority. Be very careful if you want to change the default ACL file to allow non-root userid access to classes such as IBM.Response. Do not open doors for unwanted access.

RMC has an audit log and all ERRM events are recorded. The command **lsaudrec** is used to display records from the audit log. Table 6-6 on page 144 is a quick cross-reference of the event management-related commands for PSSP and RMC.

6.6 Visual cluster monitoring

Table 6-6 Event monitor command comparison between PSSP and CSM

Function	PSSP	RMC
Daemon control	pmanctrl	rmcctrl
Query resource	SDRGetObjects	lsrsrc, lsrsrcdef, lsactdef
Define monitor	pmandef -s	mkcondition, mkresponse, mkcondresp
Query definition	pmanquery	lscondition, lsresponse, lscondresp
Custom monitor	pmanrmdloadSDR, SDRDeleteObjects	mksensor, chsensor, rmsensor, lssensor
Remove monitor	pmandef -u	rmcondition, rmresponse, rmcondresp
Change monitor		chcondition, chresponse, chcondresp
Activate monitor	pmandef -a	startcondresp
De-activate monitor	pmandef -d	stopcondresp
Query status	pmandef -q	lscondition, lsresponse, lscondresp
Copy definition		mkcondition, mkresponse

Visual cluster monitoring is provided through the Web-based System Manager, which provides a graphical representation of the nodes in a cluster, similar to Perspectives for PSSP. Figure 6-3 on page 145 shows the initial Web-based

System Manager screen with our management server and one CSM cluster listed in the left window pane. When you select an object in the left window pane, the available options are displayed to the right.

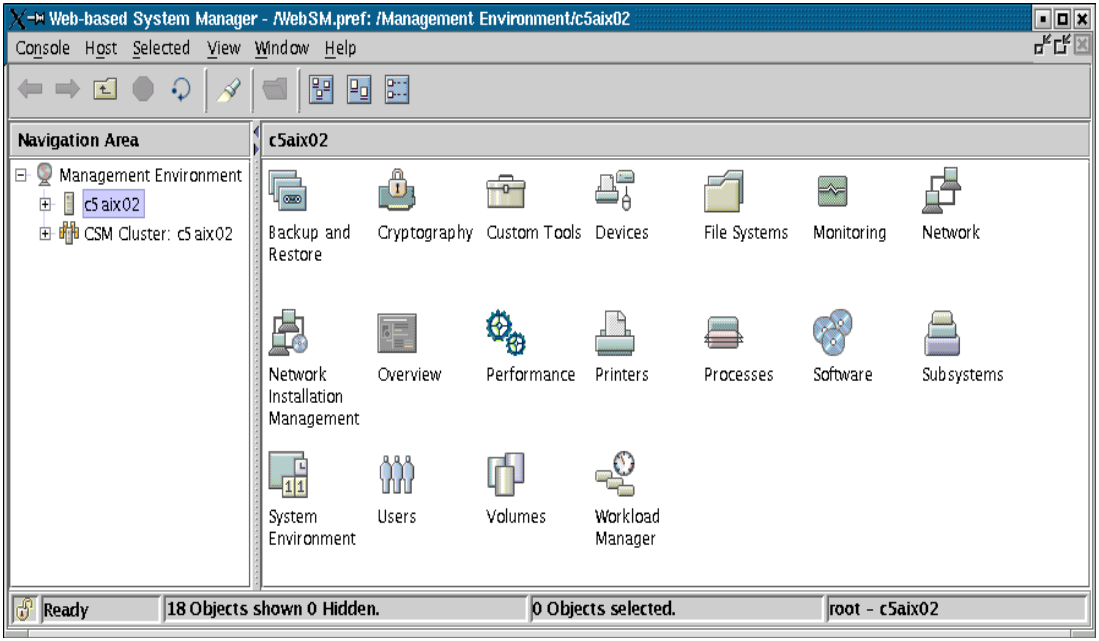


Figure 6-3 Initial Web-based System Manager (WebSM) screen

There are two types of monitoring available through the Web-based System Manager interface:

- Individual machines can be monitored by selecting the **Overview** icon.
- Cluster-wide events can be monitored by selecting the **Distributed Monitoring** icon.

Figure 6-4 on page 146 shows the information that gets displayed for an individual machine when Overview is selected. In addition to static information about the machine, such as operating system version and hardware configuration, dynamic information is displayed that shows machine utilization. Attributes such as CPU utilization and memory usage are updated in near-realtime, while file system statistics are only updated in 24-hour intervals. You can force an update of file system statistics by selecting **View** → **Reload** from the title bar. This information can be displayed for the CSM Management Server as well as for the individual cluster nodes.

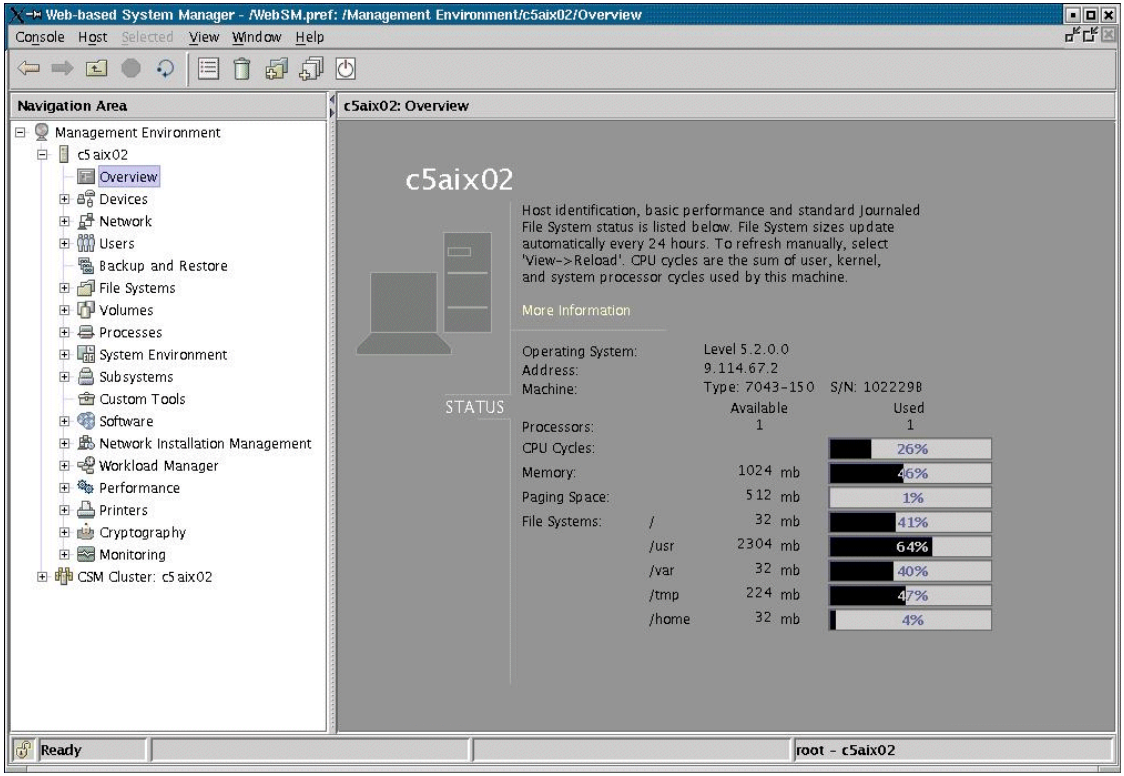


Figure 6-4 Overview screen

The Monitoring and Distributed Monitoring screens allow you to monitor events that occur on individual nodes or the entire cluster, respectively. Events are created by defining conditions and associating responses to be performed when a condition is met.

Figure 6-5 on page 147 shows a Web-based System Manager session monitoring events for cluster nodes. In this example, you can see we are monitoring file system space on a number of nodes, and node1, node7, and node9 have passed our event threshold and triggered an event. In the case of node9, the problem was cleared and a rearm event was logged indicating that the problem has been resolved and is being monitored again.

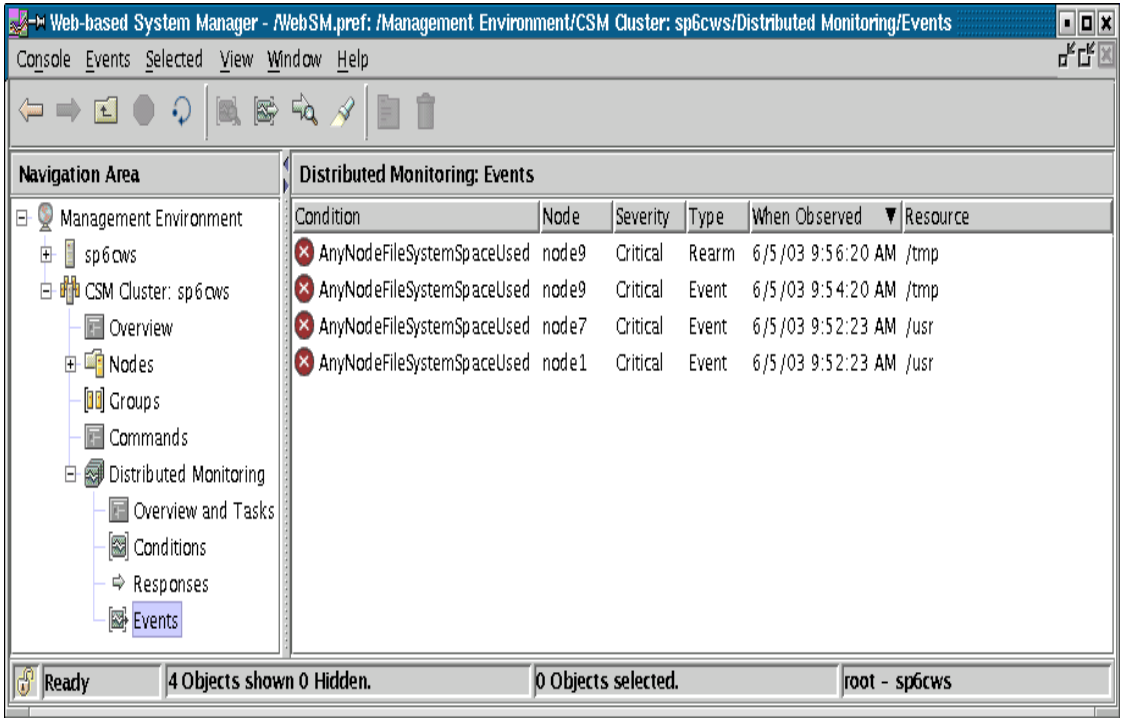


Figure 6-5 Web-based System Manager monitoring events for cluster nodes

7

Security and access control

In this chapter, we compare PSSP's and CSM's security and access control models with regard to:

- Remote shell configuration options
- Security enablement of the cluster infrastructure
- Least privilege administration

We also address the following topics:

- Access to remote hardware state control and serial connections
- Securing remote monitoring and administration through Web-based System Manager
- Access control in the NIM context

This chapter is not intended to be self-contained nor is it a low-level manual on how to work with CSM commands. Instead, the goal is to survey, from a PSSP perspective, the security functionality that CSM currently offers. We assume the reader is familiar with basic security-related notions such as identification, authentication, and authorization; and widely employed security mechanisms, for example Kerberos authentication protocol, and concepts of public key cryptography as they pertain to using and administering secure shell implementations complying with the Internet Engineering Task Force (IETF) *secsh* working group protocol.

7.1 Overview of PSSP's and CSM's security models

Basic aspects of implementation of a security model in a connected set of computers are: The mechanisms employed to ensure the authenticity of the involved communication parties and communication itself, and the authorization of processes and the owners who can execute or have influence on processes on other nodes in the cluster. In a UNIX environment, one may further distinguish between daemon processes providing the cluster services, monitoring and control of the cluster environment, and remote command execution by a system's root user or regular users.

The management model, in general, can require a common user and resource namespace in the cluster, or it may dispense with such an assumption. A further aspect is confidentiality of the administrative communication, in particular when it directly relates to protecting the cluster's resources.

To be able to delegate tasks that do not require full root access to operators, or to allow operators control over only subsets of the cluster, requires fine-grained access control mechanisms for the resources; even when restricting different operators' access to cluster resources is not an issue, it is good practice to administer a system using least privilege commands to help limit the possible side effects of operational errors.

In the standard PSSP environment, PSSP essentially assumes that there is only one superuser for the entire cluster (root). PSSP requires that the root user on any node in a system partition have full direct remote command execution privileges on every other node of the same system partition, including the control workstation. This implies that once the root identity on any node is compromised, the entire cluster is immediately under the control of the attacker. This assumes that the same PSSP and remote command authentication and authorization controls are the same (that is, common) on the control workstation and all of its nodes. When the SP contains multiple system partitions, each partition can be configured differently. The control workstation will be the union of all the security methods in all of its partitions.

PSSP supplies scripts that establish and manage the SP's authentication and authorization model through installing and updating corresponding control files when the cluster is initially set up or extended by additional nodes, respectively. The security methods can also be changed as part of steady-state operations.

Some internal services of PSSP rely on the availability of a remote root shell to any other node in the partition without interactive password entry. PSSP's inter-node root authorization structure can be illustrated as in Figure 7-1 on page 151.

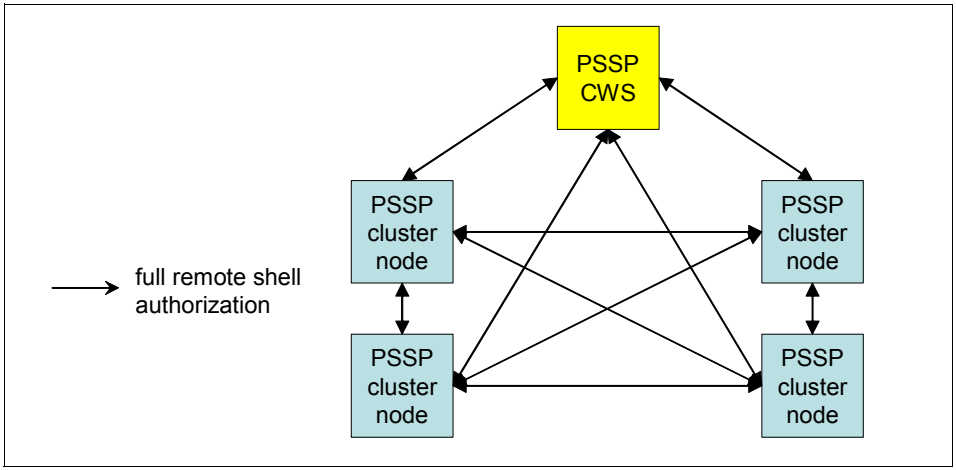


Figure 7-1 PSSP remote command authorization

For environments with higher security requirements, as of Release 3.2, PSSP offers the Restricted Root Access (RRA) option. In this configuration, the root user on the control workstation still has full remote command access to the PSSP managed nodes, but the reverse is not true.

Instead, the root user or processes running under the root user's authority on a regular PSSP node can exercise only a limited set of control commands on the control workstation. This restricted access to resources on the control workstation enjoying direct accessibility only from the control workstation's local root user is implemented with the help of the PSSP sysctl daemon running as root on the control workstation, and sysctl scripts specifically written to implement these limited accesses from outside the control workstation. As a consequence, in RRA mode, PSSP does not need to internally issue **`rsh`** and **`rcp`** commands as root from a node into the control workstation.

The sysctl daemon is also available in the standard PSSP environment. It runs on nodes and on the control workstation with local root authority, and offers a TCL-like script language environment that allows fine-grained access control to every command through authorization callback functions.

Because not all of PSSP's services have been enabled for full control through sysctl, PSSP in an RRA configuration does not support boot/install servers, VSDs and GPFS. On the other hand, RRA is a prerequisite for firewalling an SP system (see *PSSP Implementing a Firewalled RS/6000 SP System,* GA22-7874), or, as of PSSP 3.4, replacing the remote command programs with IETF secsh protocol compliant alternatives (see *PSSP Installation and Migration Guide,* GA22-7347). Note that while for RRA with rsh, PSSP still manages the

authorization files in the cluster, when employing RRA with alternative remote shells, the system partition's auth_root_cmd attribute=none and it is the administrator's responsibility to set up and maintain the authorization control files.

The access model underlying RRA is illustrated in Figure 7-2.

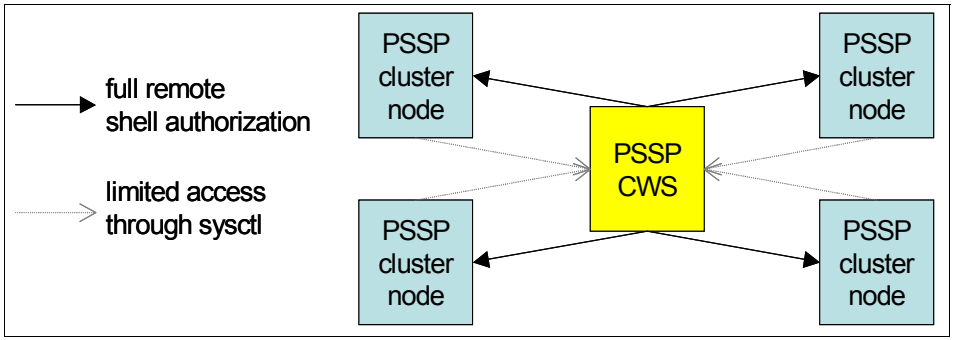


Figure 7-2 PSSP remote command authorization in Restricted Root Access

To be managed effectively, PSSP suggests a common namespace in the system partitions, especially when non-root users come into play. This requirement can easily be addressed with the various flavors of Kerberos authentication mechanisms supported by PSSP: Kerberos 5 in a DCE environment, AFS® Kerberos, or PSSP's native Kerberos 4 implementation. PSSP scripts automatically manage global and node-specific principals with the respective Kerberos database when the cluster is initially set up or extended.

CSM's security model is very similar to PSSP when Restricted Root Access is enabled. The CSM management server's root user is assumed to have full remote command authorization on every managed node, but no node's root user is required to have automatic root authorization on the management server. From the perspective of a CSM management domain, managed nodes are not aware of each other, unless they also happen to be part of a common RSCT peer domain (RPD). However, a managed node indeed knows that it is a managed node, and, through RMC communication, can trigger certain predefined RSCT resource actions to be executed on its management server.

It has been one of CSM's design goals to offer out-of-the-box security. This means CSM can establish a basic security environment without requiring the administrator to manually set up the configuration.

To this end, unlike PSSP, CSM offers the use of public key cryptography for authentication on the remote command level as well as for RSCT-based cluster internal monitoring and event handling communication, and supports the

administrator through commands that transparently handle public key exchanges where necessary.

In CSM, internal cluster infrastructure communication employed for resource monitoring is self-sufficient and does not depend on the availability of a configured remote command environment. However, on the management server, CSM's dsh distributed shell assumes that remote commands to nodes can be issued without password query, and a few of CSM's administrative commands that are usually run interactively by an administrator in turn rely on dsh (for details see 7.2, "Available remote commands options" on page 154). CSM's remote command authorization model is sketched in Figure 7-3.

Configuration file management with CFM builds upon CSM's "one-to-any" remote command model in that it utilizes an **`rdist`**-based "push" paradigm to distribute configuration files it manages out into the management domain. In contrast, PSSP's file collections employ "pull" technology of client-initiated file transfers from the file repository server, requiring some extra setup.

Currently, CSM does not natively integrate the use of Kerberos for authentication or as a namespace model. To the contrary, CSM does not assume common userids throughout the cluster, but for resource administration by non-root operators offers RSCT's Network Identity Mapping shown in 7.3, "Security for CSM internal cluster components" on page 158.

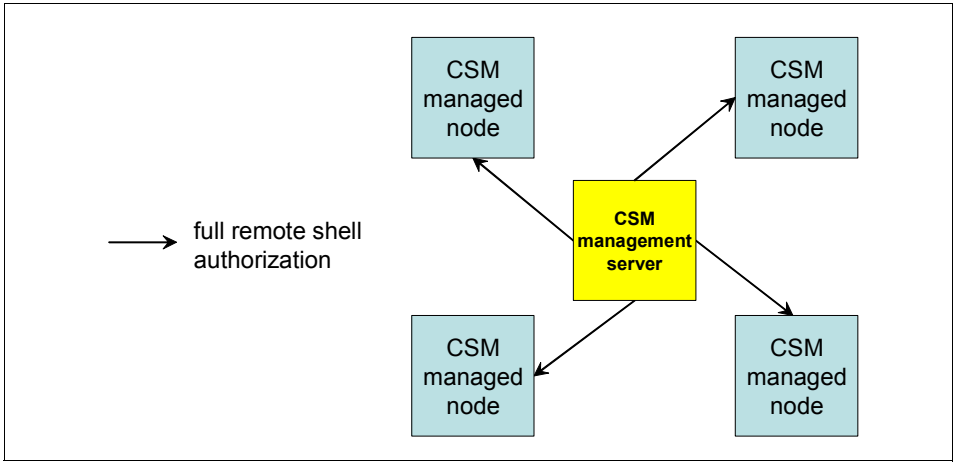


Figure 7-3 CSM remote command authorization

From the viewpoint of network and physical security requirements, in an environment with HMC attached servers, CSM does not pose any demands different from their PSSP counterpart. HMCs function as console servers for serial connections to the LPARs and CECs in full system partition mode as well

as hardware control points. The CSM management server, like the PSSP control workstation, communicates with an HMC through IP to make these serial line console and power control functions available to the cluster administrator on the management server. While the password for login into the HMC does not travel over the network in the clear, all console communication does, and it is equally important to protect the HMC from spoofed power control requests through its network interface.

Thus, for such CSM as well as PSSP environments, an isolated or very strongly protected “trusted” management Ethernet LAN for the connectivity between management server and HMCs is highly recommended (see the section on planning in *An Introduction to CSM 1.3 for AIX 5L,* SG24-6859, and search for “trusted network” in *RS/6000 SP Planning, Volume 2: Control Workstation and Software Environment,* GA22-7281).

AIX features native IP filtering and an IETF-compliant IPSec implementation. You may want to consider using this functionality to control network communication inside the cluster, as well as communication of the cluster with the outside world. It is quite easy to restrict, for example, access to certain network ports on a machine to clients only from a specified subnet. For more information, refer to:

```
http://publib16.boulder.ibm.com/doc_link/en_US/a_doc_lib/aixbman/security/ipsec_intro.htm#ipsec
```

7.2 Available remote commands options

In the following subsections, we discuss alternative mechanisms for remote command execution from the CSM management server.

