

Parallel System Support Programs for AIX



Administration Guide

Version 3 Release 1.1

Parallel System Support Programs for AIX



Administration Guide

Version 3 Release 1.1

Note!

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

Second Edition (October 1999)

This edition applies to version 3 release 1 modification 1 of the IBM Parallel System Support Programs for AIX (PSSP) Licensed Program (product number 5765-D51) and to all subsequent releases and modifications until otherwise indicated in new editions. This edition replaces SA22-7348-00. Significant changes or additions to the text and illustrations are indicated by a vertical line (|) to the left of the change.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address below.

IBM welcomes your comments. A form for readers' comments may be provided at the back of this publication, or you may address your comments to the following address:

International Business Machines Corporation
Department 55JA, Mail Station P384
522 South Road
Poughkeepsie, NY 12601-5400
United States of America

FAX (United States & Canada): 1+914+432-9405
FAX (Other Countries):
Your International Access Code +1+914+432-9405

IBMLink (United States customers only): IBMUSM10(MHVRCFS)
IBM Mail Exchange: USIB6TC9 at IBMMAIL
Internet e-mail: mhvrdfs@us.ibm.com

If you would like a reply, be sure to include your name, address, telephone number, or FAX number.

Make sure to include the following in your comment or note:

- Title and order number of this book
- Page number or topic related to your comment

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 1998, 1999. All rights reserved.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

	Notices	xiii
	Trademarks	xiv
	Publicly Available Software	xv
	About This Book	xvii
	Who Should Use This Book	xviii
	Typographic Conventions	xviii
	Interface Instructions	xviii
<hr/>		
	RS/6000 SP Overview	1
	Chapter 1. RS/6000 SP Overview	3
	Introduction	3
	Hardware Overview	4
	Software Overview	8
<hr/>		
	Routine System Administration Tasks	15
	Chapter 2. Starting Up and Shutting Down the SP System	17
	Overview of System Startup and Shutdown	17
	Controlling the Sequence of Startup and Shutdown Operations	17
	Starting the SP System	18
	Shutting Down the SP System	21
	Checking the Status of Startup and Shutdown Requests	24
	Defining Startup and Shutdown Sequence Files	25
	Creating a Special Subsystem	28
	Chapter 3. Parallel Management Commands	31
	Using dsh to Execute Parallel Management Commands	31
	The hostlist Command	33
	List of Parallel Management Commands	34
	Chapter 4. Managing File Collections	41
	Understanding File Collections	42
	Reporting File Collection Information	50
	Verifying File Collections Using scan	51
	Updating Files in a File Collection	52
	Granting Permission to Obtain a File Collection	54
	Adding and Deleting Files in a File Collection	56
	Building a File Collection	58
	Installing a File Collection	61
	Refusing Files in a File Collection	62
	Modifying the File Collection Hierarchy	63
	Removing a File Collection	65
	super list File Syntax, Keywords, and Operands	66
	Chapter 5. Managing User Accounts	69
	Understanding SP User Management	69
	Adding User Accounts	76

Changing User Accounts	77
Removing User Accounts	78
Listing a User Account	78
Changing Passwords	78
Controlling User Login	81
Chapter 6. Managing SP Resources	87
Chapter 7. Managing Time Synchronization	89
Managing NTP	89
Time Setting Options	91
Nodes and Control Workstations Out of Time Synchronization	91
Chapter 8. Managing the Automounter	93
Understanding the Automounter Implementation	93
Establishing an Automount Map for Users' Directories	99
Using Automount for Other File Systems	99
Customizing the SP Automounter Function	103
Chapter 9. Managing Mail Service	113
Chapter 10. Accounting	115
Accounting by Node or by Class	115
Setting Up SP Accounting	117
Accounting Files	122
Chapter 11. Maintaining the crontabs File	127
Updating crontabs on the SP Nodes	127

Security on the SP System 129

Chapter 12. Security Features of the SP System	131
Basic Security Concepts	131
Suggested Reading	132
Terminology	133
Conceptual Information Regarding Kerberos V4	135
Conceptual Information Regarding DCE	142
Installing and Configuring Distributed Authentication Services	144
Administering Security for Kerberos V4	153
Managing Your Authentication Credentials When Using Kerberos	168
Using the AIX Remote Commands	173
Using AFS to Provide Kerberos V4 Authentication	174
Chapter 13. Sysctl	177
Sysctl Components	177
Relationship to the SP Authentication Services	179
Overview	179
Installation and Configuration	182
Using Sysctl	188
Developing a Sysctl Application	190
Sysctl Scripts	193
Sysctl Reference Information	196

Managing Sets of Nodes	201
Chapter 14. Using a Switch	203
Understanding the Switch and the Supported Switch Types	203
Switch Communication Modes	204
Primary Node Takeover	206
System Partitioning and the Switch	206
Extension Nodes and the SP Switch	207
SP-Attached Servers	207
Selecting Appropriate Switch Clocks	208
Selecting the Primary and Primary Backup Nodes	209
Managing the Switch Topology File	211
Starting the Switch	213
Determining Switch Connectivity	214
Automatic Node Unfence	214
Preparing an SP Switch for Repartitioning	216
Monitoring Nodes on a Switch	216
Global Shutdowns and Reboots of Nodes with a Switch	219
NFS Mounts Over a Switch	220
Chapter 15. Managing System Partitions	221
Understanding System Partitioning	221
Partitioning the SP System	224
Displaying System Partition Configuration Information	239
Verifying System Partition Configuration	241
Managing System Partition-Sensitive Subsystems Using <code>syspar_ctrl</code>	242
Chapter 16. Managing Node Groups	245
Understanding Node Groups	245
Working with Node Groups	247
Using Node Groups with Commands	249
Chapter 17. Managing Extension Nodes	251
Understanding Extension Nodes	251
Specific Implementations of Extension Nodes and Extension Node Adapters	251
Transferring Extension Node Configuration Information	253
Adding Extension Nodes and Extension Node Adapters	253
Changing Extension Nodes and Extension Node Adapters	255
Removing Extension Nodes and Extension Node Adapters	256
Reconfiguring Extension Nodes and Extension Node Adapters	257
Resetting Extension Nodes	258
Listing Extension Node and Extension Node Adapter Information	259
Monitoring and Controlling the SP System	261
Chapter 18. Using SP Perspectives	263
Getting Started	264
Understanding the SP Perspectives Interface	267
Accessing Help	274
Performing Common Administration Tasks with SP Perspectives Previously Performed Using System Monitor GUI	275

Chapter 19. Using the SP Command Line System Monitor	279
Understanding the Hardware Monitor	280
Configuring the SP System Monitor	282
Using the Command Line Interfaces	283
Chapter 20. Integrating TME 10 on the SP System	285
Related Publications	285
TME 10 Terminology	285
Understanding the TME 10 Enterprise Console	286
Defining the PSSP T/EC Adapter	287
Defining an Event and Forwarding it to the T/EC Using the Event Perspective	287
Defining an Event and Forwarding it to the T/EC Using the pmandef Command	288
Debugging Problems with Event Generation and Reception by the T/EC	288
Chapter 21. Managing an SP-Controlled Netfinity Server	289
Netfinity Management Interfaces	289
Managing Netfinity Nodes from Perspectives	290

Availability and Problem Management 293

Chapter 22. Managing a High Availability Control Workstation Configuration	295
Understanding HACWS Function	296
Managing SP Subsystems for High Availability Control Workstation Function	304
Customizing HACMP	310
Verifying Frame Supervisor Cables	312
Chapter 23. The Topology Services Subsystem	313
Introducing Topology Services	313
Topology Services Components	314
Components on Which Topology Services Depends	317
Configuring and Operating Topology Services	318
Topology Services Procedures	323
Chapter 24. The Group Services Subsystem	325
Introducing Group Services	325
Group Services Components	326
Components on Which Group Services Depends	330
Configuring and Operating Group Services	330
Group Services Procedures	334
Chapter 25. The Event Management Subsystem	337
Introducing Event Management	337
Event Management Components	338
Components on Which Event Management Depends	348
Configuring and Operating Event Management	349
Event Management Procedures	355
Chapter 26. Using the Problem Management Subsystem	361
Understanding the Problem Management Daemon	363
Understanding the Problem Management Resource Monitor Daemon	370

Chapter 27. Managing SP System Events in a Network Environment . . .	373
Understanding Network Management	374
Enabling Network Management for the SP System	375
Disabling Network Management for the SP System	380
Configuring the SP Proxy Agent	380
Configuring NetView for AIX for the SP System	381
Chapter 28. Managing Error Logs	383
Installing and Configuring Error Log Management	383
Managing the AIX Error Log Facility	384
Managing the BSD syslog Facility	388
Viewing Error Logs	390
Archiving Error Logs	390
Collecting System Data for IBM Service	391

The Communications Low-Level Application Programming Interface 393

Chapter 29. Understanding and Using the Communications Low-Level Application Programming Interface (LAPI)	395
Introducing the LAPI	395
Understanding the LAPI	396
Using the LAPI	401

Appendixes 433

Appendix A. Managing Root Volume Groups	435
Terminology	435
Using Mirrored Root Volume Groups	437
Using Alternate Root Volume Groups	439
Appendix B. SP Daemons	441
Appendix C. Sample Files	443
afsclient.cust	443
block_usr_sample	445
bootptab.info File	446
ibmSPDepNode.my	446
ibmSPMIB.my	452
Appendix D. Listing SP System Resource Variables	473
Looking at the Event Management Load List Files	473
Getting Information about Event Management Objects in the SDR	473
Displaying Event Management Descriptions	476
Getting Resource Variable Information from SP Perspectives	476
Appendix E. The System Data Repository	477
Understanding the SDR	477
Accessing the SDR	479
Updating the host_responds Class (the hrd Daemon)	485
Backing Up the SDR	486
SDR Shadow Files	486
System Data Repository Classes and Attributes	486

Appendix F. IP Address and Host Name Changes for SP Systems	517
SDR Objects with IP Address or Host Names	517
SP System Files with IP Address and Host Names	518
Changing the IP Address and Host Names for PSSP Systems	520
Glossary of Terms and Abbreviations	531
Bibliography	539
Finding Documentation on the World Wide Web	539
Accessing PSSP Documentation Online	539
Manual Pages for Public Code	540
RS/6000 SP Planning Publications	540
RS/6000 SP Hardware Publications	540
RS/6000 SP Switch Router Publications	540
RS/6000 SP Software Publications	541
AIX and Related Product Publications	542
Red Books	543
Non-IBM Publications	543
Index	545

Figures

1.	Basic SP Configuration	4
2.	SP Default File Collection Hierarchy	46
3.	/var/sysman/sup Files and Directories	46
4.	sup.admin Master Files	47
5.	SP Hierarchy of an Offline Boot/install Server	64
6.	Example of Adding an SP User	77
7.	Accounting Template for crontabs	122
8.	An Overview of Kerberos Authentication	141
9.	Sysctl Elements	178
10.	User Space Message Passing	205
11.	User Space Message Passing	205
12.	The Change/Show Switch Clock Source Settings SMIT Menu	208
13.	Initial Values	210
14.	Estart Values	210
15.	Backup Takeover Values	210
16.	Primary Takeover Values	211
17.	Archive the System Data Repository	227
18.	Select System Partition Configuration	229
19.	Select System Partition Configuration	230
20.	Display Information for Given Configuration or Layout menu	231
21.	Display Information for the Selected Configuration	231
22.	Select a System Partition Layout	232
23.	Select a System Partition Layout	233
24.	Customize a System Partition	234
25.	Customize a System Partition	234
26.	Apply System Partition Configuration menu	236
27.	Apply System Partition Configuration SMIT panel	238
28.	List System Partition Database Information Output	241
29.	Verify System Partition Configuration SMIT Panel	242
30.	The RS/6000 SP Switch Router and SP Switch Router Adapter Extension Node Implementation	252
31.	The SP Perspectives Launch Pad	268
32.	Elements of an SP Perspectives Window	270
33.	The SP Perspectives Hardware Perspective	271
34.	An SP Perspectives Dialog Box	272
35.	An SP Perspectives Notebook	273
36.	An SP Perspectives Message Box	273
37.	The SP System Monitor	280
38.	An HACWS Configuration	297
39.	Addressing in an HACWS Configuration	300
40.	HACWS Pre- and Post-Events	311
41.	An Example of Problem Management Daemon Configuration	363
42.	A LAPI Thread Model	404
43.	Sample Output from PSSP SDRGetObjects commands	475
44.	Comparing Two SDR Classes	477

Tables

1.	Specifying Nodes to Start with cstartup	20
2.	Specifying Nodes to Shut Down with cshutdown.	21
3.	New LoadLeveler Functions Previously Part of the Resource Manager	87
4.	New Job Switch Resource Table Functions Previously Part of the Resource Manager	88
5.	Time Service Choices	89
6.	Authorization Variables	193
7.	Built-in Tcl Commands and Their Default Authorization Callbacks	197
8.	Built-in Extended Tcl Commands and Their Default Authorization Callbacks	197
9.	ACL Processing Commands	198
10.	Service Commands	198
11.	Other Commands	199
12.	Perspective High-Level Tasks	263
13.	Authorization required for each Perspective	265
14.	System Monitor to Perspectives Cross Reference	275
15.	High Availability Control Workstation Configuration and Task Summary	304
16.	High Availability Control Workstation and Task Summary (SP-attached servers)	304
17.	SDR Authorizations	479
18.	Commands Invoked by SMIT Panels	483

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.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

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.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Department LJEB/P905
522 South Road
Poughkeepsie, NY 12601-5400
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

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.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

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

AIX
AIX/6000
DATABASE 2
DB2
ES/9000
ESCON

HACMP/6000
IBM
IBMLink
LoadLeveler
Netfinity
Netfinity Manager
NQS/MVS
POWERparallel
POWERserver
POWERstation
RS/6000
RS/6000 Scalable POWERparallel Systems
Scalable POWERparallel Systems
SP
System/370
System/390
TURBOWAYS

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.

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

Tivoli Enterprise Console is a trademark of Tivoli Systems Inc. in the United States, other countries, or both.

UNIX is a registered trademark in the United States, other countries, or both and is licensed exclusively through X/Open Company Limited.

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

Publicly Available Software

This product includes software that is publicly available:

expect Programmed dialogue with interactive programs
Kerberos Provides authentication of the execution of remote commands
NTP Network Time Protocol
Perl Practical Extraction and Report Language
SUP Software Update Protocol
Tcl Tool Command Language
TclX Tool Command Language Extended
Tk Tcl-based Tool Kit for X-windows

This book discusses the use of these products only as they apply specifically to the RS/6000 SP system. The distribution for these products includes the source code and associated documentation. (Kerberos does not ship source code.)

/usr/lpp/ssp/public contains the compressed **tar** files of the publicly available software. (IBM has made minor modifications to the versions of Tcl and Tk used in

| the SP system to improve their security characteristics. Therefore, the IBM-supplied
| versions do not match exactly the versions you may build from the compressed **tar**
| files.) All copyright notices in the documentation must be respected. You can find
| version and distribution information for each of these products that are part of your
| selected install options in the **/usr/lpp/ssp/README/ssp.public.README** file.