7.2.1 The rsh and ssh commands

For a long time, PSSP has only offered **rsh** and **rcp** for remote command and copy service. In terms of the underlying authentication mechanism for these “rcmds”, the administrator, through setting the **auth_methods** command for a system partition, can choose from Kerberos 5 in a DCE environment, PSSP-supplied or AFS-integrated Kerberos 4, and standard Berkeley .rhosts files. The latter method, based solely on the IP address and UIDs as provided by the requesting system, gives only very weak authentication. Irrespective of the configured authentication method for **rsh/rcp**, the session data flows over the network in the clear.

As of PSSP 3.4, when running PSSP in RRA and with attribute *auth_root_cmds=none* and remote command method set to *secrshell*, PSSP lets one use one’s favorite secure remote shell mechanism, provided it complies with

the IETF secsh working protocol. This allows for encryption and public key-based authentication, but the cluster administrator has to manually distribute the keyfiles for authorization to the nodes.

In CSM, RSCT-based cluster components do not have an internal dependency on remote shell and copy commands. The following external cluster management commands in CSM, however, require an **rsh**/**rcp** or IETF secsh-compliant remote command implementation:

- dsh
- cfmupdatenode, updatenode (as well as smsupdatenode)
- rmnode

Note: **cfmupdatenode**, by default, is run once a day as a cron job. For **cfmupdatenode** to function properly, remote root command and copy authorization to the target nodes is necessary.

If *RemoteShell=/usr/bin/rsh* and *setupRemoteShell=1* have been set via the **csmconfig** command, CSM transparently configures /.rhosts authorization files when premanaged nodes are adopted into the CSM management domain with the **updatenode** command.

CSM also fully supports strong encryption and public key cryptography-based authentication when using OpenSSH on the management server and managed nodes. When the **updatenode** command is run on premanaged nodes and *setupRemoteShell=1* and *RemoteShell=/usr/bin/ssh*, CSM, unlike PSSP, transparently imports the public host key into the management server's /.ssh/known_hosts file and authorizes remote command access to the node by adding the public DSA, RSA and RSA1 keys of the management server's root user to /.ssh/authorized_keys and /.ssh/authorized_keys2 on the node, respectively.

Adding the node's public host key to the /.ssh/known_hosts file on the management server is only enabled when the node has *AllowManagementRequest=1* in the CSM node database. The import of the node's host key is accomplished by the remoteshell.expect script in /opt/csm/csmbin; this script is also called from the IBM.DMSRM daemon when the management server is notified of a management request from a node that is ready for being managed through RSCT.

Automatic key exchange requires that the keyfile names comply with the conventions used by OpenSSH.

Attention: If your network might be vulnerable to spoofing attacks, you need to manually verify the authenticity of the exchanged keys in the aforementioned files on the management server and managed nodes.

You can run **updatenode -k** to have the CSM management server exchange keys with a node that is already managed. **updatenode** on premanaged nodes queries the operator for the root password of the node if no working SSH authorization for remote access is currently found.

Precompiled binaries as well as source code for OpenSSH can be downloaded from the IBM Web site at:

http://www.ibm.com/servers/aix/products/bonuspack/aix5l/wpcontent.html

OpenSSH can also be found on the AIX Bonus Pack CD-ROM. For detailed instructions on setting up OpenSSH for CSM, search for “OpenSSH installation on AIX” in *An Introduction to Security in a CSM 1.3 for AIX 5L Environment,* SG24-6873 or “Setting up remote access manually” in *CSM for AIX 5L: Software Planning and Installation Guide,* SA22-7919.

Distributed remote commands require non-interactive authorization. Thus, if you want to protect private SSH keys with a passphrase, you need to employ *ssh-agent* and *ssh-add* to provide the SSH client with transparent access to the private keys.

Keep in mind that on AIX, as on many other platforms, *ssh-agent* uses UNIX domain sockets to communicate with the SSH client processes asking for access to the keys, so once keys have been added to an *ssh-agent* session by a user, root or anyone else running under the same UID on the system has easy access to these decrypted keys. Encrypting private keys with a passphrase will also be problematic if you want to run **cfmupdatenode** or other scripts calling **dsh** or **ssh/scp** as cron jobs.

Although using OpenSSH is recommended, it is possible to configure a different remote command/copy suite, you need to establish authorization on the nodes on your own such that root on the management server can issue remote commands to all managed nodes. In this case, you should keep CSM from interfering with your setup by issuing **csmconfig setupRemoteShell=0**. This is just the counterpart of setting *auth_root_cmds=none* in PSSP, keeping PSSP’s updauthfiles script from managing /.rhosts or /.klogin files on the nodes.

7.2.2 Kerberizing rsh/rcp in CSM

CSM is currently not integrated with any Kerberos-based authentication scheme. However, in our residency’s test environment, we found that CSM administration

still worked fine after Kerberizing **rsh**/**rcp** via the Kerberos 5-based Network Authentication Services, which is available on the AIX Expansion Pack CD-ROM.

If you do not need to integrate the CSM cluster into an already established Kerberos environment, Kerberos 5 clients as well as servers can be easily configured using the mkkrb5srv and mkkrb5clnt scripts provided with AIX. For detailed examples refer to the AIX documentation, which can be found at:

http://publib16.boulder.ibm.com/doc_link/en_US/a_doc_lib/aixbman/security/kerberos_auth_only_load_module.htm

When using Kerberos under CSM, it is the administrator's responsibility to enroll principals with the Kerberos database and set up /.k5login or /.klogin files on the managed nodes such that these principals are authorized to issue remote commands to these systems.

You also need to enable remote commands for Kerberos authentication through the AIX **chauthent** command. For more information, refer to:

http://publib16.boulder.ibm.com/doc_link/en_US/a_doc_lib/aixuser/usrcomm/tcp_secure_r.htm

If you use the management server as NIM master for installation of AIX, you may want to keep std configured on the management server as an authentication option for remote commands; refer to 7.7, "Access control relating to NIM" on page 173.

If you plan to run **cfmupdatenode** or other scripts that eventually make calls to **rsh**/**rcp** as cron jobs, you need to ensure that throughout their runtime the UID executing these scripts has valid Kerberos service tickets for the nodes accessed. To temporarily obtain tickets in a background job, it is helpful to first register a principal dedicated for background jobs with the Kerberos database and then stash this principal's Kerberos keys to a keytab file using the **ktadd** command in the **kadmin** facility.

Be careful when adding existing principals to a keytab, because the **ktadd** operation randomizes the principal's Kerberos password before stashing it to the keytab file. A user with read access to the <keytab> file can then obtain a Kerberos ticket-granting-ticket using the **kinit -kt <keytab>** command, which stores the credential in a file specified by the KRB5CCNAME environment variable. Make sure that the temporary credential file is created allowing read access only to the UID under which the background script is run, and that the tickets are destroyed after the script is done.

7.3 Security for CSM internal cluster components

PSSP defines trusted services as cluster management components enabled for the cluster's security mechanisms. For a system partition in PSSP, authentication mechanisms like DCE/Kerberos 5 or the *compat* combination of Kerberos 4 and AIX UID-based authentication for trusted services, such as the SDR daemon, hardmon, sysctl, VSDs, or GPFS, can be configured through the ts_auth_method attribute.

Access control lists for the individual resources listing the principals from the corresponding namespace determine which principals are allowed to modify the resources, or just read their status. A good example for this are the hardmon access control lists in /spdata/sys1/spmon/hmacls.

7.3.1 Host Based Authentication (HBA) in RSCT

In CSM, internal cluster communication (such as heartbeating to determine node liveliness status, as well as event monitoring and responses) is based on RSCT in a management domain. Unlike in an RSCT peer domain, in which every member node communicates with any other peer through RMC and Topology and Group Services, in a CSM management domain-managed nodes communicate with just the management server and only via RMC; see Figure 7-4.

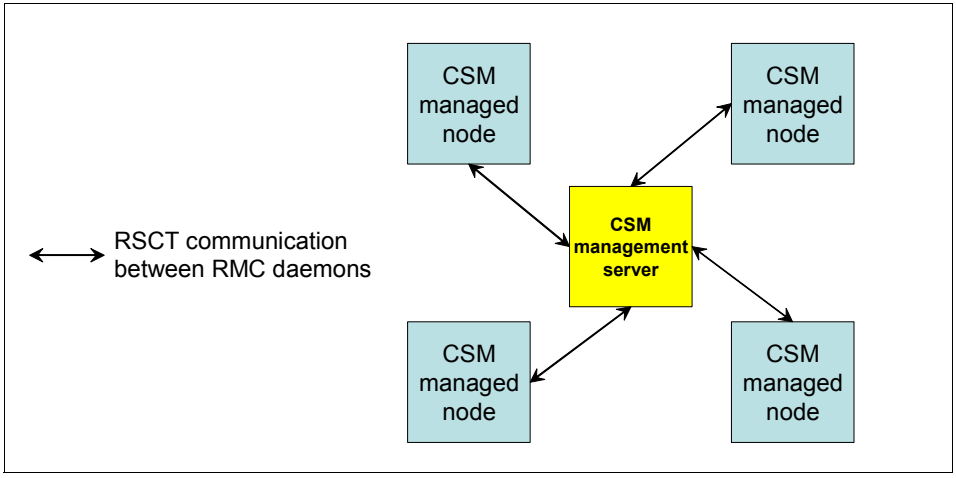


Figure 7-4 RMC communication in a CSM management domain

To stay flexible about authentication options, RSCT's ctsec security service offers the Mechanism Pluggable Module framework to allow for exchangeable components to implement security (see the chapter "Cluster Systems

Management security infrastructure" in *An Introduction to Security in a CSM 1.3 for AIX 5L Environment,* SG24-6873).

At the time of writing, other than a trivial mechanism always yielding unauthenticated contexts, only the Host Based Authentication (HBA) module is available. In HBA, albeit in HBA's specific format, hosts in an RSCT domain are authenticated using RSA public/private key cryptography, very much like authentication of communication partners is accomplished in SSH.

It has been a design goal for HBA to offer out-of-the-box security without requiring manual configuration. When initially setting up HBA for an RSCT domain, every RSCT domain member generates an RSA private/public key pair, stored in /var/ct/cfg/ct_has.qkf and ct_has.pkf (see Figure 7-5 on page 160). CSM exchanges the public host keys of the management server and cluster nodes.

The resulting associations between the public keys and hostnames are stored in binary format in trusted host files in /var/ct/cfg/ct_has.thl. The **ctsthl -l** command can be used to investigate the contents of these files.

These public key exchanges between the management server and managed nodes are carried out by RSCT resource actions when a node is brought into the management domain through the **updatenode** command, or when a node joins the domain after an integrated NIM/CSM installation executes the csmfirstboot script from its /etc/inittab; see Appendix D, "Installation process details" on page 231.

HBA key exchanges between managed node and management server can also be forced by the administrator using **updatenode -k**.

Note: If your network is vulnerable to spoofing, you need to manually verify the associations between hostnames and public keys represented in the /var/ct/cfg/ct_has.thl files. For details on how to do this, refer to "Guarding against address and identity spoofing when transferring public keys" in *RSCT for AIX 5L: Guide and Reference,* SA22-7889. In CSM management domains it is also essential to have consistent hostname resolution in the cluster, be it through DNS or /etc/hosts files.

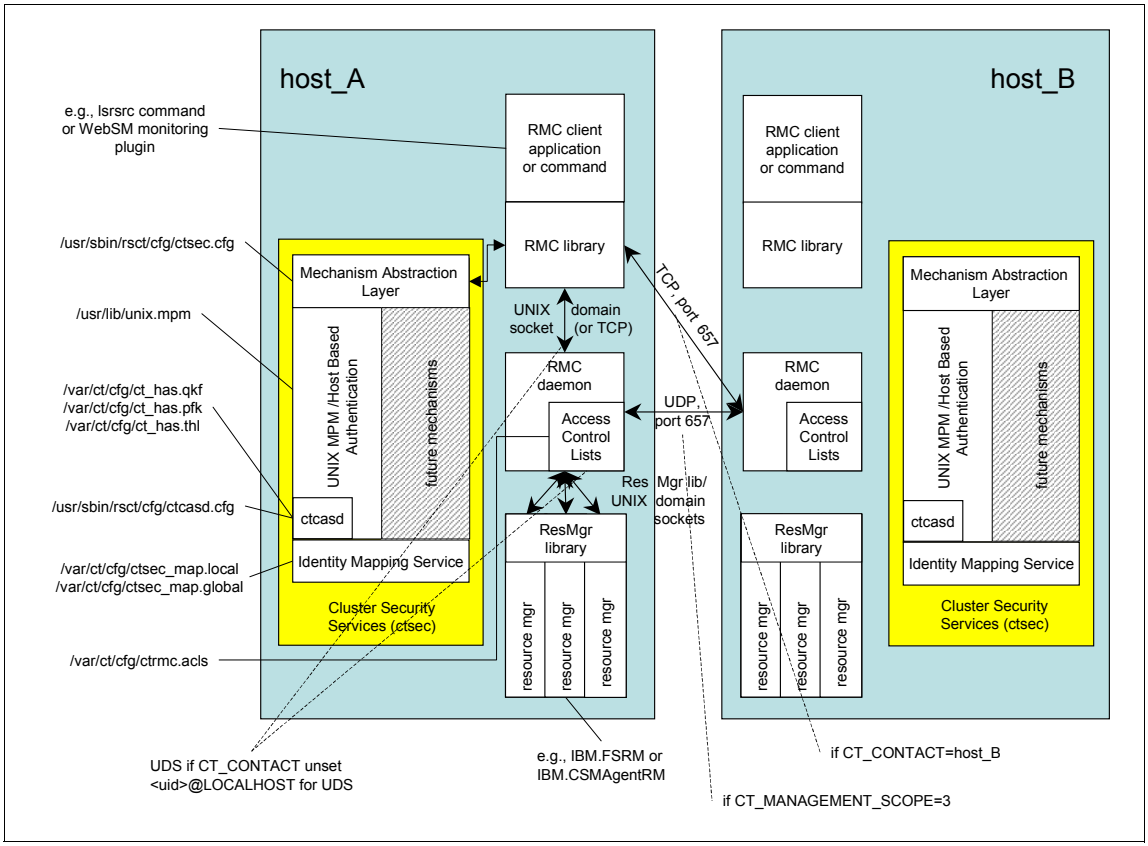


Figure 7-5 RSCT communication flow

When an RMC client (see Figure 7-5) binds against an RMC daemon in the RSCT domain through an AF_INET socket (that is, using an IP address, including 127.0.0.1/loopback/localhost as specified in the CT_CONTACT environment variable) to query or control a remote resource manager through the RMC daemon, HBA attempts to set up a mutually authenticated context between client and RMC daemon. Such a session can be quite short-lived, for example, when running a simple command such as:

```
CT_CONTACT=host_B lsrsrc -Ab IBM.FileSystem
```

The session can also be persistent, for instance when monitoring resources through a Web-based System Manager plug-in. For this, HBA uses the trusted host lists and private key files to establish mutual authentication for the involved hosts. Furthermore, in HBA an RMC client request is associated with the user ID of the owner of the request. The proper authentication of this UNIX ID in turn is credited to the UNIX operating system on the node. For this reason, the HBA

module is also called the “unix.mpm”. In this way, credentials established by HBA as proof of authentication in RMC are attributed to identities of the form <user>@<hostname>.

The necessary public key cryptographic operations in HBA are carried out by the ctcasd daemon. This daemon is automatically started by HBA through the system resource controller, but only when it is needed. ctcasd does not listen on any network socket, it solely interfaces with the local HBA module through UDS.

When an RMC client connects to an RMC daemon on the same host through a UNIX domain socket, HBA simply trusts the authentication of the user’s UNIX ID to the UNIX operating system. In this case, resorting to public key cryptography is not necessary, as no hosts need to be authenticated. The RMC identity of the requester then has the form <user>@nodeid.

Figure 7-5 on page 160 shows that RMC communication to an RMC daemon can in fact take place over yet another channel. While an RMC client can connect (through the RMC library) one-to-one to an RMC daemon using UNIX domain sockets or a TCP session using an AF_INET socket, there also can be RMC daemon-to-RMC daemon communication between hosts using the UDP protocol. For example, if on the CSM management server one issues:

```
CT_MANAGEMENT_SCOPE=3 lsrsrc -Ab IBM.FileSystem
```

The **lsrsrc** command only connects to the management server’s local RMC daemon through a UNIX domain socket. This local RMC daemon then in turn collects the file system information from all the nodes in the management domain by contacting their RMC daemons via UDP communication.

With RSCT 2.3.1 as utilized by CSM 1.3.1, TCP-based communication between an RMC client and an RMC daemon specified through the CT_CONTACT environment variable is enabled to enjoy a mutually authenticated security context established through HBA. This security context also encompasses message signing to assure integrity of all information exchanged.

The **rmcctrl -m** command can be used to enable, require, or disable authentication of RMC communication; see *RSCT for AIX 5L: Technical Reference,* SA22-7890. You can also switch off HBA getting called by ctsec by setting the environment variable CTSEC_CC_MECH=none.

Note: If RSCT commands are run on AIX 5.1 or are from an earlier version of RSCT than 2.3.1, RMC message authentication must not be required via **rmcctrl -mR**, see *RSCT for AIX 5L: Technical Reference,* SA22-7890.

In RSCT 2.3.1, the RMC daemon accepts UDP datagrams on port 657 from any sender. To reduce the possibility of an attack on the RMC daemons, you may want to ensure that the administrative cluster VLAN is non-routed and no machine on this VLAN with other external network interfaces has ipforwarding enabled. In this case, no UDP packets with spoofed IP source addresses can enter this administrative cluster VLAN from an external network. Then use the AIX native IP filtering capability to allow communication to the RMC UDP socket on port 657 only from legitimate IP addresses within this administrative VLAN.

For highest security requirements, the AIX IETF-compliant implementation of IPSec could be used to set up for persistent virtual private network (VPN) tunnels, providing authentication or, if desired, even confidentiality for the UDP RMC daemon-to-daemon communication; note, however, that in RSCT peer domains a node must be able to communicate with any other node in its current cluster. This requires a greater number of tunnels than a management domain, in which there is only a management server with managed node communication, but not managed node to managed node. For more details on setting up the AIX native IP filtering and VPNs using IPSec, refer to:

http://publib16.boulder.ibm.com/doc_link/en_US/a_doc_lib/aixbman/security/ipsec_intro.htm#ipsec

Figure 7-5 on page 160 outlines the steps for setting up filtering so that IP accepts UDP input for the RMC daemon on port 657 only if the datagrams originate from the 192.168.1/24 network, as follows:

- Install the bos.net.ipsec.rte fileset from the AIX BOS CD-ROM.
- Using the **smitty ips4_add_filter** fastpath, assuming that en0 designates the interface to the private cluster VLAN, define a filtering rule that allows UDP datagrams from 192.168.1/24 from port 657 to 657, as in Example 7-1.

Example 7-1 Permit IP filter rule for RMC from private network

```
Add an IP Security Filter Rule

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

                                                      [Entry Fields]
* Rule Action                                        [permit]                +
* IP Source Address                                  [192.168.1.0]
* IP Source Mask                                     [255.255.255.0]
  IP Destination Address                             []
  IP Destination Mask                                []
* Apply to Source Routing? (PERMIT/inbound only)     [yes]                   +
* Protocol                                           [udp]                   +
* Source Port / ICMP Type Operation                  [eq]                    +
```

```
* Source Port Number / ICMP Type                    [657]                 #
* Destination Port / ICMP Code Operation            [eq]                  +
* Destination Port Number / ICMP Type               [657]                 #
* Routing                                           [both]                +
* Direction                                         [both]                +
* Log Control                                       [no]                  +
* Fragmentation Control                             [Whole Packets]       +
* Tunnel ID                                         [0]                   +#
* Interface                                         [en0]                 +
  Description                                       []
```

- Use the same smitty fastpath to define a baseline rule that denies all other incoming UDP datagrams to port 657, as in Example 7-2.

Example 7-2 Baseline deny filter rule for RMC datagrams

```
Add an IP Security Filter Rule

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

                                                    [Entry Fields]
* Rule Action                                       [deny]                +
* IP Source Address                                 [0.0.0.0]
* IP Source Mask                                    [0.0.0.0]
  IP Destination Address                            []
  IP Destination Mask                               []
* Apply to Source Routing? (PERMIT/inbound only)    [yes]                 +
* Protocol                                          [udp]                 +
* Source Port / ICMP Type Operation                 [any]                 +
* Source Port Number / ICMP Type                    [0]                   #
* Destination Port / ICMP Code Operation            [eq]                  +
* Destination Port Number / ICMP Type               [657]                 #
* Routing                                           [both]                +
* Direction                                         [both]                +
* Log Control                                       [yes]                 +
* Fragmentation Control                             [Whole Packets]       +
* Tunnel ID                                         [0]                   +#
* Interface                                         [all]                 +
  Description                                       []
```

- Verify that AIX IP filtering tries to match an incoming IP packet with the just defined “permit” rule *before* the “deny” rule, such that the latter rule has a higher filter ID number, as shown in Example 7-3 on page 164.

Example 7-3 Listing current filter definitions

```
root@sp6cws:/
#> lsfilt -v4 -O
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|al
l packets|0|all|Default Rule
2|*** Dynamic filter placement rule for IKE tunnels ***|no
3|permit|192.168.1.0|255.255.255.0|0.0.0.0|255.255.255.255|yes|udp|eq|657|eq|65
7|both|inbound|no|all packets|0|en0|
4|deny|0.0.0.0|0.0.0.0|0.0.0.0|255.255.255.255|yes|udp|any|0|eq|657|both|both|y
es|all packets|0|all|

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|Default Rule
```

- Finally, start IPSec for IPv4 using the **smitty ips4_start** fastpath.

7.3.2 RMC access control files and identity mapping service

On every host running RSCT, the file /var/ct/cfg/ctrmc.acls determines who is allowed access to a resource class and to all the resource instances of a resource class. “Read” access allows you to list class definitions, to query the state of a resource, and to register (keep parallelism) events related to it, while “write” access enables one to modify resource classes and resources.