About This Book

This book contains information to help you understand and perform SP system administration. It includes concepts and instructions pertaining to:

- Understanding IBM RS/6000 Scalable POWERparallel (SP) systems
- Performing routine system administration tasks
- Making the system secure
- Managing sets of nodes
- Monitoring the system
- Providing for system availability and problem management
- Understanding and using the Communications Low-level Application Programming Interface (LAPI)

Additional task information for system administrators is covered in separate manuals.

For a list of related books and information about accessing online information, see the Bibliography in the back of the book.

This book applies to PSSP Version 3 Release 1 Modification 1. To find out what version of PSSP is running on your control workstation (node 0), enter the following:

```
sp1st_versions -t -n0
```

In response, the system displays something similar to:

```
0 PSSP-3.1.1
```

If the response indicates **PSSP-3.1.1**, this book applies to the version of PSSP that is running on your system.

To find out what version of PSSP is running on the nodes of your system, enter the following from your control workstation:

```
sp1st_versions -G -t
```

In response, the system displays something similar to:

```
1 PSSP-3.1.1  
2 PSSP-3.1.1  
7 PSSP-2.4  
8 PSSP-2.2
```

If the response indicates **PSSP-3.1.1**, this book applies to the version of PSSP that is running on your system.

If you are running mixed levels of PSSP, be sure to maintain and refer to the appropriate documentation for whatever versions of PSSP you are running.

Who Should Use This Book

This book is intended for system administrators responsible for setting up and maintaining the SP system. It assumes the administrators have a working knowledge of AIX or UNIX and experience with network systems.

The System Administrators Guide of USENIX (SAGE), has developed a classification for skills required for system administration as published in “Job Descriptions for System Administrators.” Administrators of small SP systems (one or two frames in a relatively simple environment) are expected to have level II skills (Junior System Administrator). Administrators of larger SP systems are expected to have level III skills (Intermediate/Advanced System Administrator).

Parts of this book can also be used by system operators or others that need to monitor the status of the hardware and software or interact with the power and reset controls.

Typographic Conventions

This book uses the following typographic conventions:

Typographic	Usage
Bold	Bold words or characters represent system elements that you must use literally, such as commands, flags, and path names.
<i>Italic</i>	<ul style="list-style-type: none">• <i>Italic</i> words or characters represent variable values that you must supply.• <i>Italics</i> are also used for book titles and for general emphasis in text.
Constant width	Examples and information that the system displays appear in constant width typeface.
[]	Brackets enclose optional items in format and syntax descriptions.
{ }	Braces enclose a list from which you must choose an item in format and syntax descriptions.
	A vertical bar separates items in a list of choices. (In other words, it means “or.”)
< >	Angle brackets (less-than and greater-than) enclose the name of a key on the keyboard.
...	An ellipsis indicates that you can repeat the preceding item one or more times.
<Ctrl-x>	The notation <Ctrl-x> indicates a control character sequence. For example, <Ctrl-c> means that you hold down the control key while pressing <c>.
\	The continuation character is used in coding examples in this book for formatting purposes.

Interface Instructions

Some sections of this book give step-by-step instructions for performing tasks with a graphical user interface. The instructions use a format that distinguishes between the user action and the system response.

User actions appear in uppercase bold type.

PRESS Cancel

Selections from a menu bar are indicated with an →.

SELECT SP → Topology

The system response to an action follows the • symbol. For example:

- The diagnostic display appears.

RS/6000 SP Overview

Chapter 1. RS/6000 SP Overview

This chapter provides an introduction to the IBM RS/6000 Scalable POWERparallel (SP) system, as well as a brief overview of the hardware and software that comprise the SP system.

Keep in mind that this is just a *brief* overview. For more information about the nodes, control workstation, and software, see the book *IBM RS/6000 SP: Planning Volume 2, Control Workstation and Software Environment*. For detailed hardware information see the book *IBM RS/6000 SP: Planning Volume 1, Hardware and Physical Environment*. Remember to reference the edition of the book that relates to the release of PSSP that you have or that you want to have.

Introduction

SP systems offer an information processing solution with the following features:

- Scalability

Scalable standard system configurations of one to 128 processor nodes are supported. You can expand your system over time to meet your changing computing requirements.

- Cost effectiveness

Because the SP system is based on AIX and RS/6000 technology, thousands of existing RS/6000 applications can be run on the SP system, eliminating the need for costly reprogramming.

The scalability of the SP system ensures that hardware and software invested in now will continue to serve you over time.

- Parallel computing

The SP system provides a strong platform for parallel application development and execution. System management applications provided by the IBM Parallel System Support Programs for AIX (PSSP) software, the SP Switch, and related IBM licensed program products (LPPs), such as IBM Parallel Environment for AIX, enhance the parallel computing capabilities of the SP system.

- Support of open system standards

The SP system accommodates a wide number and variety of I/O attachments and disk storage.

Enablers are provided that permit a number of commercial applications to exploit the parallel capabilities of the SP system.

- Flexibility

The IBM PSSP software allows you to run various combinations of serial, parallel, batch and interactive jobs concurrently.

- Simplified operation

The control workstation serves as a single point of control for installing, managing, and monitoring the SP system. It can also manage and monitor IBM Netfinity servers.

These features and capabilities make the SP an outstanding system for technical computing in a variety of market segments including Engineering Analysis, Chemistry, and Seismic/Petroleum. At the same time, the SP system is an excellent choice for the parallel commercial market, supporting database query, business management, decision support, online transaction processing, and batch business applications.

Hardware Overview

The basic hardware components of an SP system are:

- Processor nodes
- Frames
- Optional switch
- A control workstation
- Network connectivity adapters

These components connect to each other, comprising the SP system, by the SP administrative network (also known as the SP Ethernet network). They connect to your existing computer network through a Local Area Network (LAN), making the SP system accessible from any network-attached workstation.

Figure 1 illustrates a basic SP suitable for parallel and serial batch technical computing in a departmental setting.

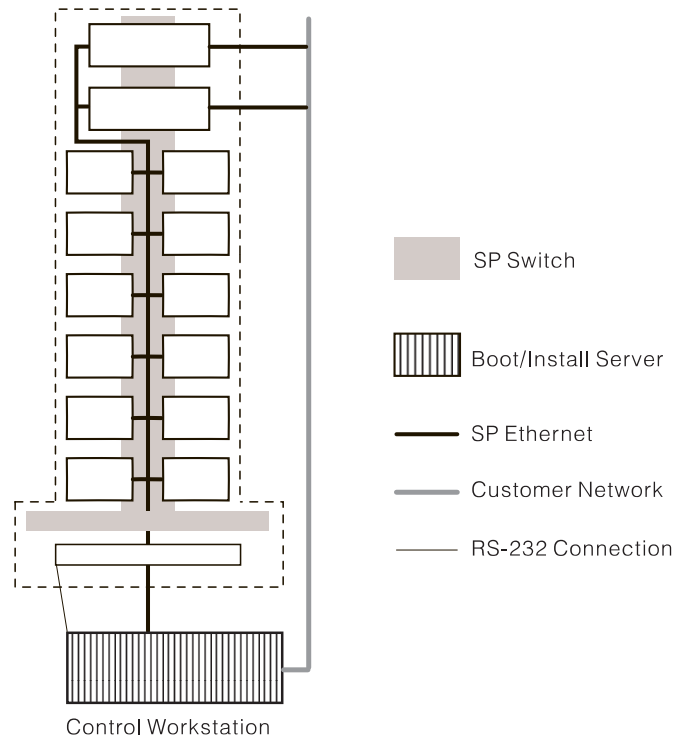


Figure 1. Basic SP Configuration

Processor Nodes

SP processor nodes have been available for mounting within short or tall SP frames. As of PSSP 3.1, *SP-attached servers* are also supported. SP-attached servers are other IBM RS/6000 servers that can be attached to the SP system and can logically function as SP processor nodes, but are not physically SP nodes mounted in an SP frame; for example, the RS/6000 Enterprise Server S70 or S70 Advanced.

SP processor nodes that get mounted in frames are available in three types: thin nodes, wide nodes, and high nodes. The frame spaces into which nodes fit are called drawers. A tall frame has eight drawers, while a short frame has four drawers. Each drawer is further divided into two slots. One slot can hold one thin node. A single thin node in a drawer, one that is not paired with another thin node in the same drawer, must occupy the odd numbered slot. A wide node occupies one drawer (two slots) and a high node occupies two drawers (four slots). The SP system is scalable from one to 128 processor nodes that can be contained in multiple SP frames. The maximum number of high nodes supported can range from 64 to 128 depending on which high nodes you use. Systems that can have from 129 to 512 nodes are available by special bid.

SP-attached servers do not mount in an SP frame. They connect to the SP directly by the SP Ethernet network and connect to the control workstation with two RS-232 cables. You can think of them as *self-framed*. There is limited hardware control and monitoring from the control workstation because they have no SP frame supervisor or SP node supervisor. Except for the physical differences, once installed, they function like any SP node running PSSP software and interacting with other nodes of the SP system. The number of SP-attached servers in an SP system is limited according to the restrictions for SP frames, since each one logically, though not physically, is managed by the PSSP components as though it occupies a separate SP frame. The maximum number of SP-attached servers supported in an SP system is sixteen.

Keep in mind

Unless otherwise explicitly stated, the information in this book about SP nodes in general applies to SP-attached servers as well.

An SP-controlled Netfinity server is not an SP-attached server. A Netfinity server is *SP-controlled* when it is connected to an SP control workstation that runs the SP/Netfinity Server Consolidation component of PSSP, resulting in a single point of control for both the SP and Netfinity systems. PSSP does not run on Netfinity servers nor do the processor nodes in this discussion interact with Netfinity servers in any way. See Chapter 21, "Managing an SP-Controlled Netfinity Server" on page 289 for more information.

SP processor nodes are uniprocessor or symmetric multiprocessor (SMP) systems available with varying levels of function, capacity, and performance. Each processor node includes memory, direct access storage devices (DASD), and a method for Ethernet connection. The variety of nodes and optional equipment can also lead to a variety of administrative concerns.

Extension Nodes

Extension nodes are non-standard nodes that extend the capabilities of the SP system, but cannot be used in all of 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, but much of the switch related protocol that standard nodes use is implemented on the SP Switch.

A physical dependent node, such as an SP Switch Router, can support multiple dependent node adapters. If a dependent node like an SP Switch Router contains more than one dependent node adapter, it can route data between SP system partitions. Data transmission is accomplished by linking the dependent node adapters in the SP Switch Router with valid switch ports on an SP Switch. If these SP Switches are located in different SP system partitions, data can be routed at high speed between the system partitions.

The SP Switch Router can be used to scale your SP system into larger systems through high speed external networks such as a FDDI backbone. It can also dramatically speed up TCP/IP, file transfers, remote procedure calls, and relational database functions.

Extension Nodes **require** a control workstation (CWS), primary node, and a primary backup node operating PSSP 2.3 or later.

Frames

SP frames have spaces into which the nodes fit. These spaces are called drawers. A tall frame has eight drawers, while a short frame has four drawers. Each drawer is further divided into two slots. One slot can hold one thin node or SP Expansion I/O Unit. A wide node occupies one drawer (two slots) and a high node occupies two drawers (four slots). An internal power system is included with each frame. Frames get equipped with the optional processor nodes and switches that you order.

SP processor nodes can be multiply mounted in a standard tall or short frame. SP-attached servers are conceptually *self-framed*. They do not sit in an SP frame, but they get connected to a tall SP frame. SP-attached servers are not supported with short frames.

The maximum number of frames supported on an SP system is 128. SP-attached servers apply 1-to-1 toward this count. The maximum number of SP nodes in a 128-frame system is 128. You can have up to 16 SP-attached servers. The maximum number of high nodes supported in a 128-frame system varies depending on which high nodes you have. You can have up to 64 of the older 604 High nodes or up to 128 of the POWER3 Symmetric MultiProcessor (SMP) High nodes.

SP Switch

Switches are used to connect nodes, providing the message passing network through which SP processor nodes communicate with a minimum of four disjoint paths between any pair of nodes. The switch currently supported is known as the SP Switch.

The SP Switch provides low latency, high-bandwidth communication between nodes. It consists of a switch assembly and the internal cables and ports to support

connection to eight or sixteen processor nodes in a system (one switch per frame). The SP Switch offers the following capabilities:

- Interframe connectivity and communication
- Scalability up to 256 node connections, including intermediate switch frames
- Constant bandwidth and latency between node pairs
- Support for Internet Protocol (IP) communication between nodes
- IP Address Resolution Protocol (ARP) support
- Support for dedicated or multi-user environments
- Error detection and retry
- High availability
- Fault isolation
- Concurrent maintenance for nodes
- Improved switch chip bandwidth

Adapters are required to connect each SP processor node, SP-attached server, and extension node to the SP Switch subsystem. See the book *IBM RS/6000 SP: Planning Volume 1, Hardware and Physical Environment* for which adapter is required per node or SP-attached server.

Note: The High Performance series of switches are not supported as of PSSP 3.1 and you cannot mix a High Performance switch with an SP Switch in an SP system, not even in separate system partitions. To use PSSP 3.1 or any nodes introduced since PSSP 2.4, you must upgrade all your switches to SP Switches.

Control Workstation

The SP system requires a customer-supplied IBM RS/6000 workstation with a color monitor. The control workstation serves as a central point of control with the PSSP and other optional software for managing and maintaining the SP processor nodes and related hardware and software. The control workstation connects to each frame with an RS-232 cable to provide hardware control functions. It can also connect to an SP-attached server with two RS-232 cables, but the hardware control is minimal because these servers do not have an SP frame or SP node supervisor. A system administrator can log in to the control workstation from any other workstation on the network to perform system management, monitoring, and control tasks.

The control workstation also acts as a boot/install server for other servers or nodes in the SP system. If you have a Netfinity server running Windows NT, you can make it *SP-controlled* when you connect it to the control workstation and use the SP/Netfinity Server Consolidation component of PSSP, resulting in a single point of control for both the SP and Netfinity systems. See Chapter 21, "Managing an SP-Controlled Netfinity Server" on page 289 for more information.

In addition, the control workstation can be set up as an authentication server. For example, it can be the Kerberos V4 primary server, with the master database and administration service as well as the ticket-granting service. Or it can be set up as a Kerberos V4 secondary server, with a backup database and just the ticket-granting service.

The High Availability Control Workstation option enables you to have two control workstations, one as primary and the other as secondary, for automatic failover and reintegration to a backup control workstation in the event that the primary control workstation becomes unavailable.

Network Connectivity and I/O Adapters

Network connectivity for the SP system is supplied by various adapters, some built in, some optional, that can provide connection to I/O devices, networks of workstations, and mainframe networks. Ethernet, FDDI, token-ring, HIPPI, SCSI, FCS, and ATM are some types of adapters that can be used as part of an SP system.

The SP Ethernet is the network that connects all nodes to each other and to the control workstation in the SP system. A 15 meter (50 feet) Ethernet cable is provided with each frame to use in the wiring of this network. Additional optional adapters such as Ethernet, FDDI, and token-ring are automatically configured on each node. Other optional supported adapters can be individually configured on each node.