Example 7-4 Typical stanza in /var/ct/ctrmc.acls

```
IBM.FileSystem
    root@LOCALHOST * rw
    LOCALHOST * r
    joeuser@joesbox R r
```

The settings in Example 7-4 allow the root user, when connecting to the RMC daemon through a UNIX domain socket, read and write access to the IBM.FileSystem resource class, as well as all file system resources. Anyone connecting through a UNIX domain socket is granted read access to the resource class and resources, while `joeuser`, when connecting through TCP from the host `joesbox` is only allowed read access to the file system resources, but not the IBM.FileSystem resource class.

After ctrmc.acls has been edited, it is necessary to issue **refresh -s ctrmc** to make the RMC daemon pick up the new access definitions.

For more details and examples regarding RMC access control files, refer to *RSCT for AIX 5L: Guide and Reference,* SA22-7889, *A Practical Guide for Resource Monitoring and Control (RMC),* SG24-6615, or *An Introduction to Security in a CSM 1.3 for AIX 5L Environment,* SG24-6873.

In RSCT 2.3.1/CSM 1.3.1, access control can be specified for a specific resource *class* on the one hand, and, on the other hand, for *all* resource instances of the class together. IBM intends to enable access control lists for individual resource instances of a resource class in the future.

Note: RMC resource actions defined as responses to RMC events are executed by the RMC daemon under root authorization. Thus, if based on the RMC access control file a user is entitled to create responses (for example, has read/write access to the IBM.Response resource class), they can effectively run any arbitrary commands with root privileges in such a response.

Upon integration of a node in a management domain, CSM automatically authorizes root on the management server for read/write access to all resource classes and resource instances on the managed node. Any other user on the management server is granted read access only.

Because CSM by itself does not require a common namespace for users throughout the management domain, having available only identities of the form <user>@<hostname> for access control files can become cumbersome in a distributed management environment when individual authorization configuration is necessary for several administrators and operators.

To alleviate this situation, RSCT offers per-host Network Identity Mapping (IDM) services. IDM makes it possible to map identities of the form <user>@<hostname> to " identifiers" (net.id mapped to local/UNIX id) through wildcard expressions not limited to asterisk (*) expressions.

Association rules are configured in the ctsec_map.local and ctsec_map.global files in the /var/ct/cfg directory. As an example, the line in a map file `unix: joeuser@*=operator` associates the network identifier joeuser to the UNIX UID operator irrespective of which host joeuser runs an RMC client application on. Such network identifiers can then be referenced in the ctrmc.acls file's stanzas on any host in the RMC domain; see Example 7-5.

Example 7-5 ctrmc.acls stanza using an IDM network identifier

```
IBM.FileSystem
    root@LOCALHOST * rw
    none:operator * r
```

The **ctsidmck** command can be used to find out the mapping for a network identifier based on the associations in the local map files. For more details on IDM map files and RMC access control lists, refer to *RSCT for AIX 5L: Guide and Reference,* SA22-7889.

Notes:

- The mapping of a network identifier of the form <user>@<hostname> to a local uid is performed on the host on which the RMC daemon that serves the requested resource runs. For example, if a user issues on the management server:

  ```
  CT_MANAGEMENT_SCOPE=3 lsrsrc -Ab -p0 IBM.FileSystem
  ```

  then <user>@<managementserver> ID is only mapped by the map files on the management server, not by those on the managed nodes in the CSM management domain.
- You can use CFM to keep the identity mapping files on all the nodes synchronized.
- The examples given in the “Identity mapping service” section of *An Introduction to Security in a CSM 1.3 for AIX 5L Environment,* SG24-6873, and “Allowing a non-root user to administer CSM” in *An Introduction to CSM 1.3 for AIX 5L,* SG24-6859, are not correct. The <cluster> wildcard token used in these examples does not apply to CSM management domains, only to RSCT peer domains. Also, when referencing network identifiers in the ctrmc.acls file, it is mandatory that these be preceded by the “none:” specifier; see Example 7-5 on page 165.

7.4 Least privilege administration

PSSP’s sysctl facility allows one to run commands and TCL scripts remotely and parallel in the cluster. The sysctl daemon runs with root privileges on all nodes and the control workstation. Each command executed through sysctl can be paired with a callback function that determines whether the requester is authorized to run the command. sysctl authorization files can be specified in terms of Kerberos V4 principals, DCE principals, or (only weakly authenticated) AIX UIDs.

While sysctl is used by internal PSSP processes, in particular when PSSP runs in RRA mode, it also offers an external interface that can be used to implement fine-grained access control to allow non-root administrators and operators to run custom scripts that require root authorization.

CSM 1.3.1 does not feature a direct counterpart to PSSP’s sysctl. As an alternative in the CSM environment, one might consider using the well-known open source tool sudo.

Like sysctl, sudo can run specified commands and programs with root privilege or under a different UID. Note that sysctl does not run commands under a different UID. The /etc/sudoers configuration file (this, of course, is only to be edited by root) determines which commands and command options a specific user is allowed to run through sudo under a different UID. The sudoers configuration file syntax allows for grouping users, setting up aliases for command specifications, and a limited use of matching on wildcards. To help provide an audit trail, sudo can also log to BSD syslog.

Sudo is simply implemented as a local SUID program, while PSSP's sysctl is a continuously running daemon, listening by default on port 6680.

Precompiled binaries, as well as sudo source code, can be found on the AIX toolbox for Linux application CD-ROM for AIX, or downloaded from:

http://www.ibm.com/servers/aix/products/aixos/linux/download.html

Full documentation for sudo is available on sudo's homepage at:

http://www.courtesan.com/sudo/

Attention: Setting up the /etc/sudoers authorization control file requires great care. Understanding its syntax, how it is parsed by sudo and what room wildcards leave (especially when specified to restrict command options) is absolutely essential to avoid unintended backdoors to root authority.

Example 7-6 /etc/sudoers sudo access control file

```
# sudoers file.
# This file MUST be edited with the 'visudo' command as root.
# See the sudoers man page for the details on how to write a sudoers file.

Host_Alias    NODES     = node1, node3, node5, node7, node9, node10
Host_Alias    MGMTSVR   = sp6cws
User_Alias    OPERATORS = joeuser, janeuser

Cmnd_Alias    MKSYSB    = /bin/mksysb -i /backups/mksysb
Cmnd_Alias    RPOWER    = /opt/csm/bin/rpower -n node[13] [!-]*

OPERATORS     NODES     = MKSYSB
janeuser      MGMTSVR   = MKSYSB, RPOWER
```

With the sudoers file as in Example 7-6 in place on managed nodes, possibly distributed and maintained with CFM, joeuser can take mksysbs on all nodes by issuing simply: **dsh -av sudo /bin/mksysb -i /backups/mksysb**.

For another example, recall that at the time of writing, access control to RMC resource managers can only be specified for a resource class and all of its resource instances. Thus, for example, one can only use the ctrmc.acls file to give janeuser read/write access to the IBM.NodeHwCtrl resources (this is required to run the **rpower** command), but cannot further restrict this privilege to apply to only node1 and node3. Using the sudoers file as in Example 7-6, on the management server, janeuser may run, for instance,

```
sudo /opt/csm/bin/rpower -n node3 reboot
```

or

```
sudo /opt/csm/bin/rpower -n node1 query.
```

Note that had the `Cmnd_Alias` definition for `RPOWER` in Example 7-6 instead read:

```
Cmnd_Alias    RPOWER    = /opt/csm/bin/rpower node[13] *
```

sudo would also let janeuser run:

```
sudo /opt/csm/bin rpower -n node3 -N AIXNodes reboot
```

which might not be what had been intended.

Unfortunately, sudo does not take full regular expressions syntax in the /etc/sudoers file. Wildcard matching is done via the POSIX fnmatch function. Back to Example 7-6—this would give us some problems allowing access to the **rpower** command on node1, node3, and node10 through a single command alias.

Tips:

- If you consider using sudo or similar tools, it might be beneficial to think about layout of node naming to allow for efficient wildcard specifications in configuration files.
- To find out about its environment, the dsh script queries the IBM.DmsCtrl resource manager to retrieve the RemoteShell attribute configured for CSM. A non-root user on the management server needs read access to this resource, specify **dsh -r**, or set the DSH_REMOTE_CMD environment variable, otherwise dsh will default to using /usr/bin/rsh. If dsh's `-a` or `-v` options are specified, read access to IBM.ManagedNode is also necessary.

Another technique for command authorization that might come in handy occasionally is *forced command keys* in the SSH Version 2 protocol. A forced command key is nothing more than a command preceding the public key in a user's authorized_keys2 file; see Example 7-7.

Example 7-7 forced command key in /.ssh/authorized_keys2

```
...
command="/bin/mksysb -i /backups/mksysb" ssh-rsa
AAAAB3NzaC1yc2EAAAABJQAAAIBGM3NQfOJOgw9lmKQwKZORjLasD/F8s5N+T14iDJHCOUvwVfuVB3e
zszmDeOc+5Alypc+cDm2vwZFp8u2FGq6h817NJ+IfJ2HUFgF84pdlqI1H/SGp+txrAFaQKef6OjXxlf
YRjhONx4cQXALdRjCYXGIZvbUMuc9pjrDGHB2pAQ== rsa-key-20030612
...
```

If connecting to the system via ssh with the corresponding private key, instead of giving a remote shell, the SSH server just executes the specified command. So, if the forced command key as in Example 7-7 is present in the /.ssh/authorized_keys2 file of all nodes in the AIXNodes nodegroup, to take backups joeuser can issue the following on the management server:

```
dsh -N AIXNodes -r /usr/bin/ssh -o "-i <key.mksysb>" -l root date
```

assuming that the file <key.mksysb> contains the corresponding private key and joeuser has read access to this file. Although the date command is given to be run by dsh, the ssh daemon on the nodes will just execute the forced command defined along with the public key in the authorization file using the UID specified on the ssh call.

7.5 Hardware control and serial connections access

In PSSP, access to the Virtual Front Operator Panel (VFOP) and s1terms through serial connections can be specified per frame or HMC-controlled CEC in the hmacls file in /spdata/sys1/spmon. The settings in this file determine which of the **hmmon**, **hmcmds**, **spmon**, **s1term**, and **nodecond** commands an operator can run.

In CSM, hardware state control is accomplished using the **rpower** command. Serial connections are established with the **rconsole** command.

- **rpower** requires read/write access to IBM.NodeHwCtrl and can, in RSCT 2.3.1/CSM 1.3.1, only be granted or denied globally (for example, not on a per node basis).
- **lshwinfo** requires read access to IBM.HwCtrlPoint.
- **netboot** requires write access to the /var/log/csm/netboot directory and read/write access to IBM.NodeHwCtrl.
- **rconsole** using the **-c** option (which bypasses conserver) requires read access to IBM.ManagedNode and read/write access to IBM.HwCtrlPoint.
- **rconsole** via conserver requires read access to IBM.ManagedNode and access to conserver.

- Binding to conserver directly through a network connection using the **console** client command in /opt/conserver/bin only requires conserver access.

Note: At the time of writing, these access control mechanisms for **rpower, lshwinfo**, **rconsole**, and **netboot** are not correctly reflected in *CSM for AIX 5L: Command and Technical Reference,* SA22-7934.

The conserver daemon runs as root and listens on port 782 for incoming connections initiated through the console client command. Unless the **-c** option is given, CSM's **rconsole** command also connects to conserver through an AF_INET socket. Upon a connection request, conserver directs clients to reconnect to an arbitrary listening socket. Access control to conserver is configured through conserver.cf and conserver.passwd in /etc/opt/conserver.

Example 7-8 /etc/opt/conserver/conserver.cf configuration file

```
LOGDIR=/var/log/consoles
TIMESTAMP=1da
#list of consoles we serve
node1:|/opt/csm/bin/rconsole -t -c -n node1::&
node3:|/opt/csm/bin/rconsole -t -c -n node3::&
node5:|/opt/csm/bin/rconsole -t -c -n node5::&
%%
#list of clients we allow
allowed:sp6cws-en0
trusted:testbox
```

In Example 7-8, the `allowed` keyword specifies that connections to conserver from the listed hosts are allowed, but users need to authenticate their UID token according to the configuration in the conserver.passwd file. The `trusted` keyword means that connection requests from the listed hosts are honored without any further authentication.

Example 7-9 /etc/opt/conserver/conserver.passwd password file

```
root::any
joeuser:*passwd*:node1,node3
```

The conserver password file configuration in Example 7-9 allows any root user on a host with the `allowed` attribute in conserver.cf to connect to any console served without password authentication. joeuser can only connect to the node1 and node3 consoles and needs to authenticate with his UNIX password on the host running the conserver, for example, the management server.

Note: CSM's **netboot** command assumes that the user calling it can open a console via **rconsole** without being prompted for a password.

The default conserver.cf and conserver.passwd files allow any user on the management server to start a console.

Note: If a console has been opened through conserver and interactive login into a node has been accomplished, any user with conserver access can simply take over this console session with write access by issuing **rconsole -f**. Thus, it is advisable to restrict access in conserver.passwd to root and not allow (or even trust) any connections from hosts other than the management server. Alternatively, a write console can be opened bypassing conserver through **rconsole -c.** Such a console session cannot be hijacked through another conserver session for write access, as at any time at most one write console can be opened.

For more details on conserver, refer to:

```
http://www.conserver.com
```

Running the **systemid** command for each HMC specifies the UID (usually hscroot) and the password that the management server uses to connect to the HMC for hardware control commands. A hash of this password is stashed to a file. To erase this information, simply delete the file named with the HMC IP address in the /etc/opt/csm/system_config directory.

7.6 Securing Web-based System Manager

Web-based System Manager offers a graphical user interface to administer and monitor a single AIX node or an entire CSM cluster. Web-based System Manager sessions can also be established from remote clients if the wsmserver daemon is running. Remote clients can be either other Web-based System Manager sessions running on a different AIX system or the standalone Web-based System Manager clients for Microsoft® Windows® or Linux platforms. Web-based System Manager can also be used in applet-mode, just requiring a Web browser on the client side, but this necessitates setting up a Web server on the system which needs to be administrated and monitored.

Upon establishing a remote session, Web-based System Manager queries the client for a user ID and password to connect with. In unsecured mode, this password goes over the network in the clear and might be picked up by sniffers. To provide encryption for the entire Web-based System Manager session as well

as server authentication, for Web-based System Manager applet-mode the employed Web server can be configured to offer Secure Socket Layer (SSL) connections. Native Web-based System Manager connections can be protected using SSL as well. For details on how to set this up, refer to Web-based System Manager documentation at:

http://publib16.boulder.ibm.com/doc_link/en_US/a_doc_lib/aixbman/wsmadmn/enable_sec.htm

By default, wsmserver listens for connection requests (irrespective of using SSL or not) on a TCP socket at port 9090, unless specified otherwise by the `-listenport` option to the **wsmserver** command. When a remote connection request comes in on this port, wsmserver opens another TCP listening socket on an arbitrary port and tells the client to connect to this port. Version and language environment information is exchanged, and the wsmserver switches to listening on yet another arbitrary port to which the client is directed to reconnect. Authentication information and all of the remaining communication in the Web-based System Manager session is then handled through this TCP connection.

For firewalling, it might be desirable to limit the range of ports on which connection requests from remote clients need to be reckoned with. The range of arbitrary ports to which the wsmserver process will redirect clients to issue TCP connection requests to can be restricted using the **-portstart** and **-portend** options of the **wsmserver** command. The specified range needs to be at least three ports wide, and a range of N+1 ports can support N concurrent Web-based System Manager sessions. This limited range of ports can then be configured into a firewall ruleset.

If the firewall allows for SSH traffic, an alternative way of setting up remote Web-based System Manager sessions is to tunnel TCP through SSH connections. A regular Web-based System Manager console can be securely forwarded to a remote X server by SSH's X11 forwarding option.

If X is too demanding for the network bandwidth available, the Virtual Network Computing (VNC) protocol could be of interest. To view a graphics application (such as a Web-based System Manager console) executing on host1 remotely on host2, VNC's **vncserver** process runs a virtual X server on host1 to which the application's graphics output is directed. This virtual screen can be visualized remotely on host2 with the **vncviewer** program through the VNC protocol much leaner than X; it is also possible to share access to a virtual VNC X server with several other vncviewers (to enable this, start vncserver with the -alwaysshared option) or have multiple vncservers running on the same host. For general information on VNC, as well as free downloads of vncviewers for many platforms, including Microsoft Windows and Linux, visit:

http://www.uk.research.att.com/vnc

Ready-to-run vncserver and vncviewer programs for AIX can be found for free downloads at:

http://www.ibm.com/servers/aix/products/aixos/linux/download.html

To connect to a virtual X vncserver session, a vncviewer client needs to authenticate with a password specific for the UID running the vncserver instance. This password is set the first time a user starts a vncserver. Authentication in the VNC protocol uses a challenge-response algorithm, so when a vncviewer attempts to connect, the password does not flow over the network in the clear. All subsequent communication for screen updates and remote input, though, is not encrypted and might be susceptible to sniffing. To protect this VNC TCP communication and traverse a firewall, one can again use TCP forwarding through SSH tunnels. A nice and short tutorial for this is given at:

http://www.uk.research.att.com/vnc/sshvnc.html

For moderately changing graphics output such as Web-based System Manager consoles or PSSP perspectives, the author has found tunneling VNC through SSH very workable for 768 kb/s download or 128 kb/s upload DSL connections, and still not useless for 56 kb/s modem dial-up.

7.7 Access control relating to NIM

NIM requires that, during operations that involve a NIM client, the NIM master can run remote commands on the client. At the time of writing, remote command execution in NIM is only attempted through the /usr/bin/rsh command; this means that on the client, a corresponding authorizing entry in the /.rhosts file, or, if Kerberos is configured on the NIM master and NIM client, in the /.k5login file, is necessary.

During a NIM installation, the NIM scripts automatically create an /.rhosts file on the NIM client, enabling remote command execution from the NIM master. Also, Kerberos services are not available during an installation. Thus, even when you are exclusively using Kerberos in your environment for remote command authentication, the NIM master must be configured to attempt standard AIX authentication through /.rhosts during the installation of a node. For example, the *std* method must be listed in the output of the **lsauthent** command. Of course, this does not require that a /.rhosts file be present on the NIM master itself.

Attention: Unlike PSSP, after a NIM installation, CSM does not automatically remove the /.rhosts file created during the NIM installation process. You may want to remove it manually if you use ssh for remote commands. Keep in mind, however, that NIM itself cannot make use of ssh.

NIM clients can pull resources from the NIM master using the **nimclient** command. To disallow all clients to use resources, on the NIM master, issue:

```
nim -o change -a client_alloc=no master
```

To prevent a single client from pulling resources, use:

```
nim -o change -a client_alloc=no clientname
```

It is also possible to protect particular resources from client allocation via:

```
nim -o change -a client_alloc=no resourcename
```

Specifying “yes” for the client_alloc attribute in the previous commands reenables client resource allocation.

Running **nim -o change -a client_reg=no master** disables arbitrary AIX systems from registering themselves with the NIM master as NIM clients using the **niminit** command.

nimclient -P, issued on a NIM client, disables the NIM master from pushing operations onto this client. However, the NIM master can override this setting by using the **-F** force flag on NIM commands.

A

Command cross-references

In this appendix, we list the commands that are available in PSSP version 3.5, as described in *PSSP Command and Technical Reference (2 Volumes),* SA22-7351 and provide some cross-reference information for the “CSM world” as shown in Table A-1 on page 176.

If the PSSP command:

- Has a functional equivalent in the CSM world, we list that command together with the component that provides it, which may be CSM, RSCT, or AIX 5L.
- Relates to a feature that is (or will be) provided by a separate product in the CSM world, we mark the command as “*n/a*” and refer to that product.
- Is discussed in detail in this redbook, we provide a cross-reference to the section that covers it.

Note that we do not cover any of the SP partitioning commands since these do not apply to the “CSM world”, and also because SP partitioning is a legacy feature not widely used in PSSP-based clusters.

Table A-1 PSSP command cross-references

PSSP command	CSM command	CSM component	Notes
add_principal	*n/a*	CtSec (RSCT)	*no more Kerberos4*
allnimres		NIM	nim -o allocate nim -o bos_inst
arp	arp	AIX	
cfghsd	*n/a*	VSD	HSD dropped
cfghsdvsd	*n/a*	VSD	HSD dropped
cfgvsd	*n/a*	VSD	
chauthpar	dsh chauthent	AIX	
chauthpts	*n/a*	CtSec (RSCT)	Always HBA
chauthts	*n/a*	CtSec (RSCT)	Always HBA
chgcss	*n/a*	n/a	
chkp	*n/a*	CtSec (RSCT)	*no more Kerberos4*
cmonacct			SP accounting
config_spsec	*n/a*	CtSec (RSCT)	DCE SP security
cprdaily			SP accounting
cptuning			May use NIM script resources
create_dcehostname	*n/a*	CtSec (RSCT)	DCE SP security
create_keyfiles	*n/a*	CtSec (RSCT)	*no more Kerberos4*
create_krb_files	*n/a*	CtSec (RSCT)	*no more Kerberos4*
createhsd	*n/a*	VSD	HSD dropped
createvsd	*n/a*	VSD	
crunacct			SP accounting
cshutdown			See Appendix B, “Cluster startup and shutdown” on page 193
CSS_test	*n/a*	n/a	

PSSP command	CSM command	CSM component	Notes
css.snap	*n/a*	n/a	Refer to Section 5.11, “Additional administration tools (snap commands)” on page 110.
css.vpd	*n/a*	n/a	
cstartup			See Appendix B, “Cluster startup and shutdown” on page 193.
ctlhsd	*n/a*	VSD	HSD dropped
ctlvsd	*n/a*	VSD	
defhsd	*n/a*	VSD	HSD dropped
defvsd	*n/a*	VSD	
delnimclient		NIM	nim -Fo reset <node> nim -Fo deallocate -a subclass=all <node> nim -o remove <node> nim -o remove <bosinst_data> rm <files>
delnimmast		NIM	nim -o remove *<spot>* nim -o unconfig master rm *<files>* installp -u bos.sysmgmt.nim.*
dsh	dsh	CSM	
dshbak	dshbak	CSM	
dsrvtgt	*n/a*	CtSec (RSCT)	DCE SP sec
Eannotator	*n/a*	n/a	No more E commands with the High Performance Switch.
Eclock	*n/a*	n/a	No more E commands with the High Performance Switch.

PSSP command	CSM command	CSM component	Notes
Efence	*n/a*	n/a	No more E commands with the High Performance Switch.
Emaster	*n/a*	n/a	No more E commands with the High Performance Switch.
emconditionctrl	*n/a*	ERRM (RSCT)	
emonctrl	*n/a*	n/a	
Emonitor daemon	*n/a*	n/a	No more E commands with the High Performance Switch.
enadmin	*n/a*		extension node (alas 9077)
endefadapter	*n/a*		extension node (alas 9077)
endefnode	*n/a*		extension node (alas 9077)
enrmadapter	*n/a*		extension node (alas 9077)
enrmnode	*n/a*		extension node (alas 9077)
Eprimary	*n/a*	n/a	No more E commands with the High Performance Switch.
Equiesce	*n/a*	n/a	No more E commands with the High Performance Switch.
Estart	*n/a*	n/a	No more E commands with the High Performance Switch.
Etopology	*n/a*	n/a	No more E commands with the High Performance Switch.
Eunfence	*n/a*	n/a	No more E commands with the High Performance Switch.

PSSP command	CSM command	CSM component	Notes
Eunpartition	*n/a*	n/a	No more E commands with the High Performance Switch.
export_clients	csmsetupnim	NIM	exportfs *<pssplpp>* exportfs *<lppsource>*
ext_srvtab	*n/a*	CtSec (RSCT)	*no more Kerberos4*
fencevsd	*n/a*	VSD	
get_keyfiles	*n/a*	CtSec (RSCT)	*no more Kerberos4*
get_vpd			Type/model/serial# stored in SR, else use dsh and lshwinfo
ha_vsd	*n/a*	VSD	
ha.vsd	*n/a*	VSD	
haadm	n/a	n/a	
hacws_verify	*n/a*		HACWS
haemcfg	*n/a*	ERRM (RSCT)	
haemctrl	rmcctrl	ERRM (RSCT)	
haemd daemon	rmcd haemd (compat)	ERRM (RSCT)	
haemd_SP	*n/a*	ERRM (RSCT)	
haemloadcfg	*n/a*	ERRM (RSCT)	
haemqvar	haemqvar (compat)	ERRM (RSCT)	
haemtrcoff	haemtrcoff (compat)	ERRM (RSCT)	
haemtrcon	haemtrcon (compat)	ERRM (RSCT)	
haemunlkrm	haemunlkrm (compat)	ERRM (RSCT)	
hagsctrl	cthagsctrl	HAGS (RSCT)	
hagsd daemon	hagsd	HAGS (RSCT)	
hagsglsmd daemon	*n/a*	n/a	SP Switch adapter membership
hagsns	hagsns	HAGS (RSCT)	

PSSP command	CSM command	CSM component	Notes
hagsvote	hagsvote	HAGS (RSCT)	
hardmon daemon	IBM.HWCTRLRM cspd	CSM	RM starts cspd for CSP hardware support
hats		HATS (RSCT)	
hatsctrl	cthatsctrl	HATS (RSCT)	
hatsoptions	hatsoptions	HATS (RSCT)	
hatstune	cthatstune	HATS (RSCT)	
hmadm	cspadm (CSP only)	CSM	SP hardmon
hmckacls	vi `/var/ct/cfg/ctrmc.acls`	RSCT	SP hardmon
hmcmds	rpower cspcmds (CSP only)	CSM	SP hardmon
hmdceobj	*n/a*	CtSec (RSCT)	SP hardmon SP DCE sec
hmgetacls	cat `/var/ct/cfg/ctrmc.acls`	CtSec (RSCT)	SP hardmon
hmmon	cspmon (CSP only)	CSM	SP hardmon
hmreinit	*n/a*	CSM	Should be done automatically through RMC event notification
hostlist	lsnode	CSM	Tweaking WCOLLs
hr	*n/a*	RSCT	host_responds is Status in CSM; RMC heartbeating
hrctrl	rmcctrl	RSCT	
hsdatalst	*n/a*	VSD	HSD dropped
hsdvts	*n/a*	VSD	HSD dropped
ifconfig	ifconfig	AIX	
install_cw	installms	CSM	SMIT panels, SDR, PSSP daemons, ODM nodenum=0, tuning.default

PSSP command	CSM command	CSM component	Notes
install_hacws	*n/a*	--	HACWS
kadmin	*n/a*	CtSec (RSCT)	*no more Kerberos4*
kadmind daemon	*n/a*	CtSec (RSCT)	*no more Kerberos4*
kdb_destroy	*n/a*	CtSec (RSCT)	*no more Kerberos4*
kdb_edit	*n/a*	CtSec (RSCT)	*no more Kerberos4*
kdb_init	*n/a*	CtSec (RSCT)	*no more Kerberos4*
kdb_util	*n/a*	CtSec (RSCT)	*no more Kerberos4*
k4destroy(kdestroy)	*n/a*	CtSec (RSCT)	*no more Kerberos4*
kerberos daemon	*n/a*	CtSec (RSCT)	*no more Kerberos4*
kfserver	*n/a*	CtSec (RSCT)	*no more Kerberos4*
k4init(kinit)	*n/a*	CtSec (RSCT)	*no more Kerberos4*
k4list(klist)	*n/a*	CtSec (RSCT)	*no more Kerberos4*
kpasswd	*n/a*	CtSec (RSCT)	*no more Kerberos4*
kprop	*n/a*	CtSec (RSCT)	*no more Kerberos4*
kpropd daemon	*n/a*	CtSec (RSCT)	*no more Kerberos4*
ksrvtgt	*n/a*	CtSec (RSCT)	*no more Kerberos4*
ksrvutil	*n/a*	CtSec (RSCT)	*no more Kerberos4*
kstash	*n/a*	CtSec (RSCT)	*no more Kerberos4*
lppdiff	dsh lslpp	AIX	
lsauthpar	dsh lsauthent	AIX	r-cmds
lsauthpts	*n/a*	CtSec (RSCT)	Always HBA
lsauthts	*n/a*	CtSec (RSCT)	Always HBA
lsvsd -l	*n/a*	VSD	
lshacws	*n/a*		HACWS
lshsd	*n/a*	VSD	HSD dropped
lskp	*n/a*	CtSec (RSCT)	*no more Kerberos4*
lsvsd	*n/a*	VSD	

PSSP command	CSM command	CSM component	Notes
mkamdent			usermgmt $HOME
mkautomap			usermgmt $HOME
mkconfig	csmsetupnim	CSM	config_info
mkinstall	csmsetupnim	CSM	install_info
mkkp	*n/a*	CtSec (RSCT)	*no more Kerberos4*
mknimclient	csm2nimnodes (csm2nimgrps)	NIM	
mknimint		NIM	NIM's netname
mknimmast	installp -a bos.sysmgmt.nim	NIM	See step 6 in chapter 5 of *CSM for AIX 5L: Software Planning and Installation Guide,* SA22-7919
mknimres		NIM	nim -o define
mult_senders_test	*n/a*	n/a	
ngaddto	nodegrp	CSM	
ngclean	nodegrp	CSM	
ngcreate	nodegrp	CSM	
ngdelete	nodegrp -D	CSM	
ngdelfrom	nodegrp -x	CSM	
ngfind	nodegrp	CSM	
nglist	nodegrp	CSM	
ngnew	nodegrp	CSM	
ngresolve	nodegrp	CSM	
nlssrc		AIX	en_US locale lssrc, for PSSP-RSCT
node_number	lsrsrc IBM.ManagementServer LocalHostname		PSSP node number in local ODM
nodecond	netboot	CSM	SP hardmon

PSSP command	CSM command	CSM component	Notes
nrunacct			SP accounting
p_cat	dsh cat		SP p-cmds
pcp	shell loop using rcp/scp		SP p-cmds (check dcp on alphaworks)
pdf	dsh df		SP p-cmds
penotify	use dsh	AIX	AIX errpt
perspectives	wsm	AIX, CSM	Web-based System Manager
pexec			SP p-cmds
pexscr			SP p-cmds
pfck	dsh df		SP p-cmds
pfind	dsh find		SP p-cmds
pfps	dsh ps & action		SP p-cmds
pls	dsh ls		SP p-cmds
pmanchown	*n/a*	ERRM (RSCT)	
pmanctrl	rmctrl	ERRM (RSCT)	
pmandef	mkcondition mkresponse mkcondresp	ERRM (RSCT)	
pmanquery	lscondition lsresponse lscondresp	ERRM (RSCT)	
pmanrmdloadSDR	*n/a*	ERRM (RSCT)	
pmanrminput	*n/a*	ERRM (RSCT)	
pmv	dsh mv		SP p-cmds
ppred	dsh “if”		SP p-cmds
pps	dsh ps		SP p-cmds
preparevsd	*n/a*	VSD	
prm	dsh rm		SP p-cmds

PSSP command	CSM command	CSM component	Notes
psyslclr			logmgmt syslog
psyslrpt			logmgmt syslog
rcmdtgt	*n/a*	CtSec (RSCT)	*no more Kerberos4*
removehsd	*n/a*	VSD	HSD dropped
removevsd	*n/a*	VSD	
resource_center	netscape		
resumevsd	*n/a*	VSD	
rm_spsec	*n/a*	CtSec (RSCT)	DCE SP security
rmkp	*n/a*	CtSec (RSCT)	*no more Kerberos4*
rvsdrestrict	*n/a*	VSD	
SDR_config		SR (RSCT)	
SDR_init		SR (RSCT)	
SDR_test	probemgr	SR (RSCT)	
SDRAddSyspar	*n/a*		Partitioning
SDRArchive		SR (RSCT)	CSM has documented process only, see chapter 10 of *CSM for AIX 5L: Software Planning and Installation Guide,* SA22-7919.
SDRChangeAttrValues	chrsrc (persistent attributes only)	SR (RSCT)	
SDRClearLock		SR (RSCT)	
SDRCreateAttrs	chrsrc	SR (RSCT)	
SDRCreateClass		SR (RSCT)	User-defined classes not supported
SDRCreateFile	*n/a*		No files in SR
SDRCreateObjects	mkrsrc	SR (RSCT)	
SDRCreateSystemClass		SR (RSCT)	User-defined classes not supported

PSSP command	CSM command	CSM component	Notes
SDRCreateSystemFile	*n/a*		No files in SR
SDRDeleteFile	*n/a*		No files in SR
SDRDeleteObjects	rmrsrc	SR (RSCT)	
SDRGetObjects	lsrsrc lsrsrcdef	SR (RSCT)	
SDRListClasses	lsrsrc -c lsrsrcdef -c	SR (RSCT)	
SDRListFiles	*n/a*		No files in SR
SDRMoveObjects	*n/a*		Partitioning
SDRRemoveSyspar	*n/a*		Partitioning
SDRReplaceFile	*n/a*		No files in SR
SDRRestore		SR (RSCT)	CSM has documented process only, see chapter 10 of *CSM for AIX 5L: Software Planning and Installation Guide,* SA22-7919.
SDRRetrieveFile	*n/a*		No files in SR
SDRScan		SR (RSCT)	Searches for non-ASCII data in SDR
SDRSetTsAuth	*n/a*	SR (RSCT)	Always HBA
SDRValidateString		SR (RSCT)	
SDRWhoHasLock		SR (RSCT)	
seqfile			cstartup configuration, see Appendix B, “Cluster startup and shutdown” on page 193.
services_config			
sethacws	*n/a*		HACWS
setsuppwd	*n/a*	CFM (CSM)	CFM runs as root

PSSP command	CSM command	CSM component	Notes
setup_authent	*n/a*	CtSec (RSCT)	*no more Kerberos4*; refer to AIX K5 server.
setup_CWS		CSM	
setup_logd			Installation helper
setup_server		NIM, CSM	See Appendix C, “NIM examples” on page 209.
setupdce	*n/a*		DCE SP security
sp_configd daemon	snmpevent (action script)	ERRM (RSCT)	SNMP agent
sp_configdctrl	snmpevent (action script)	ERRM (RSCT)	SNMP agent
spacctnd	*n/a*		SP accounting
spacl	*n/a*		DCE SP security
spacs_cntrl	dsh chsec -f /etc/security/user -s <user> -a login=false -a rlogin=false	AIX	Restrict interactive login on nodes
spadaptr_loc	getadapters	CSM	Get location codes and MAC address for HMC-controlled nodes (uses nodecond)
spadaptrs	chnode getadapters (v1.3.2) nimadapters	CSM, NIM	
spaggip	*n/a*	n/a	SP Switch2 (ml IF to SDR)
spapply_config	*n/a*		Partitioning
spauthconfig			bootscript helper (rc.sp, firstboot)
spbootins	nim -o bos_inst	NIM, CSM	
spbootlist	dsh bootlist	AIX	
spchuser	*n/a*		User management, $HOME

PSSP command	CSM command	CSM component	Notes
spchvgobj		NIM, CSM	Must be done manually
spcustomize_syspar	*n/a*		Partitioning
spcw_addevents	*n/a*		HACWS/HACMP
spcw_apps	*n/a*		HACWS
spdeladap	chnode	CSM	Inverse of spadaptrs
spdelagg	*n/a*	n/a	SP Switch2, inverse of spaggip
spdelexp	*n/a*		Expansion nodes
spdelfram	rmnode	CSM	
spdelhmcid	rm /etc/opt/csm/system_config/<ip_addr>	CSM	HMC pwd
spdelnode	rmnode	CSM	
spdisplay_config	*n/a*		Partitioning
spethernt	*n/a*		Obsolete in PSSP, see spadaptrs
spevent	wsm	AIX, ERRM (RSCT)	Perspectives
spframe	lshwinfo definenode	CSM	No frame concept in CSM, but hardware control point (HCP)
spget_syspar	*n/a*		Partitioning
spgetdesc		CSM	HW descriptions to SDR
spgrpname	*n/a*		DCE SP security
sphardware	wsm	AIX, CSM plugin	Perspectives
sphmcid	systemid	CSM	HMC password
sphostnam			Tweak hostname<=>interface binding & short/long
sphrdwrad	getadapters	CSM	Store MAC address. NOTE: CSM has no bootptab.info file

PSSP command	CSM command	CSM component	Notes
spled	csmstat -l		Perspectives LED/LCD
splm			Log management
splogd daemon	IBM.AuditRM	RSCT	SP hardmon
splst_syspars	*n/a*		Partitioning
splst_versions	lsnode -a InstallCSMVersion, CSMVersion	CSM	Effectively an SDRGetObjects Node code_version | sort -u
splstadapters	lsnode	CSM	
splstdata	lsnode lsrsrc	CSM, RSCT	See 4.5, “Listing configuration data: splstdata equivalents” on page 76
splstnodes	lsnode	CSM	
splsuser	*n/a*		User management
spmgrd daemon	*n/a*		Extension node
spmirrorvg			Must be done manually
spmkuser	*n/a*		User management
spmkvgobj			Must be done manually
spmon	csmstat (v1.3.2)	CSM	SP hardmon
spmon_ctest			
spmon_itest			
spnkeyman_start	*n/a*		SP DCE security
sprebuildsysmap	*n/a*		Partitioning
sprestore_config	*n/a*		Partitioning
sprmuser	*n/a*		User management
sprmvgobj			Must be done manually
spseccfg	*n/a*	CtSec (RSCT)	DCE SP security
spsetauth	csmconfig	CSM	
spsitenv	csmconfig	CSM	

PSSP command	CSM command	CSM component	Notes
spsvrmgr	cspmon -Qsv codeVersion {-C ttyl-n nodename}		SP supervisor ucode status CSP only
spswplane	*n/a*	n/a	SP Switch2
spsyspar	*n/a*		Partitioning, Perspectives
sptgtprin	*n/a*	CtSec (RSCT)	DCE SP security
spunmirrorvg			Must be done manually
spverify_config			
spvsd	n/a	VSD	No equivalent function in CSM
st_clean_table	*n/a*	n/a	
st_status	*n/a*	n/a	
st_verify	*n/a*	n/a	
startvsd	*n/a*	VSD	
statvsd	*n/a*	VSD	
stopvsd	*n/a*	VSD	
supfilesrv daemon	*n/a* (uses rdist)	CFM (CSM)	
supper	cfmupdatenode	CFM (CSM)	/var/sysman/supp
suspendvsd	*n/a*	VSD	
switch_stress	*n/a*	n/a	
sysctl	*n/a*	n/a	Least privilege
sysctld daemon	*n/a*	n/a	Least privilege
SYSMAN_test			
syspar_ctrl	*n/a* (use rmcctrl + SRC)		Controls partition-sensitive subsystems
sysparaid	*n/a*		Partitioning
s1term	rconsole	CSM	SP hardmon

PSSP command	CSM command	CSM component	Notes
tecad_pssp	*n/a*		PSSP->Tivoli®; in CSM use SNMP traps
ucfghsd	*n/a*	VSD	HSD dropped
ucfghsdvsd	*n/a*	VSD	HSD dropped
ucfgvsd	*n/a*	VSD	
unallnimres		NIM	See Example 3-11 on page 45
undefhsd	*n/a*	VSD	HSD dropped
undefvsd	*n/a*	VSD	
unfencevsd	*n/a*	VSD	
updatehsd	*n/a*	VSD	HSD dropped
updatevsdnode	*n/a*	VSD	
updatevsdtab	*n/a*	VSD	
updatevsdvg	*n/a*	VSD	
updauthfiles	updatenode	CSM	create/update authorization files (.rhost / .k[5]login)
updsuppwd	*n/a*	CFM (CSM)	
usesuppwd	*n/a*	CFM (CSM)	
vhostname	*n/a*		HACWS
vsdatalst	*n/a*	VSD	
vsdchgserver	*n/a*	VSD	
vsdelnode	*n/a*	VSD	
vsdelvg	*n/a*	VSD	
vsdnode	*n/a*	VSD	
vsdsklst	*n/a*	VSD	
vsdvg	*n/a*	VSD	
vsdvgts	*n/a*	VSD	
wrap_test	*n/a*	n/a	

B

Cluster startup and shutdown

This appendix describes an approach for performing controlled *shutdowns* and *startups* of nodes in a CSM cluster, which is a function performed by **`cshutdown`** and **`cstartup`** in PSSP.

We make use of the ERRM and Sensor functions of RMC to implement this; the method utilized is described.

B.1 Monitoring node status for startup and shutdown

To provide an equivalent **cstartup** and **cshutdown**, the script needs to know when the nodes being processed have restarted, and when they have shut down. This is necessary to process dependent nodes.

PSSP provides the **spmon** command, which can be used to monitor conditions on the nodes. When first called, it will output the current values to STDOUT and will output further values as they change.

This processing can be emulated using ERRM by defining conditions for the values we want to monitor and having a response that will pass the data to a central point for processing. We can also add our own values to indicate when the startup of the application on the node has completed. This allows additional intelligence in the startup processing.

B.2 Detecting node startup completion

The PSSP **cstartup** command assumes that a node has started once it is participating in the PSSP cluster. This means that the hostResponds value is set to 1. Once this happens, the next set of dependent nodes is powered on.

If the node is running an application that requires additional time to start up, it is possible that the dependent nodes will boot and issue client requests before the application is ready to serve them.

The RSCT infrastructure provides an IBM.Sensor resource class that allows scripts to be run to check the status of a system component. The output from the script determines the value of the attributes of the IBM.Sensor instance for that sensor.

For this exercise, we have defined a sensor called ServiceRunning (see Example B-1 for the command used to do this). This sensor has to be defined separately on each node so the script that performs the check can be different on each system. The alternative is to vary the processing of the script based upon the node where it is being executed.

Example: B-1 Defining the ServiceRunning sensor

```
$ mksensor -n node1 ServiceRunning /usr/local/bin/CheckService
$ lssensor -n node1 ServiceRunning
 Name = ServiceRunning
 Command = /usr/local/bin/CheckService
 ConfigChanged = 0
 Description =
```

```
 ErrorExitValue = 1
 ExitValue = 0
 Float32 = 0
 Float64 = 0
 Int32 = 0
 Int64 = 5
 NodeNameList = {node1}
 RefreshInterval = 60
 String =
 Uint32 = 1
 Uint64 = 0
 UserName = root
$
```

We defined the sensor using default values so that the check is performed every 60 seconds. The code that performs the check should be as lean as possible to prevent general system overhead. If it cannot perform a fast check, then the RefreshInterval should be set to a higher value.

For testing purposes, a script was used that indicated the application was down for 5 minutes after the reboot; this script is shown in Example B-2.

Example: B-2 Script to simulate a delayed application startup

```
#!/bin/ksh
#-------------------------------------------------------------------------------#
#                                                                               #
#   CheckService : Sample code to check service status                          #
#                                                                               #
#   Function : Demonstrate the use of a sensor to indicate when a service       #
#              has started.                                                     #
#                                                                               #
#              The code will maintain a count of the number of calls, when      #
#              the count reaches 5 it will indicate that the Service is up      #
#              Until then it will indicate that the service is down.            #
#                                                                               #
#   Attributes: The attributes used by this sensor are:                         #
#                                                                               #
#               o Uint32 - set to 1 if the service is up                        #
#               o Int64  - Incremented on each call until the service is up     #
#                          Just updated to indicate progress                    #
#                                                                               #
#   Note: The sensor can't interrogate it's own values as each lsrsrc for an    #
#         IBM.Sensor resource will run the sensor, causing recursion.           #
#                                                                               #
#-------------------------------------------------------------------------------#
me=$(basename $0)
```

```
#
#-------------Use the Parent Process id as an indicator of a reboot-----------#
#---Note: a restart of the IBM.Sensor RM will be seen as a reboot also--------#
   FILEBASE="/tmp/${me}.Sensor."
   FILE="${FILEBASE}${PPID}"

   if [ -e "${FILE}" ] ; then
      cat ${FILE} 2>/dev/null| read count rest
   else
      rm ${FILEBASE}* 2>/dev/null
      count=0
   fi

   if [ "${count}" -lt 5 ] ; then
      let count=count+1
      echo "Int64=${count} \c"
      status="0"
   else
      status="1"
   fi

   echo "${count}" > ${FILE}
   echo "Uint32=${status}"

exit 0
```

Note: When a sensor is executed, the code has no access to the values returned on the previous call. Do not use the **lsrsrc** command to access this data because this will result in the sensor being called again. The processes will need to be killed to get the sensor resource manager running normally again.