SP Expansion I/O Unit

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. These hard disks are considered internal DASD of the associated node.

The SP Expansion I/O Unit has the following characteristics and restrictions:

- It is available only for connecting to a POWER3 SMP **high** node.
- It connects to a POWER3 SMP high node directly by cable.
- Up to six SP Expansion I/O Units can connect to one POWER3 SMP high node.
- Each unit is about the size of a thin node and occupies a thin node slot in a tall SP frame (it is not supported in a short frame).
- It can be in any tall frame in the SP system, not necessarily in the same frame as the node to which it connects.
- It is irrelevant to SP system partitioning. Only the node to which it is connected is considered to be in a system partition.
- It does not support connection to an SP Switch. The node to which it is connected can be connected to an SP Switch.
- It does not support twin-tailed internal I/O.

Software Overview

The SP system software infrastructure includes:

- AIX, the base operating system
- PSSP, the higher-level set of support programs and interfaces that enables you to take advantage of the powerful parallel processing features of the SP system
- Other IBM system and application software products
- Independent software vendor products

AIX

AIX provides operating system functions such as the AIXwindows user interface, extended real-time support, network installation management, advanced file system support, physical disk space management, and a platform for application development and execution. AIX provides UNIX functionality and conformance to industry standards for open systems. AIX capabilities that can be exploited for system management on an SP system include the following:

- The Network Installation Management (NIM) environment provides the ability to install an AIX **mksysb** image over the network.
- The Logical Volume Manager (LVM) improves data management productivity, enables files to span multiple disk drives, and provides, through disk mirroring, a high availability option for critical data.
- The System Management Interface Tool (SMIT) provides a single, consistent, and expandable interface to system management commands.
- The System Resource Controller (SRC) simplifies the management of subsystems and daemons.
- Security features include a shadow password file and comprehensive auditing facilities.
- AIX device support lets you add and delete devices at any time without disrupting the system.

PSSP

PSSP is a comprehensive suite of applications to manage an SP system as a full-function parallel processing system. It provides a single point of control for administrative tasks and helps increase productivity by letting administrators view, monitor, and control how the system operates. Most functions are base components of PSSP while others are optional; they come with PSSP, but you choose whether to install and use them.

Highlights

The PSSP software provides a comprehensive suite of applications for installing, operating, managing, and administering the SP system. Most are base components of PSSP while others are optional, which come with PSSP and you choose whether to install and use them.

Notes:

1. The SP Print Management System has been removed from PSSP. That is, the SP Print Management System cannot be configured on nodes running PSSP 2.3 or later. IBM suggests you use the Printing Systems Manager (PSM) for AIX as a more general solution to managing printing on the SP system. However, if you are running earlier versions of PSSP on some of your nodes, the SP Print Management System is still supported on those nodes. The **print_config** routine running on the control workstation will configure the SP Print Management System on nodes running versions of PSSP earlier than PSSP 2.3.
2. There is some information on tuning, such as setting initial tuning parameters or reviewing sample tuning files, in the book *PSSP: Installation and Migration Guide*. Tuning information that used to be in this book (PSSP Administration Guide) is now available on the Web site <http://www.rs6000.ibm.com/support/sp/>

3. If you are running mixed levels of PSSP in a system partition, be sure to keep and refer to the respective documentation for each version of PSSP you are running.

Resource Center: The SP Resource Center provides one simple interface for all softcopy SP documentation and information resources. It consists of HTML, Java, and Javascript files and works with a Web browser. The Resource Center provides access to a variety of information including publications, READMEs, Redbooks, White Papers, product information, as well as up-to-date service information.

Installation: The Network Installation Management (NIM) environment supports the installation of AIX 4.2.1 and later versions on nodes.

Configuration Management: Node configuration data is entered into the System Data Repository (SDR) using SMIT or line commands. The SDR provides storage and retrieval of system data across the control workstation, file servers, and nodes.

Operation: The following are available to help you operate and manage your SP system:

- **SP TaskGuides**

SP TaskGuides are a form of advanced online assistance designed to walk you through complex or infrequently performed tasks. Each TaskGuide does not simply list the required steps. It actually performs the steps for you, automating the steps to the highest degree possible and prompting you for input only when absolutely necessary. You might recognize them as *wizards*.

The following TaskGuides are available:

- Set Site Environment Information
- Add Frames
- Configure New Nodes
- Create Node Image

- **SP Perspectives**

This is a set of graphical user interfaces with which you can monitor and control hardware, create and monitor system events, manage IBM virtual shared disks, generate and save system partition configurations, and set up performance monitoring hierarchies and archives.

- **Centralized Management Interface**

This provides a SMIT-based interface for system management commands, as well as command line equivalents.

- Tools are provided for orderly shutdown and restart of the system.

File Management: Groups of files can be defined as a file collection and any changes to files in that collection are propagated to the appropriate nodes.

Virtual Shared Disk Management: The PSSP components that help you create and manage virtual shared disks are:

- IBM Virtual Shared Disk, the component with which you create and manage virtual shared disks. It has a device driver that operates with the AIX Logical Volume Manager for your applications that use the virtual shared disks.

- Recoverable Virtual Shared Disk, the component that provides recoverability of your virtual shared disks when a node, adapter, or disk failure occurs.
- Hashed Shared Disk, the component that works with the IBM Virtual Shared Disk component to offer data striping for your virtual shared disks.

These components are documented in the book *PSSP: Managing Shared Disks*.

User Management:

- Commands are supplied to add, change, and delete users and passwords.
- Users have the same account, home directory, and environment across all nodes in the system.
- The **user.admin** file collection is provided for propagating user administration files to the nodes on the system.

Job Management: As of PSSP 3.1, the job management function has been integrated into the IBM LoadLeveler product.

Accounting: Accounting support provides accounting record consolidation and parallel job accounting.

Change Management:

- System partitions can be used to apply and test new levels of software.
- Parallel management commands can be used to apply maintenance directly to nodes.
- File collections can be used to propagate changed files to nodes that have the file collection installed.
- For extensive service a new **mksysb** can be created and the nodes reinstalled using NIM.

Security: Authentication services, based on Kerberos V4, are provided for:

- The Sysctl remote command execution facility
- The **rsh** and **rcp** commands
- The system monitor

Communication Subsystem Support: The Communication Subsystem supports the reliability and performance of the SP Switch.

Availability and Problem Management:

- Group Services provide a distributed coordination and synchronization service. For more information, refer to Chapter 24, “The Group Services Subsystem” on page 325.
- Event Management matches information about the state of system resources with information about resource conditions that are of interest to client programs. For more information, refer to Chapter 25, “The Event Management Subsystem” on page 337.
- Problem Management provides for pairing actions with system events. For more information, refer to Chapter 26, “Using the Problem Management Subsystem” on page 361.

- A single point of control for configuring, archiving, and maintaining various system logs on individual nodes is provided.
- Additional logs may be specified for use by applications.

SP/Netfinity Server Consolidation: Enables control of a Netfinity server from the control workstation of the SP system, a single point of control for both systems.

Packaging

The PSSP software is packaged as several **installp** images. Each image contains one or more filesets. For complete information on these images and the filesets that each image contains, see the book *PSSP: Installation and Migration Guide* or the *File Set List for PSSP* table in the book *IBM RS/6000 SP: Planning Volume 2, Control Workstation and Software Environment*.