B.3 Responding to status changes

In a PSSP environment, the **spmon** command can be used to monitor the state of the nodes. The command format and sample output for this are shown in Example B-3.

Example: B-3 spmon monitoring output

```
$ spmon -query -M -l node*/hostResponds/value node*/nodePower/value
/SP/frame/frame1/node14/hostResponds/value/0
/SP/frame/frame1/node13/hostResponds/value/0
/SP/frame/frame1/node12/hostResponds/value/0
/SP/frame/frame1/node11/hostResponds/value/0
/SP/frame/frame1/node10/hostResponds/value/1
```

```
/SP/frame/frame1/node9/hostResponds/value/1
/SP/frame/frame1/node7/hostResponds/value/1
/SP/frame/frame1/node5/hostResponds/value/1
/SP/frame/frame1/node3/hostResponds/value/1
/SP/frame/frame1/node1/hostResponds/value/1
/SP/frame/frame1/node1/nodePower/value/1
/SP/frame/frame1/node3/nodePower/value/1
/SP/frame/frame1/node5/nodePower/value/1
/SP/frame/frame1/node7/nodePower/value/1
/SP/frame/frame1/node9/nodePower/value/1
/SP/frame/frame1/node10/nodePower/value/1
/SP/frame/frame1/node11/nodePower/value/1
/SP/frame/frame1/node12/nodePower/value/1
/SP/frame/frame1/node13/nodePower/value/1
/SP/frame/frame1/node14/nodePower/value/1
```

When the command is executed, the current status of the requested values is shown, and as soon as the status changes a new line is put out. There is no CSM equivalent to this, but it is possible to emulate the processing using RSCT and ERRM. The emulation is more flexible since SensorRM values can be used in the monitored conditions. With **spmon**, only predefined PSSP values are available.

B.3.1 Defining the monitored conditions

The main conditions we want to monitor are in the IBM.ManagedNode class. A map with data supplied by PSSP is shown in Table B-1.

Table B-1 PSSP status values mapped to IBM.ManagedNode class attributes

PSSP value	IBM.ManagedNode attribute	Comment
hostResponds	Status	This indicates that the node is participating as part of the cluster. PSSP uses hats to monitor cluster membership, CSM uses RSCT.
nodePower	PowerStatus	This indicates that the node has power. PSSP gets this information from hardmon, CSM gets it from the hardware control point for the node. If there is no HCP for the node the value will not be 0 or 1.

The ERRM conditions for IBM.ManagedNode data are shown in Example B-4 on page 198 and Example B-5 on page 198.

Example: B-4 hostResponds ERRM condition on IBM.ManagedNode

```
$ mkcondition -r IBM.ManagedNode \
-e “Status==1” -E “Status!=1” \
-d “Node is responding as part of the cluster” \
-D “Node is no longer responding as part of the cluster” \
-m l hostResponds
$ lscondition hostResponds
Displaying condition information:

condition 1:
        Name             = "hostResponds"
        Node             = "sp6cws"
        MonitorStatus    = "Not monitored"
        ResourceClass    = "IBM.ManagedNode"
        EventExpression  = "Status == 1"
        EventDescription = "Node is responding as part of the cluster"
        RearmExpression  = "Status != 1"
        RearmDescription = "Node is no longer responding as part of the
cluster"
        SelectionString  = ""
        Severity         = "i"
        NodeNames        = {}
        MgtScope         = "l"
$
```

Note: The IBM.ManagedNode conditions have local scope since the resource manager for this class exists on the management server only.

Example: B-5 nodePower ERRM condition on IBM.ManagedNode

```
$ mkcondition -r IBM.ManagedNode \
-e “PowerStatus==1” -E “PowerStatus!=1” \
-d “Node has power” -D “Node power has gone” \
-m l nodePower
$ lscondition nodePower
Displaying condition information:

condition 1:
        Name             = "nodePower"
        Node             = "sp6cws"
        MonitorStatus    = "Not monitored"
        ResourceClass    = "IBM.ManagedNode"
        EventExpression  = "PowerStatus==1"
        EventDescription = "Node has power"
        RearmExpression  = "PowerStatus!=1"
        RearmDescription = "Node power has gone"
        SelectionString  = ""
```

```
        Severity          = "i"
        NodeNames         = {}
        MgtScope          = "l"
$
```

When Status or PowerStatus are set to 1, ERRM triggers a response. Any other value only triggers a response if the value was previously 1. If there is no hardware control point defined for a node, the value for PowerStatus will never be 1, so ERRM never triggers a response.

The ERRM condition for the ServiceRunning sensor (see Example B-1 on page 194) is shown in Example B-6.

Note: The IBM.Sensor condition has management scope because the sensor is populated on the nodes. ERRM on the management server has to subscribe to the updates on all the managed nodes.

Example: B-6 ServiceRunning ERRM condition on IBM.Sensor

```
$ mkcondition -r IBM.Sensor \
-e “Uint32==1” -E “Uint32!=1” \
-d “Node applications are all running” \
-D “Node applications are no longer running” \
-s “Name==\”ServiceRunning\”” \
-m m ServiceRunning
$ lscondition ServiceRunning
Displaying condition information:

condition 1:
        Name             = "ServiceRunning"
        Node             = "sp6cws"
        MonitorStatus    = "Not monitored"
        ResourceClass    = "IBM.Sensor"
        EventExpression  = "Uint32==1"
        EventDescription = "Node applications are all running"
        RearmExpression  = "Uint32!=1"
        RearmDescription = "Node applications are no longer running"
        SelectionString  = "Name==\"ServiceRunning\""
        Severity         = "i"
        NodeNames        = {}
        MgtScope         = "m"
$
```

B.3.2 Defining the response to the conditions

We now need to define a response to ERRM so that we can record the change in status. The definition of the response is shown in Example B-7. We look at what the script does in B.3.3, “Making the status updates available to scripts” on page 200.

Example: B-7 LogStatusData ERRM response definition

```
$ mkresponse -n LogStatusDatatoFIFO \
-s /usr/local/bin/LogStatusData \
-E "ITSO_STATUS_FILE=/tmp/spmondata" LogStatusData
$ lsresponse LogStatusData
Displaying response information:

        ResponseName    = "LogStatusData"
        Node            = "sp6cws"
        Action          = "LogStatusDatatoFIFO"
        DaysOfWeek      = 1-7
        TimeOfDay       = 0000-2400
        ActionScript    = "/usr/local/bin/LogStatusData"
        ReturnCode      = 0
        CheckReturnCode = "n"
        EventType       = "b"
        StandardOut     = "n"
        EnvironmentVars = '"ITSO_STATUS_FILE=/tmp/spmondata"'
        UndefRes        = "n"
$
```

B.3.3 Making the status updates available to scripts

The script that needs to monitor the status updates creates a FIFO file with a name based on its process id (<PID>). The filename is /tmp/spmondata.<PID>. Note that the first part of the name matches the ITSO_STATUS_FILE environment variable we set in Example B-7. Once the file has been created, and opened for read, a file named /tmp/spmondata.<PID>.active is created. This file is an indicator to the response script that should output any status changes to the /tmp/spmondata.<PID> file.

Before writing the update, the script checks to determine whether the file is in use by another process. If it is not, then both files are erased. This cleans up files left behind when the script monitoring the files does not exit cleanly. The response script used to transfer the status updates is shown in Example B-8 on page 201.

Example: B-8 LogStatusData response script

```
#!/bin/ksh
#-------------------------------------------------------------------------------#
#                                                                               #
#   LogStatusData : Sample code to record status updates                        #
#                                                                               #
#   Function : Called as a response to ERRM conditions and will output the      #
#              changed value to all files referenced by the ITSO_STATUS_FILE    #
#              environment variable, if there is a corresponding .active file   #
#              and the file if in use.                                          #
#                                                                               #
#-------------------------------------------------------------------------------#

MainLine ()
{
   if [ -z "${ITSO_STATUS_FILE}" ] ; then
      return                                # Need variable set
   fi

   FILES=$(ls -1 ${ITSO_STATUS_FILE}.*[0-9])

   if [ -z "${FILES}" ] ; then
      return                                # No files found - do nothing
   fi

#
#-------------Build status line to output - of form:--------------------------#
#---<nodename>:<attribute name>:<attribute value>-----------------------------#
   if [ "${ERRM_RSRC_CLASS_PNAME}" = "IBM.Sensor" ] ; then
      STATLINE="${ERRM_NODE_NAME}:${ERRM_RSRC_NAME}:${ERRM_VALUE}"
   else
      STATLINE="${ERRM_RSRC_NAME}:${ERRM_ATTR_PNAME}:${ERRM_VALUE}"
   fi

#
#-------------Process all the files we found----------------------------------#
   for file in $FILES ; do
      ACTIVE="${file}.active"

      if [ ! -e "${ACTIVE}" ] ; then
         continue                           # Not active yet - bypass
      fi

      fuser $file 2>/dev/null|read PIDlist

      if [ -z "${PIDlist}" ] ; then      # File in use?
         rm -f $file $ACTIVE             # .. No - remove it
         continue
```

```
      fi

      echo "${STATLINE}" >> $file        # Output status line to file
   done
}

   MainLine

exit 0
```

B.3.4 Activating the ERRM responses

To activate the responses, they must be associated with the conditions and started as shown in Example B-9.

Example: B-9 Defining and activating the relationship between conditions and responses

```
$ mkcondresp "hostResponds" "LogStatusData"
$ mkcondresp "nodePower" "LogStatusData"
$ mkcondresp "ServiceRunning" "LogStatusData"
$ startcondresp "hostResponds" "LogStatusData"
$ startcondresp "nodePower" "LogStatusData"
$ startcondresp "ServiceRunning" "LogStatusData"
$ lscondresp "*" "LogStatusData"
Displaying condition with response information:

condition-response link 1:
        Condition = "hostResponds"
        Response  = "LogStatusData"
        Node      = "sp6cws"
        State     = "Active"

condition-response link 2:
        Condition = "nodePower"
        Response  = "LogStatusData"
        Node      = "sp6cws"
        State     = "Active"

condition-response link 3:
        Condition = "ServiceRunning"
        Response  = "LogStatusData"
        Node      = "sp6cws"
        State     = "Active"
$
```

B.4 Cluster startup and shutdown processing

The code we developed to emulate the cstartup and cshutdown processing is too large to include in this redbook, but it can be downloaded from the Redbooks Web site. The site details are in E.1, “Locating the sample code” on page 249. The file to be downloaded is AppendixB.samples.tar.

Figure B-1 on page 204 shows the links between the different RSCT components and how the conditions and resources discussed in this appendix are linked to the clstartup and clshutdown scripts. The clshutdown script is actually a link to clstartup, so the same code is executed for both commands.

A number of assumptions were made when writing these scripts:

- If a node is not in a “Managed” state, it is ignored by the script. This includes dependencies defined in the sequence files.
- A node is shut down using the **at** command on the node to schedule an immediate execution of the **shutdown -F** command. If there is any special processing required when shutting down an application, it must be added into /etc/rc.shutdown on the node.
- The clstartup code has the prefix for the FIFO file hardcoded, so the value of the ITSO_STATUS_FILE variable used in the ERRM response definition must be the one shown in Example B-8 on page 201. It is possible to define multiple actions on the responses so that the same code could be executed with different values of the ITSO_STATUS_FILE variable. This would be useful if additional scripts are coded to monitor changes. They must use different file names to keep the data separate.
- The ERRM conditions must be defined and activated because the clstartup code does not activate the conditions. The names used for the conditions are not important because its attribute names will be written to the FIFO file.
- If the PowerStatus attribute value is not set to 1, the node is assumed to be powered off regardless of the value of the Status attribute. As a result, **clshutdown** does not shut down any nodes that do not have a hardware control point. This is probably a good thing, since it is not possible to start these nodes using **clstartup**, which calls **rpower**.

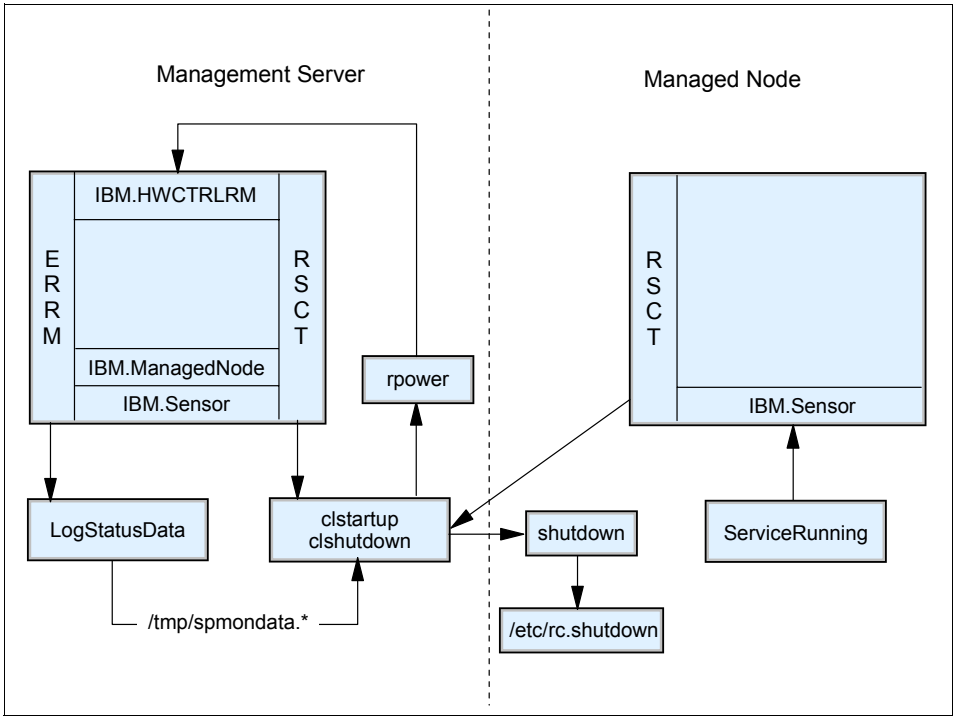


Figure B-1 Cluster startup/shutdown architecture diagram

B.4.1 Controlling the startup and shutdown sequencing

The **clstartup** and **clshutdown** code performs basic sequencing using files with the same format as the sequence files used by **cstartup** and **cshutdown**. Each line in the file can be one of the following:

comment All lines that start with the "#" character. Any text following a "#" in other lines is also ignored.

group A group line starts with the name of a group followed by a colon (":"). The remainder of the line consists of node names separated by spaces or commas. If a group name is used more than once, the group contains the nodes listed on all the lines.

sequence A ">" character is used to define relationships between groups. When executing **clstartup**, an entry of `groupA > groupB` means that all nodes in groupA must be up before any nodes in groupB are started. The same entry when executing **clshutdown** means that all nodes in groupB must be shut down before any nodes in groupA are shut down.
Multiple conditions can be defined using `groupA>groupB>groupC`,

but brackets are not supported, so `(groupA groupB)>groupC` must be split into two separate lines of `groupA>groupC` and `groupB>groupC`.

The sample code looks for the file in the same directory as the code, since a separate file is used for each command:

`clstartup` clstartup.seq
`clshutdown` clshutdown.seq

The sample sequence file in Example B-10 makes sure that the nodes in a drawer are not started until all nodes below that drawer have been started. The shutdown sequence will be from the top down.

Example: B-10 Sample sequence file

```
drawer1: node1
drawer2: node3
drawer3: node5
drawer4: node7
drawer5: node9 node10
drawer6: node11 node12
```

Figure 7-6 drawer7: node13 node14 # comment

```
drawer8: node15 node16
drawer1 > drawer2 > drawer3
drawer3>drawer4    >   drawer5  > drawer6
```

Using the sequence file in Example B-10, a shutdown and restart of all the nodes in our test SP frame produced the output in Example B-11. You should note that node11 to node14 are in a “PreManaged” state. CSM is not installed or active, so they cannot be controlled through CSM.

Example: B-11 clshutdown -r sample output

```
$ clshutdown -r -N CSPNodes
node11 is not managed - will not be processed
node12 is not managed - will not be processed
node13 is not managed - will not be processed
node14 is not managed - will not be processed
DEBUG: ResolveStatus called with node7 2 ServiceRunning 1 - updated status is 3
DEBUG: ResolveStatus called with node1 2 ServiceRunning 1 - updated status is 3
DEBUG: ResolveStatus called with node3 2 ServiceRunning 1 - updated status is 3
DEBUG: ResolveStatus called with node10 2 ServiceRunning 1 - updated status is
3
DEBUG: ResolveStatus called with node5 2 ServiceRunning 1 - updated status is 3
DEBUG: ResolveStatus called with node9 2 ServiceRunning 1 - updated status is 3
Processing sequencing data in /usr/local/bin/clshutdown.seq
WARNING: node11 is not managed by this server sequence entry ignored
```

```
WARNING: node12 is not managed by this server sequence entry ignored
WARNING: node13 is not managed by this server sequence entry ignored
WARNING: node14 is not managed by this server sequence entry ignored
WARNING: node15 is not managed by this server sequence entry ignored
WARNING: node16 is not managed by this server sequence entry ignored
node9 will now be processed
node9: Job root.1055444156.a will be run at Thu Jun 12 14:55:56 EDT 2003.
node10 will now be processed
node10: Job root.1055444159.a will be run at Thu Jun 12 14:55:59 EDT 2003.
DEBUG: ResolveStatus called with node10 3 PowerStatus 0 - updated status is 0
node10 has completed processing
DEBUG: ResolveStatus called with node9 3 PowerStatus 0 - updated status is 0
node9 has completed processing
node7 will now be processed
node7: Job root.1055444249.a will be run at Thu Jun 12 14:57:29 EDT 2003.
DEBUG: ResolveStatus called with node9 0 Status 0 - updated status is 0
DEBUG: ResolveStatus called with node10 0 Status 0 - updated status is 0
DEBUG: ResolveStatus called with node7 3 PowerStatus 0 - updated status is 0
node7 has completed processing
node5 will now be processed
node5: Job root.1055444328.a will be run at Thu Jun 12 14:58:48 EDT 2003.
DEBUG: ResolveStatus called with node7 0 Status 0 - updated status is 0
DEBUG: ResolveStatus called with node5 3 PowerStatus 0 - updated status is 0
node5 has completed processing
node3 will now be processed
node3: Job root.1055444408.a will be run at Thu Jun 12 15:00:08 EDT 2003.
DEBUG: ResolveStatus called with node5 0 Status 0 - updated status is 0
DEBUG: ResolveStatus called with node3 3 PowerStatus 0 - updated status is 0
node3 has completed processing
node1 will now be processed
node1: Job root.1055444493.a will be run at Thu Jun 12 15:01:33 EDT 2003.
DEBUG: ResolveStatus called with node3 0 Status 0 - updated status is 0
DEBUG: ResolveStatus called with node1 3 PowerStatus 0 - updated status is 0
node1 has completed processing
Shutdown complete - now restarting nodes
Processing sequencing data in /usr/local/bin/clstartup.seq
WARNING: node11 is not managed by this server sequence entry ignored
WARNING: node12 is not managed by this server sequence entry ignored
WARNING: node13 is not managed by this server sequence entry ignored
WARNING: node14 is not managed by this server sequence entry ignored
WARNING: node15 is not managed by this server sequence entry ignored
WARNING: node16 is not managed by this server sequence entry ignored
node1 will now be processed
node1 on complete rc=0
DEBUG: ResolveStatus called with node1 0 PowerStatus 1 - updated status is 1
DEBUG: ResolveStatus called with node1 1 Status 0 - updated status is 1
DEBUG: ResolveStatus called with node1 1 Status 1 - updated status is 2
DEBUG: ResolveStatus called with node1 2 Status 0 - updated status is 1
DEBUG: ResolveStatus called with node1 1 Status 1 - updated status is 2
```

```
DEBUG: ResolveStatus called with node1 2 ServiceRunning 1 - updated status is 3
node1 has completed processing
node3 will now be processed
node3 on complete rc=0
DEBUG: ResolveStatus called with node3 0 PowerStatus 1 - updated status is 1
DEBUG: ResolveStatus called with node3 1 Status 1 - updated status is 2
DEBUG: ResolveStatus called with node3 2 ServiceRunning 1 - updated status is 3
node3 has completed processing
node5 will now be processed
node5 on complete rc=0
DEBUG: ResolveStatus called with node5 0 PowerStatus 1 - updated status is 1
DEBUG: ResolveStatus called with node5 1 Status 1 - updated status is 2
DEBUG: ResolveStatus called with node5 2 ServiceRunning 1 - updated status is 3
node5 has completed processing
node7 will now be processed
node7 on complete rc=0
DEBUG: ResolveStatus called with node7 0 PowerStatus 1 - updated status is 1
DEBUG: ResolveStatus called with node7 1 Status 1 - updated status is 2
DEBUG: ResolveStatus called with node7 2 ServiceRunning 1 - updated status is 3
node7 has completed processing
node9 will now be processed
node9 on complete rc=0
node10 will now be processed
node10 on complete rc=0
DEBUG: ResolveStatus called with node10 0 PowerStatus 1 - updated status is 1
DEBUG: ResolveStatus called with node9 0 PowerStatus 1 - updated status is 1
DEBUG: ResolveStatus called with node9 1 Status 1 - updated status is 2
DEBUG: ResolveStatus called with node10 1 Status 1 - updated status is 2
DEBUG: ResolveStatus called with node9 2 ServiceRunning 1 - updated status is 3
node9 has completed processing
DEBUG: ResolveStatus called with node10 2 ServiceRunning 1 - updated status is
3
node10 has completed processing
$
```

The debug messages indicate when the code received a status update from ERRM. The status values used by the code have the following meanings:

0 There is no power to the node, it is in a shutdown state. The PowerStatus attribute of IBM.ManagedNode is 0.

1 The node has power but is not yet participating in the RSCT cluster. The PowerStatus attribute of IBM.ManagedNode is 1 and Status attribute is 0.

2 The node is participating in the RSCT cluster, but the sensor checking the application status indicates that the application is not running.
If there is no sensor defined, the node never has this status.