Other IBM Products

A large number of system and application software products developed by IBM can be run on the SP system. For example:

- IBM LoadLeveler
Distributed, network-wide job management with parallel batch jobs a specialty
- IBM Parallel Environment for AIX
Parallel application development and execution
Message passing parallel task communications
- IBM Engineering and Scientific Subroutine Library for AIX (ESSL)
Library of subroutines for use by applications
- IBM Parallel Engineering and Scientific Subroutine Library for AIX (Parallel ESSL)
Set of advanced parallel subroutines which extend the ESSL product
- IBM High Availability Cluster Multi-Processing for AIX (HACMP)
Ensures that critical resources are available for processing
- IBM Network Tape Access and Control System for AIX (NetTAPE) and IBM NetTAPE Tape Library Connection for AIX
Simplifies tape access and operations management in SP systems
- IBM Client Input Output/Sockets for AIX (CLIO/S)
High speed, transparent data transfer and tape access between MVS/ESA systems and AIX systems or between AIX systems
- IBM General Parallel File System for AIX
Concurrent shared access to files spanning multiple disks located on multiple nodes
- ADSM/6000
System and user data management
- PTX/6000
Performance monitoring
- NetView for AIX

Enterprise network management

- Trouble Ticket for AIX

Problem management

- PSF/6000

Printing support

- Job Scheduler for AIX

Schedules production batch workload in a distributed AIX environment

Independent Software Vendor Products

There is an aggressive program in place to enable and encourage independent software vendors to port their applications to the SP system.

Routine System Administration Tasks

Chapter 2. Starting Up and Shutting Down the SP System

This chapter describes how to start up and shut down nodes on the SP system. It explains how to specify the sequence in which the nodes and subsystems in your system are started up and shut down. It also describes the tools that save you time and effort by operating on all or some of the nodes in your SP system with a single command. These tools include the **cstartup**, **cshutdown**, and **seqfile** commands, plus the files they use to control the sequence of startup or shutdown processing.

This chapter describes the interface that lets you coordinate the shutdown of a subsystem that operates across multiple nodes.

Overview of System Startup and Shutdown

You control the operation of the SP system with commands issued at the control workstation.

Use the **cstartup** command to power on SP nodes and boot up the operating system. You can also reset SP nodes with the **cstartup** command.

Use the **cshutdown** command to halt or reboot SP nodes.

Your SP might be a complex group of nodes, each dependent on other nodes for some resources. Startup and shutdown operations must occur in the proper sequence so that these dependencies are honored. As a simple example, a server node might have to be running before its client nodes are started up. A server node should not shut down before its clients have been shut down. The **seqfile** command defines sequencing relationships that result from using boot/install servers. You can define additional dependencies.

To provide best performance, **cstartup** and **cshutdown** can operate on multiple nodes concurrently.

Note: Given appropriate time-out values, the **cstartup** and **cshutdown** commands, properly handle any SP Expansion I/O Units that are connected to POWER3 SMP high nodes.

Similar considerations apply to the shutdown of *special subsystems* (see “Creating a Special Subsystem” on page 28) that run on multiple nodes in the SP system. An example of a subsystem is a file system that spans several nodes within a system partition. Subsystems might need to do special processing to prepare for the shutdown of one or more nodes, or a set of subsystems might need to shut down in a specific order.

Controlling the Sequence of Startup and Shutdown Operations

You must define the sequencing relationships between your nodes. For example, boot/install servers must start up before their clients. Other programs on your SP system might create additional sequencing relationships.

The node sequence files, **/etc/cshutSeq** and **/etc/cstartSeq**, have lines that describe these dependencies. In these files, you define the groups of nodes and the sequence in which the groups are started up or shut down. If you don't have a

node sequence file, some sequencing is performed automatically. The **seqfile** command helps you create these files by generating definitions for groups of nodes and their sequencing order for dependencies relating to boot/install servers and their clients.

You can also create a subsystem sequence file, **/etc/subsysSeq**, which lists the subsystems, defines groups of subsystems, and defines relationships between subsystems. If you don't have a subsystem sequence file, no subsystems are notified of an impending node shutdown.

The sequence files, which reside on the control workstation, are:

/etc/cstartSeq	Startup. The list defining groups of nodes and the order of startup processing.
/etc/cshutSeq	Shutdown. The list defining groups of nodes and the order of shutdown processing.
/etc/subsysSeq	Special subsystems. The list defining special subsystems, groups of special subsystems, and the order of shutdown processing.

The node sequence files are separate to allow for different startup and shutdown dependencies. However, if your startup and shutdown dependencies are mirror images, you can maintain a single file; create a symbolic link to provide both file names.

"Defining Startup and Shutdown Sequence Files" on page 25 describes the format of the sequence files and their interactions with command line flags.

Default Node Sequencing

The SP system can do some sequencing automatically during system startup or shutdown. The **seqfile** command shows you the default sequencing relationships.

If you don't create the **/etc/cstartSeq** file, the **cstartup** command uses a default sequence based on the SDR. Any nodes defined in the SDR as boot/install servers are started up concurrently. After the servers are running, nodes defined as boot/install clients are started up. If you create an empty **/etc/cstartSeq** file, then all nodes are started up concurrently.

If you don't create an **/etc/cshutSeq** file, **cshutdown** uses a default sequence based on the SDR. If you create an empty **/etc/cshutSeq** file, then all nodes are shut down concurrently. If you don't have a **/etc/subsysSeq** file or the file is empty, no special subsystem shutdown occurs.

Starting the SP System

To start the SP system, log in to the control workstation and issue the **kinit** command, specifying a principal name for which there is an entry in the **hardmon** access control list file (acl or acs is typically part of the file name) with control authorization for the frames to be started up. Then, use the **cstartup** command.

Note for High Nodes and SP-attached Servers

High nodes and SP-attached servers (like models S70 and S7A) take a considerably longer time to power on than thin and wide nodes. It can take a high node 10 to 20 minutes to power on, depending on the number of processors in the node and the number of adapters, while a fully configured SP-attached server might take 40 minutes to power on. A fully configured POWER3 SMP high node can take even longer. After you have experienced the amount of time your nodes take, you can better estimate the amount of time to specify with the `-W` flag for the **cstartup** and **cshutdown** commands.

There might be times when you need to control the startup and shutdown of these nodes manually, rather than with the **cstartup** and **cshutdown** commands. If you do, be very careful of the sequencing that is otherwise automatically handled.

For a POWER3 SMP high node with SP Expansion I/O Units in particular, the following additional considerations apply:

- Be sure the access control list file gives you authorization to each frame that contains SP Expansion I/O Units connected to the nodes for which you have authority.
- If you do not use the **cstartup** command or other equivalent PSSP function, first power on the SP Expansion I/O Units, then the node to which they are connected, then reboot or power on and startup the node.
- If you do not use the **cshutdown** command or other equivalent PSSP function, first shutdown the nodes, then power off the SP Expansion I/O Units that are connected to them.

The **cstartup** command powers on specified nodes that are not powered on, and might reset specified nodes that are already powered on. Those actions are equivalent to manually pressing the hardware switches. Once the hardware-equivalent actions are done, control is passed to **boot** and **init**.

Caution!

The **cstartup** command attempts to power on nodes that are powered off. This has safety implications if someone is working on the nodes. Take proper precautions when using this command.

To restart an SP system that is already running, use **cshutdown -r -G**.

Note: If some nodes cannot be started in a specific sequence, or if some cannot be started, the system might still be usable.

The complete syntax of the **cstartup** command is described in the *PSSP: Command and Technical Reference*.

Starting or Resetting All Nodes

You can start or reset nodes with a single **cstartup** command, as shown in Table 1.

Command	Meaning
cstartup ALL	Starts all nodes in the current system partition.
cstartup target_nodes	Starts specific nodes in the current system partition. All nodes listed must be in the current system partition or the command will fail.
cstartup -G ALL	Starts all nodes in the physical SP system.
cstartup -G target_nodes	Starts specific nodes in the physical SP system, regardless of system partitioning.

If a node is powered off, the **cstartup** command powers it on. If the node is already powered on and is not running, this command resets it. The nodes are started in the sequence specified by */etc/cstartSeq*. If that file doesn't exist, the nodes are started up in the default sequence.

To start all nodes in the SP system concurrently, ignoring the sequence file if it exists or bypassing the default sequencing if the file does not exist, use the **-E** flag, as in the following example:

```
cstartup -E -G ALL
```

To create the */etc/cstartSeq* file, see “Defining Startup and Shutdown Sequence Files” on page 25.

Starting or Resetting Selected Nodes

To start or reset a specific node within the current system partition, put its name on the **cstartup** command line. The following example starts the single node in the current system partition named **cs6**:

```
cstartup cs6
```

The command checks for dependencies before starting up the node. It uses the */etc/cstartSeq* file, if one exists, or the default sequence rules.

If node **cs6** is already running, you must specify either **-z** or **-Z** to force the node to be reset. This example resets the node **cs6** in the current system partition without checking any sequencing dependencies that **cs6** has with other nodes:

```
cstartup -Z cs6
```

You can also identify nodes by node number or range of node numbers using the **-N** flag. The following command starts node numbers 4 through 12 in the current system partition.

```
cstartup -N 4-12
```

Shutting Down the SP System

To shut down the SP system, log in to the control workstation and issue the **kinit** command, specifying a principal name for which there is an entry in the **hardmon** access control list file with control authorization for the frames to be shutdown. Then, use the **cshutdown** command. The SP **cshutdown** command is similar to the workstation **shutdown** command, except it has these advantages over using **shutdown** to shut down each node of an SP:

- **cshutdown** provides a single point of control.

Using one **cshutdown** command on the control workstation, you can shut down all or selected nodes.

- The sequencing of node shutdown and reboot is automated.

You can use the **/etc/cshutSeq** file to control the order in which nodes are shut down, or you can let the system determine the order based on System Data Repository information about servers and clients.

- Special user-defined subsystems can be notified of impending node shutdown.

The **/etc/subsysSeq** file lists these special subsystems and describes any sequencing relationships between them.

Note: The **cshutdown** command powers off the node after the **shutdown** process completes.

You can shutdown nodes with a single **cshutdown** command as shown in Table 2.

Command	Meaning
cshutdown ALL	Shuts down all nodes in the current system partition.
cshutdown <i>target_nodes</i>	Shuts down specific nodes in the current system partition. All nodes listed must be in the current system partition or the command will fail.
cshutdown -G ALL	Shuts down all nodes in the physical SP system.
cshutdown -G <i>target_nodes</i>	Shuts down specific nodes in the physical SP system, regardless of system partitioning.

The complete syntax of the **cshutdown** command is described in the *PSSP: Command and Technical Reference*. Understanding how shutdown processing works might help you understand some of the **cshutdown** command's options. See "The Phases of Shutdown Processing" on page 23.

Halting Nodes

You can halt all nodes in the current system partition with one command, using:

```
cshutdown ALL
```

The **-h** (halt) flag is the default; you don't need to use it when you invoke **cshutdown**. The nodes are shut down in the sequence specified by the **/etc/cshutSeq** file. If this file doesn't exist, the nodes are shut down in a default sequence based on the SDR. If the **/etc/subsysSeq** exists, those subsystem interfaces are invoked during shutdown.

Halting selected nodes is similar. To halt nodes **cs3**, **cs5**, and **cs7** in the current system partition, use the command:

```
cshutdown cs3 cs5 cs7
```

The command checks for dependencies before shutting down the nodes. If the **/etc/cshutSeq** file exists, those rules apply; otherwise, **cshutdown** applies default sequencing rules based on the SDR. If the **/etc/subsysSeq** file exists, those subsystem interfaces are notified of the shutdown of the target nodes.

The subsystem shutdown phase tells special subsystems to terminate their own operations on the target nodes before the node phase shuts down the target nodes. Groups for which the **/etc/subsysSeq** file does not define a sequencing relationship are invoked concurrently. Subsystems not listed in the **/etc/subsysSeq** file are not notified of the pending system shutdown.

Rebooting Nodes

Reboot all nodes in the current system partition by using the **cshutdown** command with the **-r** (reboot) flag:

```
cshutdown -r ALL
```

The target nodes are halted according to the sequencing rules in **/etc/cshutSeq**, and restarted according to the rules in **/etc/cstartSeq**. If these files don't exist, the default sequencing rules are determined at shutdown and startup.

You can control how the system starts during reboot by passing flags to the **cstartup** processor using the **cshutdown** command's **-C** flag. For example, to pass the **-E** flag to tell the **cstartup** processor to ignore the sequence file when rebooting current system partition nodes numbered 4 and 5, use the command:

```
cshutdown -r -C "-E" -N 4-5
```

To pass multiple startup flags, you must escape any space characters in the string. The double quotes in this example ("**-E**") do that, although they are not strictly required with a single flag.

Rebooting from Maintenance Mode

If you bring a node down to maintenance mode, you must ensure file system integrity before you reboot the node.

The **cshutdown** command, which runs from the control workstation, will be unable to **rsh** to the node to perform the node shutdown phase processing which includes syncing of the file systems. IBM suggests that you issue the **sync** command three times in succession from the node console before running **cshutdown**. This is especially important if any files were created while the node was in maintenance mode.

To determine which nodes are in maintenance mode, run the **spmon -d** command and look for the combination of **nodePower** on and **hostResponds** no.

Notifying Users of an Impending Shutdown

You probably want to warn your users if nodes are going down, so they can save and stop any work in progress. You can also have a customized *shutdown clean* script to gracefully terminate nonroot user processes. You can create a shell script, on each node where you want it to run, named `/etc/cshut.clean`. Make it an executable that performs the function you want to be processed before the `cshutdown` command terminates nonroot processes.

By default, when you issue a `cshutdown` command, a message is sent to all users who are logged in to the affected nodes, telling them how long they have before shutdown. You can customize the message as well as the time when the system shuts down.

To power off the entire SP system at 9 pm, for instance, issue the following command:

```
cshutdown -T 21:00 -M "All nodes will power off at 9 pm." -G ALL
```

To power off all nodes in the current system partition at 9 pm, issue the following command:

```
cshutdown -T 21:00 -M "All nodes will power off at 9 pm." ALL
```

Suppose you need to do an emergency reboot of the pair of nodes named `cs1` and `cs2` in the current system partition, but want to allow 10 minutes for jobs to complete. You can notify your users and schedule the reboot with this command:

```
cshutdown -r -T 10 -M \  
"Reboot on cs1&2 in 10 minutes--don't submit new jobs." cs1 cs2
```

Several processing phases are involved in the `cshutdown` command.

The Phases of Shutdown Processing

`cshutdown` command processing has these phases:

1. Notifying users and terminating nonroot processes
2. Subsystem shutdown
3. Node shutdown
4. Restarting nodes, if requested

Notifying Users and Terminating Nonroot Processes

The message specified with the `-M` option of the command is sent to all users who are logged in to the target nodes. If a customized shutdown clean script exists on a node (file `/etc/cshut.clean`), it is run. Then user processes that were not started by root and are running on the target nodes are sent a `SIGTERM` followed, 30 seconds later, by a `SIGKILL`. The 30 second delay gives user processes that handle `SIGTERM` a chance to do whatever cleanup is necessary.