3 The node is up and running.

C

NIM examples

While performing our testing for this redbook, we used a number of different systems to gain a better understanding of CSM. This appendix contains details of the NIM setup for one of those environments and expands upon the information in *CSM for AIX 5L: Software Planning and Installation Guide,* SA22-7919 to show the specifics of our test installation. This includes examples of how the NIM *script* and *fb_script* resources can be used to customize the nodes being installed.

The specific examples from our installation are:

The sample code shown in this chapter can be downloaded from the Redbook Web site. See E.1, "Locating the sample code" on page 249 for more information. The file to be downloaded is AppendixC.samples.tar.

C.1 NIM integration

When installing servers with PSSP, the installation and configuration of NIM was handled by PSSP. This meant that the NIM configuration was a closed environment. The administrators needed little knowledge of NIM, and were unable to configure NIM by themselves if they did. The PSSP configuration would overwrite anything the administrators set up themselves.

With CSM, the administrator has full control of the NIM configuration. Some utilities are provided to assist the administrator when the management server is configured as a NIM server:

csm2nimnodes Creates a NIM client definition for the specified nodes or node group members, using information that has been entered into CSM.

csm2nimgrps Creates NIM groups to match existing node groups. This will not maintain the NIM group since the members of the CSM group change, so its value is limited.

csmsetupnim Prepares a node for installation and configuration. This should be run before configuring the node for installation through NIM.

C.2 NIM resources

The resources that are required to perform an installation through NIM are shown in Table C-1. The table indicates which are required with an R and which are optional with an O. For many resource types there can be one (1) or more (+) instances allocated.

Table C-1 NIM resource chart

Resource Type	R/O	1/+	Description
machine	R	1	The machine definition for the node to be installed can be created using **csm2nimnodes**.
lpp_source	R	1	See “lpp_source” on page 218.
spot	R	1	See “spot” on page 221.
bosinst_data	R	1	See “bosinst_data” on page 215.
mksysb	R	1	Required for an installation with source=mksysb.

Resource Type	R/O	1/+	Description
script	O	+	See “script” on page 221.
fb_script	O	+	See “fb_script” on page 221.
installp_bundle	O	+	See “installp_bundle” on page 217.

C.2.1 NIM resource groups

A NIM resource group allows a number of NIM resources to be grouped together under a single name, which is specified on the command that allocates the resources.

A number of different groups can be defined based upon the types of servers to be installed or even the software levels to be installed.

For our testing, we included a bosinst_data resource in our resource group because we were prepared to have AIX installed on the default disk (hdisk0). This is not the case in a production environment, so we would not recommend including this resource in a resource group unless you are creating a separate resource group for each machine.

The README bosinst file (/usr/lpp/bosinst/bosinst.template.README), supplied with NIM, describes how the target disks, for an installation, should be defined. See Example 3-12 on page 48 for some sample disk definitions.

C.2.1.1 Sample resource group

The resource group we use that illustrates the use of the different NIM resources is sp6_ldap. When this resource group is used, the node will be installed with openssl, openssh, and LDAP.

Example: C-1 sp6_ldap resource group contents

```
$ lsnim -g sp6_ldap
sp6_ldap:
   class   = groups
   type    = res_group
   member1 = lppsource_aix520; lpp_source; ready for use;
   member2 = spot_aix520; spot; ready for use;
   member3 = noprompt; bosinst_data; ready for use;
   member4 = openssl; installp_bundle; ready for use;
   member5 = opensshclient; installp_bundle; ready for use;
   member6 = LDAPclient; installp_bundle; ready for use;
   member7 = opensshcopy; script; ready for use;
   member8 = opensshinstcfg; fb_script; ready for use;
   member9 = ldapconfig; script; ready for use;
```

```
   member10 = rsctsensor; installp_bundle; ready for use;
$
```

The lppsource_aix520 and spot_aix520 resources are created following the instructions in *CSM for AIX 5L: Software Planning and Installation Guide,* SA22-7919. The only difference between these resources and the ones allocated in PSSP is the location of the files.

When allocating a resource group to a machine object, the individual resources are added in the order that they are held in the ODM. This should match the order they appear in when you list the resource group, but may not if a lot of changes have been made to the group contents.

Note: We have reported this behavior to CSM and NIM development. IBM intends to develop a procedure to allocate the resources in member order to fix the current behavior.

When a node has been prepared for installation, the node-specific script generated by NIM can be checked to see the actual processing order. See Example: C-10, “node1.script NIM file” on page 219 for an example.

It is important to get the order correct if there are dependencies between the resources. In Example C-1 the openssl resource is listed before the openssh resource. This order was chosen because openssl is a prerequisite for openssh and must be processed first.

We also produced a script that can be used to compare the member sequence of a resource group with the sequence in which the resources are reported by the ODM. If there is a difference, the two sequences are shown side by side so you can determine whether this may cause a problem for your installation. This script is shown in Example C-2.

Tip: If you do experience the problem, you can just create a new resource group listing the resources in the order you want them to be allocated. The problem occurs if you perform deletions and additions to the group.

Example: C-2 check_res_group script

```
#!/bin/ksh
#-------------------------------------------------------------------------------#
#                                                                               #
#   check_res_group : Check a resource group                                    #
#                                                                               #
#   Function : Check a resource group to see whether the resources may be       #
#              allocated in a sequence that doesn't match their position        #
```

```
#              in the resource group.                                          #
#                                                                              #
#   Called as : check_res_group <group name>                                   #
#                                                                              #
#------------------------------------------------------------------------------#
me=$(basename $0)

#
#-------------Make sure a group name was passed-------------------------------#
   if [ -z "$1" ] ; then
      echo "No group name entered"
      exit 1
   fi

#
#-------------Extract the NIM id for the group name---------------------------#
   odmget -q "class=407 and name=$1" nim_object |
      grep "id =" | read id equals nimid rest

   if [ -z "${nimid}" ] ; then
      echo "Unable to locate resource group $1"
      exit 1
   fi

#
#-------------Now process the individual members of the group-----------------#
   format="%-30s %s\n"       # Format used to display comparision
   lastno=0
   seqerr=0
   odmget -q "id=${nimid}" nim_attr |
      grep "seqno =" |
      while read seqno equals number rest ; do
         if [ $lastno -lt "${number}" ] ; then
            lastno="${number}"             # Record last number processed
            continue
         fi

         seqerr=1                          # Sequence error detected
         break
      done

   if [ $seqerr -eq 0 ] ; then
      echo "ODM sequence matches membership sequence - no problems"
      exit 0
   fi

#
#-------------Provide details of the differences------------------------------#
   list=""
```

```
lsnim -g $1 |
   grep "member"|grep ";" | sed "s/;/ /"|
   while read member equals resource rest ; do
      list="${list} ${resource}"
   done

set -- $list             # Resources by member number
echo "ODM sequence doesn't match group membership, comparison below:"
printf "${format}" "By member" "ODM sequence"
printf "${format}" "---------" "------------"
odmget -q "id=${nimid}" nim_attr |
   grep "value =" |
   while read value equals name rest ; do
      resource=$(echo "${name}" | sed "s/\"//g")
      printf "${format}" $1 ${resource}
      shift
   done
```

C.2.2 Resources allocated for an install

When preparing for the installation of a node, assuming all the required information has been entered into CSM and the machine resources have been defined, a couple of steps are required:

Prepare CSM The **csmsetupnim** command prepares CSM for the installation of the node by creating customization files and updating the node in the CSM database. It will then allocate a *script* resource in NIM that will perform the CSM customization during the install.

Prepare NIM The node should be prepared for a bos_inst operation from the lppsource or an mksysb.

Following the preparation for one of our test servers (Figure 1-1 on page 4), the NIM resources allocated appear as shown in Example C-3.

Example: C-3 Allocation of NIM resources to a node

```
$ csmsetupnim -n node1
$ nim -o bos_inst -a source=rte -a group=sp6_ldap -a boot_client=no node1
$ lsnim -l node1
node1:
   class           = machines
   type            = standalone
   platform        = chrp
   netboot_kernel  = mp
   if1             = ms_net node1 0004AC49C746 ent
   cable_type1     = N/A
```

```
   Cstate          = BOS installation has been enabled
   prev_state      = ready for a NIM operation
   Mstate          = currently running
   boot            = boot
   bosinst_data    = noprompt
   installp_bundle = LDAPclient
   installp_bundle = opensshclient
   installp_bundle = openssl
   installp_bundle = rsctsensor
   lpp_source      = lppsource_aix520
   nim_script      = nim_script
   script          = csmprereboot_script
   script          = ldapconfig
   script          = opensshcopy
   spot            = spot_aix520
   cpuid           = 000132374C00
   control         = master
   fb_script       = opensshinstcfg
$
```

We now take a look at the individual resources listed in Example C-3 to understand what they are and how they affect the NIM installation of a node.

C.2.2.1 bosinst_data

The bosinst file provides answers to the many prompts that are issued during a regular installation. PSSP creates a separate bosinst_data resource for each node because each node can have its own disk configuration.

The bosinst file is stanza-based and has definitions for the processing to be performed, the locale, and the disks to be used for the installation. The control data for our SP nodes is shown in Example C-4.

Example: C-4 noprompt bosinst_data resource

```
# IBM_PROLOG_BEGIN_TAG
# This is an automatically generated prolog.
#
# bos52B src/bos/usr/lpp/bosinst/bosinst.template/bosinst.template 1.11.1.16
#
# Licensed Materials - Property of IBM
#
# (C) COPYRIGHT International Business Machines Corp. 1993,1994
# All Rights Reserved
#
# US Government Users Restricted Rights - Use, duplication or
# disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
```

```
#
# IBM_PROLOG_END_TAG
#
..
#
#     PLEASE READ /usr/lpp/bosinst/bosinst.template.README for more information
#
################################################################################
##

control_flow:
    CONSOLE = /dev/tty0
    INSTALL_METHOD = overwrite
    PROMPT = no
    EXISTING_SYSTEM_OVERWRITE = yes
    INSTALL_X_IF_ADAPTER = yes
    RUN_STARTUP = no
    RM_INST_ROOTS = no
    ERROR_EXIT =
    CUSTOMIZATION_FILE =
    TCB = no
    INSTALL_TYPE =
    BUNDLES =
    RECOVER_DEVICES = no
    BOSINST_DEBUG = no
    ACCEPT_LICENSES = yes
    DESKTOP = CDE
    INSTALL_DEVICES_AND_UPDATES = yes
    INSTALL_CONFIGURATION = Default
    IMPORT_USER_VGS =
    ENABLE_64BIT_KERNEL = no
    CREATE_JFS2_FS = no
    ALL_DEVICES_KERNELS = yes
    GRAPHICS_BUNDLE = yes
    DOC_SERVICES_BUNDLE = no
    NETSCAPE_BUNDLE = no
    HTTP_SERVER_BUNDLE = no
    KERBEROS_5_BUNDLE = no
    SERVER_BUNDLE = no
    ALT_DISK_INSTALL_BUNDLE = no
    REMOVE_JAVA_118 = no

target_disk_data:
    LOCATION =
    SIZE_MB =
    HDISKNAME =

locale:
    BOSINST_LANG = en_US
```

```
    CULTURAL_CONVENTION = en_US
    MESSAGES = en_US
    KEYBOARD = en_US
```

It has some differences from the customization described in the installation scenario in *CSM for AIX 5L: Software Planning and Installation Guide,* SA22-7919, which are shown in Table C-2.

Table C-2 Variations from installation scenario

Label	Value	Reason
RUN_STARTUP	No	Prevents the node hanging in installation assistant after the initial install.
ENABLE_64BIT_KERNEL	No	The SP nodes being used do not support the 64-bit kernel.
CREATE_JFS2_FS	No	Requires a 64-bit kernel.

C.2.2.2 installp_bundle

An installp bundle defines a number of AIX filesets or RPM files that will be installed. They must all be located in the lpp_source resource allocated for the NIM bos_inst or cust operation.

Example C-5, Example C-6 on page 218, Example C-7 on page 218 and Example C-8 on page 218 show the contents of the bundles used in our test environment.

Example: C-5 LDAPclient installp_bundle resource

```
$ lsnim -l LDAPclient
LDAPclient:
   class       = resources
   type        = installp_bundle
   Rstate      = ready for use
   prev_state  = unavailable for use
   location    = /csminstall/nim/LDAPclient.bnd
   alloc_count = 1
   server      = master
$ cat /csminstall/nim/LDAPclient.bnd
#
# Bundle file for LDAP client filesets
#
I:ldap.client
I:ldap.html.en_US.config
I:ldap.html.en_US.man
```

```
I:ldap.msg.en_US
$
```

Example: C-6 opensshclient installp_bundle resource

```
#
# Bundle file for openssh filesets from AIX BonusPak CD
#
I:openssh.base.client
I:openssh.license
I:openssh.man.en_US
I:openssh.msg.en_US
I:openssh.msg.EN_US
```

Example: C-7 openssl installp_bundle resource

```
#
# Bundle file for openssl RPM from AIX Toolbox for Linux Applications on AIX CD
#
R:openssl*
```

Example: C-8 rsctsensor installp_bundle resource

```
#
# Bundle file for RSCT Sensor Resource Manager
#
I:rsct.core.sensorrm
```

C.2.2.3 lpp_source

The lpp_source resource contains all the AIX filesets and RPMs required for an installation with source=rte, and any additional files required for installp_bundle resources that are to be processed at the same time.

When using PSSP, this resource is defined automatically as part of the configuration of the NIM master, while with CSM it needs to be defined by the CSM administrator so that the location is at the discretion of the administrator.

C.2.2.4 nim_script

The nim_script resource is allocated by NIM when a node is prepared for an operation. It defines a directory where a node-specific script has been created. This script will be executed after the node has been installed but before it has been rebooted.

Example: C-9 nim_script resource

```
$ lsnim -l nim_script
nim_script:
   class       = resources
   type        = nim_script
   comments    = directory containing customization scripts created by NIM
   Rstate      = ready for use
   location    = /export/nim/scripts
   alloc_count = 2
   server      = master
   reserved    = yes
$
```

If you look at Example C-3 on page 214, you see that there are several installp_bundle resources allocated to the node but there is no indication of the order in which they will be processed. As the order of processing is important for openssl and openssh, we need to be able to verify the order.

The file created for a node is of the form <nodename>.script, where <nodename> is the name of the machine resource for the node. The directory where it is stored can be identified by checking the location of the nim_script resource (see Example C-9 for an example of this).

So if we check the script created for node1 (see Example C-10) we can see that the order of processing for the installp_bundle resources is openssl, openssshclient, LDAPclient, and then rsctsensor.

Similarly, we can see that the order of processing for the script resources is csmprereboot, ldapconfig, and then openssshcopy.

Example: C-10 node1.script NIM file

```
#!/bin/ksh

# nim_script for node1
result=success

export NIM_NON_41_MASTER=yes
export NIM_SCRIPT=yes
# NIM client initialization
/../SPOT/usr/lpp/bos.sysmgt/nim/methods/c_mk_nimclient ${VERBOSE} \
        -ahostname=node1 \
        -aip=192.168.1.1 \
        -acable_type=N/A \
        -asnm=255.255.255.0
[[ $? != 0 ]] && result=failure

#
# Add fb_script's contents to /etc/firstboot from master
```

```
#
/../SPOT/usr/lpp/bos.sysmgt/nim/methods/c_add_fb_script ${VERBOSE} -a
location=sp6cws-en0:/csminstall/nim/opensshinstcfg

# lpp_source=lppsource_aix520
/../SPOT/usr/lpp/bos.sysmgt/nim/methods/c_installp ${VERBOSE} \
        -aboot_env=yes \
        -ainstallp_bundle=sp6cws-en0:/csminstall/nim/openssl.bnd \
        -alpp_source=sp6cws-en0:/csminstall/AIX/aix520/lppsource
[[ $? != 0 ]] && result=failure

# lpp_source=lppsource_aix520
/../SPOT/usr/lpp/bos.sysmgt/nim/methods/c_installp ${VERBOSE} \
        -aboot_env=yes \
        -ainstallp_bundle=sp6cws-en0:/csminstall/nim/opensshclient.bnd \
        -alpp_source=sp6cws-en0:/csminstall/AIX/aix520/lppsource
[[ $? != 0 ]] && result=failure

# lpp_source=lppsource_aix520
/../SPOT/usr/lpp/bos.sysmgt/nim/methods/c_installp ${VERBOSE} \
        -aboot_env=yes \
        -ainstallp_bundle=sp6cws-en0:/csminstall/nim/LDAPclient.bnd \
-alpp_source=sp6cws-en0:/csminstall/AIX/aix520/lppsource
[[ $? != 0 ]] && result=failure

# lpp_source=lppsource_aix520
/../SPOT/usr/lpp/bos.sysmgt/nim/methods/c_installp ${VERBOSE} \
        -aboot_env=yes \
        -ainstallp_bundle=sp6cws-en0:/csminstall/nim/rsctsensor.bnd \
-alpp_source=sp6cws-en0:/csminstall/AIX/aix520/lppsource
[[ $? != 0 ]] && result=failure

# script=csmprereboot_script
/../SPOT/usr/lpp/bos.sysmgt/nim/methods/c_script ${VERBOSE} \
        -alocation="sp6cws-en0:/csminstall/csm/csmprereboot"

script_rc=$?
[[ ${script_rc} != 0 ]] && result=failure

# script=ldapconfig
/../SPOT/usr/lpp/bos.sysmgt/nim/methods/c_script ${VERBOSE} \
        -alocation="sp6cws-en0:/csminstall/nim/ldapconfig"

script_rc=$?
[[ ${script_rc} != 0 ]] && result=failure

# script=opensshcopy
/../SPOT/usr/lpp/bos.sysmgt/nim/methods/c_script ${VERBOSE} \
        -alocation="sp6cws-en0:/csminstall/nim/opensshcopy"
```

```
script_rc=$?
[[ ${script_rc} != 0 ]] && result=failure

[[ ${result} = success ]] && exit 0 || exit 1
```

C.2.2.5 script

The script resource is used to perform customization of the node being installed. It will be called after the installation but before the node is rebooted during a bos_inst operation. If a cust operation is performed, it will be called directly, assuming the CSM server has remote shell access due to the /.rhosts file on the node.

When a script is called during a bos_inst operation, it can perform very limited operations. The limitations are the same as with script.cust since that script was called from a PSSP script (pssp_script) that was defined as a NIM resource and then executed by NIM.

Note: Any resources allocated through NIM are available during the first stage of the installation. After the reboot the resources are no longer assigned or attached to the managed node. If any of the resource data will be needed later, the script resource must preserve it for later use, as shown in Example C-13 on page 228.

CSM also uses a script resource to perform customization of the node but, unlike PSSP, you can still define your own script resources to be processed.

Both PSSP and CSM use a bootstrapping technique so that additional customization can be performed during the reboot after the node install. We used this technique for configuring LDAP on our test servers; see sample code in Example C-12 on page 224.

C.2.2.6 spot

The *spot* resource is created from an lpp_source resource and contains a bootable image for a node. The only difference from the spot resource that was defined with PSSP is that the CSM administrator will define and create the resource so that the location is flexible.

C.2.2.7 fb_script

The fb_script is a new feature with AIX 5.2. This resource is a korn shell script that is executed during the reboot following the install.

The script contents are appended to the /etc/firstboot script during the installation of the node. During the boot process of a node, the fbcheck inittab entry checks for the /etc/firstboot file and if it exists, it is renamed and executed. This check is performed before any network connectivity has been established.

Since the /etc/firstboot file has been renamed, it will not be executed on subsequent reboots.

There are certain limitations when using this exit that occur because the scripts are appended to /etc/firstboot, rather than being called from /etc/firstboot:

- The script has to be written so that it can be executed as a korn shell. If perl is to be used, the script will need to pass the perl code to the perl module through the STDIN of the call to perl.
- An exit statement in your script will terminate all processing in /etc/firstboot so the installation won't complete successfully. We recommend defining a function in your script to perform all the processing and then calling that function. A return statement can then be used in place of an exit so that the entire /etc/firstboot is processed. An example of this can be seen in Example C-14 on page 229.

C.3 Installing additional filesets

Whether a node is installed using an lpp_source or an mksysb resource, there are likely to be additional filesets that are needed which already exist in the lpp_source resource.

Creating an installp_bundle resource for these filesets will ensure that they are installed on the node. If they were already included in an mksysb resource, it can ensure that they are updated to a more recent level.

Example C-8 on page 218 shows an installp_bundle that will make sure that the rsct.core.sensorrm fileset is installed. It is not automatically installed with the other RSCT filesets, but is required if sensors are to be used. See *RSCT for AIX 5L: Guide and Reference,* SA22-7889 and *RSCT for AIX 5L: Technical Reference,* SA22-7890 for more information about using sensors.

If an mksysb installation is performed, the software levels on the installed node will be those contained in the mksysb, not the levels in the lpp_source. CSM has some dependencies on the levels of the RSCT software and these are not updated by the CSM node installation process. To ensure that the latest software levels are installed, if an mksysb installation is performed, the installp_bundle resource shown in Example C-11 on page 223 can be used. It can also be used to perform future software updates using the NIM cust operation.

CSM 1.3.2 changes the way that the updatenode process operates because it no longer updates the software on the managed nodes. So this operation needs to be done in other ways. One way to handle it is to add the CSM filesets to an installp_bundle resource that is allocated during node installations.

The installp_bundle shown in Example C-11 contains the RSCT fileset definitions, and the CSM filesets have been added for CSM 1.3.2 to ensure that all nodes have the new software level following a bos_inst or cust operation.

Note: Specific software levels can be specified in an installp_bundle resource, but these will only be enforced if the installation is from an mksysb resource that contains the software at a level that is earlier than, or the same as, the level defined in the installp_bundle resource.

Example: C-11 csmprereq installp_bundle resource

```
#
# Bundle file for CSM prerequisite software
#
I:rsct.core
I:rsct.compat.basic.rte
I:rsct.compat.clients.rte
I:rsct.basic.rte
I:rsct.msg.en_US
#
# Add CSM filesets from CSM 1.3.2 onwards
#
I:csm.core
I:csm.client
I:csm.diagnostics
I:csm.dsh
I:csm.msg.EN_US
```

C.4 Using NIM to install and configure LDAP clients

PSSP provides some basic user management using the AIX automounter for home directories, and file collections to distribute user passwords. There are many alternative user management functions, and we performed some basic testing with NIM to make sure that nodes are configured to use an LDAP server. We now illustrate some of the techniques that can be used with NIM.

The installation and configuration of LDAP requires the following resources:

installp_bundle This defines the filesets to be installed for LDAP. It can also be used as input to the **gencopy** command when transferring

the LDAP filesets to the lpp_source resource. Example C-5 on page 217 shows the contents of this resource.

lpp_source The LDAP filesets must be placed in the lpp_source resource that will be allocated to the managed nodes during the installation.

script This is required to perform the configuration of LDAP so that the client can work with the LDAP server.

C.4.1 The installation logic

During the installation of the managed node, the LDAP filesets are installed by NIM, and then the node needs to be configured. It is not possible to perform the configuration at the time the script resource is processed, or when the fb_script resource will be processed, because the node must be stable and have network connectivity to the LDAP server.

To resolve this, we use a bootstrapping technique. The script resource takes a copy of itself to the node being installed, adds an entry to the inittab so that script is called during the reboot and after the network has been established. After the configuration has completed, the script removes the entry from the inittab so that the script does not run again. The script is shown in Example C-12.

Example: C-12 LDAP configuration script

```
#!/bin/ksh
#-------------------------------------------------------------------------------#
#                                                                               #
#   ldapconfig : Configure the LDAP client - assumes filesets installed         #
#                                                                               #
#   Function : Check current environment to determine if code can be            #
#              executed now or whether it needs to be copied and added to       #
#              the inittab for execution at the next reboot.                    #
#                                                                               #
#-------------------------------------------------------------------------------#

#
#-------------------------------------------------------------------------------#
#   ConfigLDAP : Perform the configuration of LDAP                              #
#-------------------------------------------------------------------------------#

ConfigLDAP()
{
   ESU="/etc/security/user"
   echo "Updating ${ESU} for root settings"
```

```
#
#-------------Update SYSTEM and registry-------------------------------------#
   chsec -f ${ESU} -s root -a SYSTEM=compat -a registry=files

   if [ $? -ne 0 ] ; then
      return
   fi

#
#-------------Perform LDAP configuration-------------------------------------#
   LDAPhost="sp6cws-en0"
   LDAPadm="cn=admin,o=ibmitso,c=us"
   LDAPpwd="itsoadmin"
   LDAPdom="cn=aixdata,ou=itso,o=ibmitso,c=us"
   chmod u-x /usr/sbin/secldapclntd  # Prevent execution during setup
   mksecldap -c -h ${LDAPhost} -a ${LDAPadm} -p ${LDAPpwd} -d ${LDAPdom}
   rc=$?
   chmod u+x /usr/sbin/secldapclntd  # Reenable execution

   if [ $rc -ne 0 ] ; then
      return
   fi

#
#-------------Enable the LDAP processing-------------------------------------#
   chsec -f ${ESU} -s default -a SYSTEM=LDAP -a registry=LDAP

   if [ $? -ne 0 ] ; then
      return
   fi

#
#-------------Complete now - remove inittab entry (if it exists)-------------#
   nohup /usr/sbin/secldapclntd >/dev/console 2>&1 &     # Start LDAP code
   rmitab LDAPconfig 2>/dev/null
}

#
#------------------------------------------------------------------------------#
#   TransferScript : Transfer this script and create inittab entry             #
#------------------------------------------------------------------------------#

TransferScript()
{
   DEST="/opt/itso/install"
   SCRIPT="${DEST}/LDAPconfig"
   test -d ${DEST} -o -n "$(mkdir -p ${DEST})"
   echo "${PROGNAME} will be copied to ${SCRIPT}"
```

```
#
#-------------Copy the file to the target area - always----------------------#
   cp -p ${0} ${SCRIPT}

#
#-------------All OK so far - create inittab entry---------------------------#
   if [ $? -eq 0 ] ; then
      mkitab -i cron "LDAPconfig:2:once:${SCRIPT} >/dev/console 2>&1"
   fi

}

#
#-------------Check the script location--------------------------------------#
   PROG=${0}
   PROGNAME=${PROG##*/}
   DIRNAME=${PROG#%/*}

   if [ "${DIRNAME}" = "/tmp/_nim_dir_${PPID}" ] ; then
      TransferScript
   else
      ConfigLDAP
   fi

exit 0
```

The ldapconfig script has been coded so that it will work during a bos_inst and a cust operation. For the bos_inst, it performs the bootstrap operation and for the cust, it just performs the processing directly.

Tip: When NIM is processing a script resource, it copies the script into a temporary directory. During a bos_inst operation, the directory name references the process ID of the NIM process that calls the script. The ldapconfig script checks the location it is being executed from and if it references the parent process ID, the bootstrap process is invoked.

For completeness, there are some additional points to note about the script:

- The ldapconfig script creates a directory to copy the code to on the managed node. It cannot be left in /tmp because that directory is cleared out before the reboot.
- The LDAP server details have been hardcoded so that the script could be updated to read the information from a file on the system to be more secure. The file could be distributed by CFM as part of the CSM installation since this

is the last operation performed. After the reboot, the script needs to wait until the file has been distributed in order to get the details.

- When the **mksecldap** command is executed, it starts the secldapclntd daemon, which causes problems during a cust operation because control is never passed back to NIM. To work around this, the ldapconfig script removes the execute permission from the secldapclntd script so that the start of the code fails. Once **mksecldap** has completed, the execute permission is restored and secldapclntd is started.

C.5 A working example of openssh installation

The installation of the openssh code that is documented in *CSM for AIX 5L: Software Planning and Installation Guide,* SA22-7919 failed when we tried it on our test systems. This was due to problems installing the openssh.base.server fileset using NIM prior to the server reboot.

Note: This condition has been reported to the CSM and AIX developers, and a problem has been opened against the openssh package. A fix for this may be available at the time you read this, but the process we used to work around the problem may still be of interest.

The installation and configuration of openssh requires the following resources:

installp_bundle These will define the filesets to be installed for openssh and openssl. They can also be used as input to the **gencopy** command when transferring the openssh filesets and openssl RPM file to the lpp_source resource. Example C-6 on page 218 and Example C-7 on page 218 show the contents of these resources.

lpp_source The openssh filesets and openssl RPM file must be placed in the lpp_source resource that will be allocated to the managed nodes during the installation.

script This is required to transfer the openssh.base.server fileset to a directory on the managed node.

fb_script This is used to install the openssh.base.server fileset.

C.5.1 The installation logic

Since the openssh.base.server filesets fail to install as part of a NIM bos_inst operation, an alternative approach is required that will have the server running by

the time the final stages of the CSM installation are performed, if ssh is configured as the remote shell.

The openssl RPM and openssh client code will install successfully, so they are handled as installp_bundle resources; see Example C-7 on page 218 and Example C-6 on page 218. The openssl installp_bundle resource has to be processed first because openssh relies on it being available.

Since the openssh.base.server fileset is only available before the NIM reboot, the script resource shown in Example C-13 copies the fileset to the /usr/sys/inst.images directory so that it is available following the reboot. The limitations on the script resource environment mean that the **gencopy** and **bffcreate** commands fail to complete, which is why **cp** is used to copy the fileset.

Example: C-13 opensshcopy script resource

```
#!/bin/ksh
#-------------------------------------------------------------------------------#
#                                                                               #
#   opensshcopy : openssh installation script for NIM (part 1)                  #
#                                                                               #
#   Function : Manage the installation of openssh.base.server during a NIM     #
#              installation of a server.                                        #
#                                                                               #
#              When openssh.base.server is installed the ssh-keygen code is     #
#              executed to generate a host key, this fails during the NIM       #
#              installation for openssh.base.server 3.4.0.5203.                 #
#                                                                               #
#              2 functions need to be performed:                                #
#                                                                               #
#              1. Copy the installp fileset for openssh.base.server to the      #
#                 machine being installed (using cp from the lppsource)         #
#                                                                               #
#              2. When called during a reboot it will install the fileset       #
#                 copied during phase 1 and create the /etc/pam.conf file       #
#                 if it doesn't exist                                           #
#                                                                               #
#-------------------------------------------------------------------------------#
#                                                                               #
#   This code performs function 1.                                              #
#                                                                               #
#-------------------------------------------------------------------------------#

#
#-------------Assume fileset will be placed in /usr/sys/inst.images-----------#
#-------------and is sourced from /SPOT/usr/sys/inst.images-------------------#
cp -p /../SPOT/usr/sys/inst.images/installp/ppc/openssh.base.* \
      /usr/sys/inst.images
```

```
exit 0
```

The fb_script resource shown in Example C-14 creates a table of contents for the /usr/sys/inst.images directory and then installs the openssh.base.server fileset. As openssh is packaged to use the System V standard for starting daemons, the sshd subsystem is started when the appropriate entry in the inittab is processed.

Example: C-14 opensshinstcfg fb_script resource

```
#!/bin/ksh
#-------------------------------------------------------------------------------#
#                                                                               #
#   opensshinstcfg : openssh installation script for NIM (part 2)               #
#                                                                               #
#   Function : Manage the installation of openssh.base.server during a NIM      #
#              installation of a server.                                        #
#                                                                               #
#              When openssh.base.server is installed the ssh-keygen code is     #
#              executed to generate a host key, this fails during the NIM       #
#              installation for openssh.base.server 3.4.0.5203.                 #
#                                                                               #
#              2 functions need to be performed:                                #
#                                                                               #
#              1. Copy the installp fileset for openssh.base.server to the      #
#                 machine being installed (using cp from the lppsource)         #
#                                                                               #
#              2. When called during a reboot it will install the fileset       #
#                 copied during phase 1 and create the /etc/pam.conf file       #
#                 if it doesn't exist                                           #
#                                                                               #
#-------------------------------------------------------------------------------#
#                                                                               #
#   This code performs function 2.                                              #
#                                                                               #
#-------------------------------------------------------------------------------#


#
#-------------------------------------------------------------------------------#
#   InstallSSH : Install the openssh.base.server fileset                        #
#-------------------------------------------------------------------------------#

InstallSSH()
{

#
#-------------Assume fileset is in /usr/sys/inst.images-----------------------#
   PAMCONF="/etc/pam.conf"
```

```
   inutoc /usr/sys/inst.images

   if [ $? -ne 0 ] ; then
      return
   fi

   installp -avXYd /usr/sys/inst.images openssh.base.server

   if [ $? -ne 0 -o -e ${PAMCONF} ] ; then
      return
   fi

#
#-------------Create a "default" pam.conf-----------------------------------#
   cat << InstallSSH_EOF > ${PAMCONF}
sshd    auth            required        /usr/lib/security/pam_aix
OTHER   auth            required        /usr/lib/security/pam_aix
sshd    account         required        /usr/lib/security/pam_aix
OTHER   account         required        /usr/lib/security/pam_aix
sshd    password        required        /usr/lib/security/pam_aix
OTHER   password        required        /usr/lib/security/pam_aix
sshd    session         required        /usr/lib/security/pam_aix
OTHER   session         required        /usr/lib/security/pam_aix
InstallSSH_EOF

   return
}

InstallSSH
```

The following scripts perform very basic functions, and there is a lot more they could do, just for the openssh setup. Some improvements that could be made are:

- The opensshcopy script could be updated to check the NIM state, in the way the ldapconfig script does, so that it can actually install the openssh.base.server fileset if it is being called during a cust operation.
- If a node is being installed from an mksysb resource, it may already have openssh.base.server installed with a set of public and private keys stored in the image. The opensshinstcfg script could be updated to detect this and generate a new set of keys. This would ensure that the same keys are not being used on multiple servers.

D

Installation process details

This appendix contains details of the processing performed by CSM when installing and configuring nodes from the management server. The processing is compared with that performed by PSSP so that the differences between the products can be better understood.

Two installation types are considered:

- Installation of CSM on a machine that already has AIX installed. This is a function that cannot be performed using PSSP. This process is described in D.1, “A closer look at CSM’s updatenode script” on page 232.
- Installation of a node with AIX and CSM at the same time is covered in D.2, “CSM installation details using a NIM install” on page 235. This scenario matches the PSSP concept where nodes are installed and configured by PSSP at the same time. The two processes are compared in D.3, “Comparison with PSSP” on page 242.

D.1 A closer look at CSM's updatenode script

If a server already has AIX installed, it can be defined on a CSM management server using the **definenode** command. It can then be configured as an active member of the management domain using the **updatenode** command. In this section we examine the detailed processing performed by **updatenode**. This should assist in problem determination.

D.1.1 Remote shell setup

If the CSM configuration has SetupRemoteShell set to 1 (yes), then steps are taken to make sure that the management server has remote shell access. These steps vary according to the value of the RemoteShell attribute in **csmconfig**:

/usr/bin/ssh A public/private key pair will be generated for the root user on the management server, if they do not already exist under /.ssh. These key pairs are stored without a passphrase, so access to them later does not need keyboard intervention by the root user. The **ssh** command is then issued to the node to add the management server's root public keys to /.ssh/authorized_keys and /.ssh/authorized_keys2 on the node.

/usr/bin/rsh The **rsh** command is issued to the node to add the management server hostname to the /.rhosts on the node.

When the command is issued to the node to prepare the remote shell Authorization, it is executed from an *expect* script. The expect code will detect a password prompt and then prompt the administrator for the password when needed. When more than one node is specified for the **updatenode** command, the expect script will reuse the root password supplied to it on all remaining nodes, so the password does not need to be reentered for every node.

Important: If the SetupRemoteShell attribute under **csmconfig** is set to 0 (no), the administrator must set up the remote shell access for the command defined in the RemoteShell attribute of **csmconfig**, before executing **updatenode**.

D.1.2 Prepare RSCT

Set the node's attribute **AllowManageRequest=1** in the **IBM.ManagedNode** class, on the management server so that the node can join the management domain.

D.1.3 Prepare managed node configuration data

The node being managed does not have access to the configuration data held on the management server, so it has to be stored in files that are made available to the node while it is being integrated into the cluster.

D.1.3.1 /csminstall/csm/config/nodemap

This file contains a mapping of CSM (pre)managed nodes' nodenames as defined in the CSM database (listed in the first column), and node hostnames as returned by the hostname command on the nodes (comprising the second column), as shown in Example D-1.

Example: D-1 nodemap file

```
[c5aix03][/csminstall/csm/config]> more nodemap
c66rp13.ppd.pok.ibm.com c66rp13.ppd.pok.ibm.com
c66rp12.ppd.pok.ibm.com c66rp12.ppd.pok.ibm.com
```

D.1.3.2 /csminstall/csm/config/<node>.config_info

This file is node-specific, with <node> referring to the node name. A sample file is shown in Example D-2.

Example: D-2 config_info file

```
$ cd /csminstall/csm/config
$ cat node1.config_info
RemoteShell=/usr/bin/ssh
SetupRemoteShell=1
Hostname=node1
ManagementServer=sp6cws-en0
InstallOSName=AIX
InstallCSMVersion=1.3.1
InstallDistributionVersion=5.2.0
InstallDistributionName=
InstallPkgArchitecture=
Mode=Managed
ConsoleSerialDevice=
InstallServer=
PowerMethod=csp
$
```

D.1.4 Perform customization on the node

The directories containing the configuration files and CSM installation code are NFS-exported from the management server. The specific exports are for /csminstall/AIX/csm and /csminstall/csm.

The configured remote shell command is then used to mount the directories on the node and execute the **updatenode.client** script.

D.1.5 A close look at the updatenode.client script

This script is in one of the exported directories (/csminstall/csm), so the latest level of code on the management server is always used. The script records its progress in the /var/log/csm/install.log file on the node being processed.

D.1.5.1 Identifying the configuration data

The script identifies the name used for this host in the CSM databases by looking for its hostname in the /csminstall/csm/config/nodemap file.

D.1.5.2 Processing the configuration data

Using the resolved name from the previous step, the code then reads the <node>.config_info file, in /csminstall/csm/config, to extract the configuration information. This includes the name of the management server. This configuration file is then copied to /opt/csm/install/configinfo.

D.1.5.3 Accept the node in the domain

The **makenode** command is executed, from the mounted /csminstall/csm directory, to complete the adoption of the node into the domain. The following processing steps occur as a result:

- Prior to CSM 1.3.2, the CSM filesets, in AIX installp format, are applied by issuing **installp -agX**.
- **/opt/csm/bin/mgmtsvr** is called. This script executes the **mkrsrc-api** command on the IBM.ManagementServer resource to configure it with the hostname of the management server.
- This triggers the two hosts to exchange their RSCT public keys via network communication between their RMC daemons, and to update their respective trusted host lists, /var/ct/cfg/ct_has.thl, for UNIX host-based authentication.
- On the management server, the CSM attributes InstallStatus and Mode, for the node, are changed to Managed, and the attribute AllowManagementRequest is reset to 0.
- The node's RMC access control list in /var/ct/cfg/ctrmc.acls is updated to authorize access to the managed node's resource managers from the CSM management server, and the node's ctrmc subsystem is refreshed to pick up these new settings.

Once the updatenode.client processing has completed the NFS exports for the node, they are removed.

D.1.6 Record update completion

An RSCT resource action is taken on the IBM.ManagedNode instance representing the node to update the CSM database with details of the node that has just been added to the domain. One of the attributes that is changed is CSMVersion, which holds details of the CSM code level on the node.

D.1.7 Distribute configuration files

As the last step, **updatenode** checks whether CFM needs to be run for the new node. This is done by default unless suppressed via the **updatenode** command line options, which are described in *CSM for AIX 5L: Command and Technical Reference,* SA22-7934.

Note: Contrary to the description of the **updatenode** command in *CSM for AIX 5L: Command and Technical Reference,* SA22-7934, user customization scripts are not supported until the CSM 1.3.2 release.

D.2 CSM installation details using a NIM install

CSM has support for installing nodes through NIM, if the CSM management server is also configured as a NIM server. If nodes are installed using NIM from a machine other than the management server, they must be manually incorporated into the management domain using the **updatenode** command.

Attention: There is no support in CSM 1.3 for an integrated installation using multiple NIM servers for a single management server domain.

D.2.1 Populating the NIM database from CSM

The **csm2nimnodes** command simply defines or updates NIM resources of type *machine* based on the node definitions in the CSM database. This script assumes that the NIM master is set up on the CSM management server and issues the resulting NIM commands **nim -o define -t standalone...** or **nim -o change -t standalone...** (refer to *AIX 5L V 5.2 Installation Guide and Reference,* SC23-4389). It also changes the InstallMethod attribute in the IBM.ManagedNode entry for each node processed to `nim` in the CSM database.

The **csm2nimgroups** command defines a NIM group with members matching the CSM group being processed. If the group already exists, the members will be updated to match the CSM group.

The lpp_source, spot, mksysb, and bosinst_data resources in NIM must be defined by the administrator as they are not handled by CSM (unlike PSSP).

D.2.2 The mechanics of integrated AIX and CSM installation via NIM

To integrate new nodes into a CSM management domain combined with a NIM install, the two steps of making CSM images available and run updatenode.client on the new nodes can be integrated with the NIM installation through the NIM script resource csmprereboot_script.

Before installing a node, NIM has to be configured to set up the CSM customization and the installation of the node, from mksysb or the AIX filesets (rte). The csmsetupnim script prepares CSM for the installation. It makes sure the csmprereboot_script resource is defined and allocates it to the nodes.

By directly interfacing with NIM in the CSM environment, the administrators can also define and allocate their own NIM script and fb_script resources, taking the place of PSSP's script.cust and firstboot.cust objects. The use of these resource types is covered in C.2.2.5, "script" on page 221 and C.2.2.7, "fb_script" on page 221.

D.2.3 A closer look at csmsetupnim

Let us look at what happens when **csmsetupnim** is invoked to prepare nodes for integrated installation of AIX and CSM through NIM, with the management server acting as NIM master.

A number of these steps match the **updatenode** processing. Where this is the case, we reference the relevant section.

D.2.3.1 Prepare RSCT

Refer to D.1.2, "Prepare RSCT" on page 232.

D.2.3.2 Remote shell setup

The ssh key generation described in D.1.1, "Remote shell setup" on page 232 is performed, when ssh is being used as the RemoteShell. The resulting public keys are then copied to /csminstall/csm/config/.ssh to make them available to the nodes during post-installation processing.

Note: **csmsetupnim** clears out all /.ssh/known_host entries referring to nodes listed as its parameters. If existing managed nodes of the CSM management domain are specified as arguments, and these nodes will not be immediately reinstalled, **dsh** to these nodes will require keyboard interaction because **ssh** will wait for keyboard input to confirm the connection request to the “unknown” nodes. If you run into this challenge, issue **updatenode -k** to these nodes to reimport their host keys into the /.ssh/known_hosts file on the management server.

D.2.3.3 Prepare managed node configuration data

The <node>.config_info file is generated, as described in D.1.3, “Prepare managed node configuration data” on page 233, and the NFS exports are also defined.

The nodemap file is not needed because the nodename will match the NIM client name.

D.2.3.4 Process the CSM NIM resources

The csmprereboot script is defined as a NIM script resource and allocated to the specified nodes, as shown in Example D-3.

Example: D-3 csmprereboot_script NIM resource

```
$ lsnim -l csmprereboot_script
csmprereboot_script:
   class       = resources
   type        = script
   Rstate      = ready for use
   prev_state  = unavailable for use
   location    = /csminstall/csm/csmprereboot
   alloc_count = 2
   server      = master
```

Thus, after running **csm2nimnodes** and **csmsetupnim**, a node’s NIM state typically looks similar to Example D-4.

Example: D-4 NIM state after **csmsetupnim**

```
$ lsnim -l node1
node1:
   class          = machines
   type           = standalone
   platform       = chrp
   netboot_kernel = mp
   if1            = ms_net node1 0004AC49C746 ent
```

```
   cable_type1    = N/A
   Cstate         = ready for a NIM operation
   prev_state     = BOS installation has been enabled
   Mstate         = currently running
   script         = csmprereboot_script
   cpuid          = 000132374C00
   control        = master
   Cstate_result  = reset
```

While the administrator might define additional NIM resources of type installp_bundle, script, fb_script, or adapter_def, this configuration is sufficient to prepare the NIM rte install of a node by running the bos_inst action on it. See Example D-5.

Example: D-5 NIM state after bos_inst operation

```
$ nim -o bos_inst -a bosinst_data=noprompt -a lpp_source=lppsource_aix520 -a
spot=spot_aix520 -a boot_client=no node1
$ lsnim -l node1
node1:
   class          = machines
   type           = standalone
   platform       = chrp
   netboot_kernel = mp
   if1            = ms_net node1 0004AC49C746 ent
   cable_type1    = N/A
   Cstate         = BOS installation has been enabled
   prev_state     = ready for a NIM operation
   Mstate         = currently running
   boot           = boot
   bosinst_data   = noprompt
   lpp_source     = lppsource_aix520
   nim_script     = nim_script
   script         = csmprereboot_script
   spot           = spot_aix520
   cpuid          = 000132374C00
   control        = master
```

The node can then be rebooted such that a network boot is attempted using bootp. This can be done by modifying the bootlist of an existing server, and rebooting, or using the SMS or firmware menus of the node. If you have full CSM hardware control (HMC or supported CSP servers), you can start the boot by using the **netboot** command.

D.2.4 NIM installation processing

After the installation of the AIX operating system through NIM has completed, but before the installation process finally reboots the node, the install adapter is configured by NIM's c_mk_nimclient. NIM will then process any additional resources allocated to the node, including the csmprereboot_script script resource that was allocated by **csmsetupnim**. These resources are defined in the node-specific script located in the directory referenced by the nim_script resource. See Example D-6.

Example: D-6 simple **nim_script** *resource*

```
$ cat /export/nim/scripts/node1.script
#!/bin/ksh

# nim_script for node1
result=success

export NIM_NON_41_MASTER=yes
export NIM_SCRIPT=yes
# NIM client initialization
/../SPOT/usr/lpp/bos.sysmgt/nim/methods/c_mk_nimclient ${VERBOSE} \
        -ahostname=node1 \
        -aip=192.168.1.1 \
        -acable_type=N/A \
        -asnm=255.255.255.0
[[ $? != 0 ]] && result=failure

# script=csmprereboot_script
/../SPOT/usr/lpp/bos.sysmgt/nim/methods/c_script ${VERBOSE} \
        -alocation="sp6cws-en0:/csminstall/csm/csmprereboot"

script_rc=$?
[[ ${script_rc} != 0 ]] && result=failure

[[ ${result} = success ]] && exit 0 || exit 1
```

D.2.5 A closer look at csmprereboot

The csmprereboot script executes in a restricted NIM environment (see C.2.2.5, "script" on page 221), so it performs limited functions. It logs to /var/log/csm/install.log on the node.

D.2.5.1 Identifying the configuration data

The csmprereboot script assumes that the NIM master has been configured on the CSM management server. Based on this assumption, csmprereboot

determines the management server for the node and the node's hostname <node> as known by the management server by looking for the NIM_MASTER_HOSTNAME and NIM_HOSTNAME entries in the file /etc/niminfo, which is left behind on the node by the NIM install.

The /csminstall/csm directory can then be NFS-mounted from the CSM management server to provide access to the configuration file prepared by **csmsetupnim**.

D.2.5.2 Processing the configuration data

Using the value of <node> resolved in the previous step, the processing performed matches that are shown in D.1.5.2, "Processing the configuration data" on page 234.

D.2.5.3 Prepare csmfirstboot for execution

The csmfirstboot code, and the libraries and commands it uses, are then copied from the NFS-mounted /csminstall/csm directory to the /opt/csm/install directory on the node.

The /etc/inittab file is then updated so that the csmfirstboot script will be executed following the reboot of the node. The /csminstall/csm NFS mount is then unmounted.

Note: The csmfirstboot script is not defined and run as a NIM fb_script resource, although its name might suggest otherwise. As pointed out in C.2.2.7, "fb_script" on page 221, NIM just adds the contents of an fb_script resource (if allocated) to the client's /etc/firstboot Korn shell script. This script is invoked very early in /etc/inittab by the fbcheck entry exactly once on the first reboot after installation. Since csmfirstboot is not a Korn shell script and it requires the network to be available, it cannot be executed as an fb_script resource.

D.2.6 Additional NIM processing

If any other NIM script resources had been allocated to the node by the CSM administrator, they would also be called from <node>.script, generated for the node by the NIM bos_inst operation. There are examples of this in C.4, "Using NIM to install and configure LDAP clients" on page 223 and C.5, "A working example of openssh installation" on page 227.

Note: At this stage before the reboot, the node is still in a single user environment with a RAM file system in place, and no daemons have been started.

After finishing with all scripts provided as NIM script resources, the node reboots; when it comes up again, the NIM install network adapter is configured by NIM during the installation. Due to the assumption that the NIM master is also the CSM management server, network connectivity into the CSM management LAN is thus guaranteed. Also, the NIM installation has left behind a rudimentary /etc/hosts containing an entry for the NIM master/CSM management server. Processing of the /etc/inittab in this environment after the reboot will then eventually start csmfirstboot.

D.2.7 A closer look at the csmfirstboot script

The function of the csmfirstboot script is similar to the psspfb_script in PSSP. It is called after the node has been installed and rebooted, and the network is available to the NIM server.

The configuration data for the node is read from the /opt/csm/install/configinfo file on the node. This identifies the management server so that the /csminstall/csm and /csminstall/AIX/csm directories can be NFS-mounted.

D.2.7.1 Accept the node in the domain

The processing here is as described in D.1.5.3, “Accept the node in the domain” on page 234.

D.2.7.2 Complete remote shell configuration

If ssh is configured for CSM’s RemoteShell option and ssh has been set up for install as part of the NIM installation, the node host public ssh key is appended to the management server’s /.ssh/known_hosts file.

If SetupRemoteShell=1 in /opt/csm/install/configinfo (SetupRemoteShell=0 is found if RemoteShell is configured to ssh but the management server’s root keys could not be created and successfully copied to /csminstall/csm/config/.ssh), remote command authorization for the management server’s root on the node is established by either adding a corresponding entry to /.rhosts or executing the scriptlet .ssh/copy.perl in the NFS-mounted /csminstall/csm/config. This copies the management server’s root public keys to /.ssh/authorized_keys and /.ssh/authorized_keys2.

D.2.7.3 SP node-specific processing

If CSP has been configured as the power method for the node, csmfirstboot adds the clocal attribute to tty0’s runmodes and logmodes to assume a line without modem control.

In order to free up tty0 to do this, csmfirstboot temporarily redirects the node’s console to a file, and removes the cons: entry from /etc/inittab. Any getty

processes using tty0 are also killed. The tty0 device can then be changed, and the cons: entry is recreated so that getty will restart using the new settings.

D.2.7.4 Post-install cleanup

If the post-install cleanup executes without error, csmfirstboot removes itself from the end of /etc/inittab on the managed node. If there are errors, it leaves the entry so that it will execute again on the next reboot.

D.3 Comparison with PSSP

To contrast the installation process in CSM, as examined in D.2, “CSM installation details using a NIM install” on page 235, with that in PSSP, let us briefly recall some key points regarding node installation in PSSP. For more details, refer to Chapter 1, “Overview of the installation and migration processes” in *PSSP Installation and Migration Guide,* GA22-7347. For simplicity, we limit the discussion to a cluster without boot/install servers.

While the PSSP installation process is built on the use of NIM, with the control workstation as a NIM master, the PSSP administrator usually does not *directly* interface with NIM. After a node configuration has been entered into the SDR, the setup of NIM as well as the preparation of the security environment (depending on the PSSP security employed, registration of new Kerberos principals, and updating remote command authorization files) are achieved by the **setup_server** command. This is done after the node’s bootp response in the SDR has been set to install with the **spbootins** command.

The **setup_server** script calls many other PSSP scripts such as **mknimclient**, **mknimmast**, **mknimres**, **allnimres**, and **setup_CWS**, which accomplish the definition of the NIM and security environment.

PSSP only utilizes mksysb netinstalls. PSSP defines only one NIM script resource, pssp_script in /usr/lpp/ssp/install/bin. When a node is being installed, this script is run in the single user environment after the mksysb image has been restored but before the node reboots; in this respect, **pssp_script** is the equivalent of **csmprereboot**. However, **pssp_script** does a lot more work at this stage than **csmprereboot**.

pssp_script is also executed on a PSSP node when the node has been set to “customize” in the SDR. In this case, **pssp_script** is called from the rc.sp entry in /etc/inittab after a reboot, or if the administrator manually issues the **pssp_script** command on the node.

Note that such node customization in PSSP needs to be clearly distinguished from the NIM cust operation. NIM customization assumes that a script,

adapter_def or lppsource, and installp_bundle have been allocated to a NIM client node; the NIM master then issues remote commands to the nodes running these scripts, changing adapters and/or installing filesets specified in the resources. Currently this necessitates that the NIM master is authorized to access the clients with the **rsh** command.

When **pssp_script** is executed during customization, at some places it behaves differently than when run in the single user environment directly after the restore of the mksysb has finished. To discover under which circumstances it is being run, **pssp_script** checks whether the NIM_CUSTOM environment variable is null—in PSSP, this variable should only be defined during an install.

Note that this test cannot be used in a CSM/NIM script resource. If a CSM/NIM script needs to learn in which environment it is being run, the PPID test as demonstrated in C.4, "Using NIM to install and configure LDAP clients" on page 223 can be used.

In addition to other things, **pssp_script** does the following:

- Defines the SPLAN adapter, hostname, and adds start_net to /etc/inittab.
- Installs PSSP filesets and their AIX prerequisites.
- Stores the node's nodenumber in the ODM.
- Obtains kerberos configuration data.
- Updates /etc/hosts.
- Removes the /.rhosts entry for the NIM master's root user.
- Configures the switch if installed.
- Creates a dump device and starts the root volume group (un)mirroring.
- If a **script.cust** or **tuning.cust** have been provided by the administrator in /tftpboot on the control workstation, it obtains these scripts from the NIM master through TFTP and runs **script.cust**. The **tuning.cust** script is only run when customizing; otherwise it is run by rc.sp from /etc/inittab after a reboot.

  Note: Tuning on AIX 5.2, PSSP client does not utilize tuning.cust. It is configured through the file /etc/tunables/nextboot.

- Retrieves the **psspfb_script** via TFTP. During an install, it adds an entry after rc.tcpip to /etc/inittab so **psspfb_script** will be run on the first reboot. During customization, **pssp_script** runs **psspfb_script** directly.

On the node's first reboot after an installation, first start_net (called early in /etc/inittab) configures the SPLAN adapter. **psspfb_script** then does the following:

- Unconfigures and configures all secondary network adapters.
- If Kerberos 4 is configured, it gets srvtabs for the rcmd and hardmon principals from the control workstation using the serial connection to the node.
- Sets up remote command authorization on the node (through calls to **spauthconfig**/**updauthfiles** in /usr/lpp/ssp/bin).
- Sets the clocal flag for tty0 on SP nodes.
- Updates the SDR and sets the node's bootp response to disk.
- Acquires the script /tftpboot/**firstboot.cust** through TFTP from the NIM master, if provided by the PSSP administrator, and runs it **(firstboot.cust** is not run when **psspfb_script** has been called by **pssp_script** during customization).
- Removes itself from the inittab.

D.4 Using alternate NIM servers

The CSM customization with NIM is only supported when the management server is also a NIM server. We examined the interactions between CSM and NIM and believe that CSM can be used to define machines on other NIM servers. We provide details in this section.

Note: These suggestions are theoretical only; we did not have time to validate them on test systems.

Thus, if you need to install a very large number of nodes simultaneously, you might want to consider setting up several NIM servers and then, after nodes have been installed from these various NIM servers, issue **updatenode -P** separately to prepare the nodes for management. For this, make sure the appropriate version of CSM client filesets (csm.client, csm.core, and csm.dsh) are already contained in the NIM masters' lppsource. The CSM filesets are automatically installed by an rte install and can be installed using an installp_bundle resource for mksysb installs.

D.4.1 Defining NIM machine resources

If we want to import machine definitions to a different NIM server, in CSM 1.3.1, we can simply modify the **csm2nimnodes** Perl script to echo the NIM commands to STDOUT and pipe them to a file that we can then source on another NIM server (or, instead of just echoing, directly issue these commands to the alternate NIM master via rsh/ssh), as shown in Example D-7.

Example: D-7 Extracting CSM node information for NIM

```
$ cd /opt/csm/bin
$ cat csm2nimnodes | sed -e "s/\$::NIM/echo \$::NIM/" -e\
"s/!\$::CLIENT_EXISTS/1/" > csm2nimnodes.STDOUT
$ chmod 700 csm2nimnodes.STDOUT
$ ./csm2nimnodes.STDOUT -n node1,node3,node5 cable_type=tp
/usr/sbin/nim -o define -t standalone -a netboot_kernel=mp -a platform=chrp -a
if1=ms_net node5 0O6094E94F8F ent -a cable_type1=tp node5
/usr/sbin/nim -o define -t standalone -a netboot_kernel=mp -a platform=chrp -a
if1=ms_net node1 0OO4AC49C746 ent -a cable_type1=tp node1
/usr/sbin/nim -o define -t standalone -a netboot_kernel=mp -a platform=chrp -a
if1=ms_net node3 0OO4AC49BA46 ent -a cable_type1=tp node3
```

Similarly, we can change the csm2nimgrps Perl script to write to STDOUT as shown in Example D-8. In Example D-7, notice that the resulting NIM commands might need to be modified before issuing them on an alternate NIM master if the respective NIM objects already exist on that server.

Example: D-8 Extracting CSM nodegroup information for NIM

```
$ cd/opt/csm/bin
$cat csm2nimgrps | sed -e "s/\$::NIM/echo \$::NIM/" -e\
"s/!\$::GROUP_EXISTS/0/" > csm2nimgrps.STDOUT
$ chmod 700 csm2nimgrps.STDOUT
$ ./csm2nimgrps.STDOUT -N AIXNodes
/usr/sbin/nim -o define -t mac_group -a add_member=node1 -a add_member=node3 -a
add_member=node5 -a add_member=node7 -a add_member=node9 -a add_member=node10
AIXNodes
/usr/sbin/nim -o change -a add_member=node11 -a add_member=node12 -a
add_member=node13 -a add_member=node14 AIXNodes
```

Instead of hacking the csm2nimnodes and csm2nimgrps scripts (whose internal flows might change with the next release of CSM), you could use these scripts to generate the NIM definitions on the management server and then extract the NIM definitions through NIM commands for setting up the NIM objects on the alternate NIM masters.

As part of the installation, NIM configures the install adapter on such a node so that network connectivity to nodes is available to the NIM server. When this is

also the CSM management server, connectivity is assured for the post-installation processing. If a different NIM server is used, it may be necessary to use *NIM script* or *fb_script* resources to establish a network route to the CSM management server to allow the CSM post-installation code to execute.

Abbreviations and acronyms

ACL	Access Control List
BPA	Bulk Power Adapter
CEC	Central Electronics Complex
CFM	Configuration File Manager
CHRP	Common Hardware Reference Platform
CIMOM	Common Information Model Object Manager
CRM	Configuration Resource Manager
CSM	Cluster Systems Management
CSP	(1) Converged Service Processor (2) Cluster SP-Node Support
CtSec	Cluster Security Services (RSCT)
CU	Cluster Utilities (RSCT)
CWS	Control Workstation
DCEM	Distributed Command Execution Manager
DMSRM	Domain Resource Manager (an RM provided by CSM)
ERRM	Event Response Resource Manager (RSCT)
ESP	Equinox Ethernet Serial Provider
GPFS	General Parallel File System
HBA	Host Based Authentication
HCP	Hardware Control Point (HMC; tty device for CSP)
HMC	Hardware Management Console
HPC	High Performance Computing
IBM	International Business Machines Corporation
IETF	Internet Engineering Task Force
ISB	Intermediate Switch Board
ITSO	International Technical Support Organization
IW	Independent Workstation (RSCT)
LPP	Licensed Program Product
MPM	Mechanism Pluggable Module (CtSec)
NIM	(1) Network Install Manager (2) Network Interface Module (RSCT topology services)
NSB	Node Switch Board
NTP	Network Time Protocol
RM	Resource Manager (RSCT)
RMC	Resource Monitoring and Control (RSCT)
RPD	RSCT Peer Domain
RPM	Redhat Package Manager
RRA	Restricted Root Access
RSA	Remote Supervisor Adapter
RSA	Rivest-Shamir-Adleman cryptosystem
RSCT	Reliable Scalable Cluster Technology
SDR	System Data Repository
SR	System Registry (RSCT)
SSH	Secure SHell
VSD	Virtual Shared Disks

E

Additional material

This redbook refers to additional material that can be downloaded from the Internet as described below.

E.1 Locating the sample code

The code samples referenced in this redbook are available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

ftp://www.redbooks.ibm.com/redbooks/SG246953

Alternatively, you can go to the IBM Redbooks Web site at:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the redbook form number, SG246953. The code is stored in files of the form <chapter>.samples.tar, where <chapter> is the chapter of the redbook that references the samples. Some examples are `AppendixB.samples.tar` and `Chapter3.samples.tar`.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 253. Note that some of the documents referenced here may be available in softcopy only.

- *IBM (e)server Cluster 1600 Managed by PSSP 3.5: What’s New,* SG24-6617
- *An Introduction to CSM 1.3 for AIX 5L,* SG24-6859
- *An Introduction to Security in a CSM 1.3 for AIX 5L Environment,* SG24-6873
- *A Practical Guide for Resource Monitoring and Control (RMC),* SG24-6615
- *RS/6000 SP Cluster: The Path to Universal Clustering,* SG24-5374
- *NIM: A to Z in AIX 4.3,* SG24-5524
- *Secure Network Implementations on AIX - VPN, LDAP, PAM,* SG24-6066
- *Managing IBM (e)server Cluster 1600 - Power Recipes for PSSP 3.4,* SG24-6603
- *IBM Cluster 1600 and PSSP 3.4 Cluster Enhancements,* SG24-6604
- *Performance and Tuning Considerations for the p690 in a Cluster 1600,* SG24-6841
- *CSM for the PSSP System Administrator,* SG24-6953
- *IBM pSeries 670 and pSeries 690 System Handbook,* SG24-7040
- *Configuring p690 in an IBM (e)server Cluster 1600,* REDP-0187

Other publications

These publications are also relevant as further information sources:

- *CSM for AIX 5L: Command and Technical Reference,* SA22-7934
- *AIX 5L V 5.2 Installation Guide and Reference,* SC23-4389
- *PSSP Installation and Migration Guide,* GA22-7347

- *RSCT for AIX 5L: Guide and Reference,* SA22-7889
- *RSCT for AIX 5L: Messages,* GA22-7891
- *RSCT for AIX 5L: Technical Reference,* SA22-7890
- *CSM for AIX 5L: Hardware Control Guide,* SA22-7920
- *CSM for AIX 5L: Software Planning and Installation Guide,* SA22-7919
- *CSM for AIX 5L: Administration Guide,* SA22-7918
- *Hardware Management Console for pSeries Installation and Operations Guide,* SA38-0590
- *Hardware Management Console for pSeries Maintenance Guide,* SA38-0603
- *AIX 5L Version 5.2 Commands Reference, Volume 4, N-R,* SC23-4118
- *Systems Management Guide: Operating System and Devices,* SC23-4126
- *Network Installation Guide and Reference,* SC23-4385
- *AIX Security Guide,* SC23-4850
- *AIX 5L V 5.2 Security Guide,* SC23-4860-01
- *RS/6000 SP: Planning, Volume 1, Hardware and Physical Environment,* GA22-7280
- *RS/6000 SP Planning, Volume 2: Control Workstation and Software Environment,* GA22-7281
- *PSSP Administration Guide,* SA22-7348
- *PSSP Managing Shared Disks,* SA22-7349
- *PSSP Diagnosis Guide,* GA22-7350
- *PSSP Command and Technical Reference (2 Volumes),* SA22-7351
- *PSSP Messages Reference,* GA22-7352
- *CSM for Linux: Hardware Control Guide,* SA22-7856
- *IBM (e)server Cluster 1600: Planning, Installation, and Service,* GA22-7863
- *CSM for Linux: Administration Guide,* SA22-7873
- *PSSP Implementing a Firewalled RS/6000 SP System,* GA22-7874
- *IBM RSCT: Group Services Programming Guide and Reference,* SA22-7888
- *RSCT for Linux: Guide and Reference,* SA22-7892
- *RSCT for Linux: Technical Reference,* SA22-7893
- *RSCT for Linux: Messages,* GA22-7894

Online resources

These Web sites and URLs are also relevant as further information sources:

- IBM Alphaworks website

  `http:`http://www.alphaworks.ibm.com/
- Latest PSSP “Read This First” document

  `http://`www.ibm.com/servers/eserver/pseries/library/sp_books/pssp.html
- Latest CSM “Read This First” document

  `http://`www.ibm.com/servers/eserver/pseries/library/clusters/aix.html
- RSCT fixes can be downloaded from

  `http://techsupport.services.ibm.com/server/aix.fdc`
- CSM updates and fixes can be downloaded from

  `http://techsupport.services.ibm.com/server/cluster/fixes`
- AutoUpdate and and Perl-libnet can be downloaded from

  `http://freshmeat.net/projects/autoupdate`
- openCIMOM can be downloded from

  `http://www.ibm.com/servers/aix/products/aixos/linux/download.html`
- The AIX toolbox for Linux applications can be downloaded from

  `http://www.ibm.com/servers/aix/products/aixos/linux/download.html`

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

`ibm.com``/redbooks`

Index

M

N

P

R

S

CSM Guide for the PSSP System Administrator

(0.5” spine)
0.475”<->0.875”
250 <-> 459 pages

CSM Guide for the PSSP System Administrator

Compares management tasks in PSSP and CSM

Explores CSM features and commands

Provides hints and tips

This IBM Redbook compares and contrasts the methods, tools, and interfaces provided by PSSP and CSM to perform typical system administrator tasks. It is aimed at experienced PSSP system administrators who need to grow their knowledge about administering of PSSP clustered systems into equivalent knowledge about CSM clustered systems, regardless of whether or not they need to convert a system from PSSP to CSM system management.

This publication is a "task-oriented guide" that will help seasoned PSSP administrators to quickly figure out how to accomplish, in the new CSM environment, tasks they are already familiar with in PSSP.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-6953-00 ISBN 0738453064