Subsystem Shutdown

The subsystem shutdown phase tells special subsystems to terminate their own operations on the target nodes before the node phase shuts down the target nodes. Groups for which `/etc/subsysSeq` doesn't define a sequencing relationship are invoked concurrently. Subsystems not listed in `/etc/subsysSeq` are not notified of the pending system shutdown.

If **cshutdown** is issued without the **-Y** flag and a nonzero (failure) code is returned, you receive a prompt allowing you to continue, to quit, or to start a subshell so that you can check `/var/adm/SPIlogs/cs/cshut.MMDDhhmmss.pid` for messages and refer to the documentation for the subsystem. When you leave the subshell, you are prompted with the same choices.

The subsystem phase of shutdown waits for all subsystems to return before continuing with the node phase. If a subsystem does not return, the **cshutdown** command waits indefinitely. Use the **-W** flag to specify a time-out value.

Node Shutdown

The node phase operates on nodes specified with the **cshutdown** command. The `/etc/cshutSeq` file controls the sequencing of the node phase operation.

Node shutdown has potentially two levels of coordinated concurrent operations:

- More than one group of nodes can be shut down concurrently.
- Within each one of those groups, more than one node can be shut down concurrently.

They are coordinated such that you can specify that one group must wait for the completion of some other specific groups before starting to shut down.

In a halt operation, the node phase applies the standard workstation shutdown on the target nodes. The nodes are halted in the sequence specified in the `/etc/cshutSeq` file, if it exists, or in the default sequence. Nodes not listed in the `/etc/cshutSeq` file might be halted concurrently with any other nodes.

Restarting the Nodes, If Requested

If you request a system reboot with the **-r** flag, the node phase halts the nodes in the sequence specified in `/etc/cshutSeq` and then restarts the nodes consistent with the initialization sequence specified in `/etc/cstartSeq`. If these files do not exist, the default sequence is used. It does not power on any nodes that were powered off when the **cshutdown** command was issued. If you use the **-r** flag, the only nodes powered on during reboot are those powered off during shutdown.

Checking the Status of Startup and Shutdown Requests

Each time you invoke **cstartup** or **cshutdown**, the system creates a log of activities related to that command. The logs contain an entry for the initiation and completion of operations on each node. The log for **cshutdown** includes information about subsystem shutdown operations. These logs are in the directory `/var/adm/SPIlogs/cs`. Two types of log files are created:

cstartup `/var/adm/SPIlogs/cs/cstart.MMDDhhmmss.pid`

cshutdown `/var/adm/SPIlogs/cs/cshut.MMDDhhmmss.pid`

The name of each log file contains the date and time that the command was issued, in the format `MMDDhhmmss.pid`, where:

MMDDhhmmss The timestamp

pid The process ID of the **cstartup** or **cshutdown** command

Defining Startup and Shutdown Sequence Files

Relationships between nodes must be defined in the `/etc/cstartSeq` and `/etc/cshutSeq` files. You define groups of nodes in these files. Nodes in the same group are started up or shut down concurrently. You also define the order in which groups of nodes are processed. Sequencing files can contain nodes from any and all system partitions; sequencing rules are applied without regard to system partitions. Groups without specified sequencing relationships are started up concurrently. Nodes not listed in `/etc/cstartSeq` might be started up concurrently with any other nodes. Nodes not listed in `/etc/cshutSeq` might be shut down concurrently with any other nodes.

The sequencing of special subsystems is handled similarly. Subsystems and groups of subsystems are defined in the `/etc/subsysSeq` file. Subsystems without specified sequencing relationships are shut down concurrently. Subsystems must be defined in this file. If they are not defined, they are not invoked during shutdown.

Planning for Concurrent Startup and Shutdown

It's important that you allow the `cstartup` and `cshutdown` commands to do as much work as possible concurrently. This ensures optimal performance when starting up and shutting down a system with many nodes and/or subsystems. There are potentially two levels of coordinated concurrent operations:

- More than one group of nodes or subsystems can be started up or shut down concurrently.
- Within each one of those groups, more than one node or subsystem can be started up or shut down concurrently.

They are coordinated such that you can specify that one group must wait for the completion of some other specific groups before being started up or shut down.

Format Rules for Sequence Files

A line in `/etc/cstartSeq`, `/etc/cshutSeq`, and `/etc/subsysSeq` can be one of the following:

- Comment line starting with a pound sign (#). A comment can also start within a line.
- Group definition line containing a group name followed by a colon (:), followed by one or more node names (`/etc/cstartSeq`, `/etc/cshutSeq`) or subsystem names (`/etc/subsysSeq`). To put multiple nodes or subsystems in a group, separate their names with commas (,) and, optionally, blanks.
 - In `/etc/cstartSeq` and `/etc/cshutSeq`, a node name is the en0 Ethernet hostname of the node. These are also known as **reliable_hostname** in the **Node** object class in the System Data Repository.
 - In `/etc/subsysSeq`, a group definition line is the arbitrary subsystem group name, followed by a colon (:), followed by the full path name of the command you want issued at shutdown.

The group names in the file are arbitrary names and treated as tokens.

Each node or subsystem can be in only one group. A group can consist of from 1 to all your nodes or subsystems.

- Sequence line containing a greater than symbol (>). For instance, in the **/etc/cstartSeq** file, groupA > groupB says to start groupA before starting groupB. And, in the **/etc/cshutSeq**file, groupA > groupB says to shut down groupB before shutting down groupA.

A group cannot both precede and trail the same group. That is, the sequence A > B > A is not valid.

The lines can be in any order. Each node must be assigned to a group if you wish to control its sequencing.

A group or node name cannot contain a blank, comma, newline, tab character, greater than symbol (>), colon (:), parentheses (), or pound sign (#). Case is significant in node and group names.

Testing Your Sequence Files

You can verify that the **/etc/cstartSeq** file has the correct format without powering on or resetting any nodes by specifying:

```
cstartup -k ALL
```

The **-k** flag tells **cstartup** not to reset any nodes, just check the sequence file. If the sequence file is correct, you'll get the following message:

```
cstartup: No circular dependencies were detected
```

Verify that the **/etc/cshutSeq** and **/etc/subsysSeq** files have the correct format without shutting down any nodes by specifying:

```
cshutdown -kF ALL
```

The **-k** flag tells **cshutdown** not to shutdown any nodes, just check the sequence file. The **-F** flag tells **cshutdown** not to issue any messages to users.

Getting Started: Capturing the Default Sequences

An easy way to create your **/etc/cstartSeq** and **/etc/cshutSeq** files is to capture the default group definitions and sequencing relationships that are based on the boot/install server information in the System Data Repository. You can then add to or modify them as required for your installation.

To create the default **/etc/cstartSeq** file, use the command:

```
seqfile -b > /etc/cstartSeq
```

The **-b** flag tells **seqfile** to include boot/install relationships, which are important only during system startup.

To create the default **/etc/cshutSeq** file, use the command:

```
seqfile > /etc/cshutSeq
```

After you've created the default files, you can edit them to reflect sequencing relationships based on other aspects of your installation's configuration.

Sample Startup File `/etc/cstartSeq`

Here is a simple file for sequencing startup:

```
ServerA > Clients0fA
ServerA: node2
Clients0fA: node3, node4, node5, node6
```

The line `ServerA: node2` defines node 2 as the group `ServerA`. The line `Clients0fA: node3, node4, node5, node6` defines nodes 3 through 6 as a group that can be started up concurrently. The sequence line `ServerA > Clients0fA` says that `ServerA` must be completely started up before starting up the nodes that are clients of server A (nodes 3 through 6).

This sample file is slightly more complex:

```
groupA > groupB
groupA > (groupF, groupE) > groupD > groupC
groupD > groupG
groupA: node0 #node0 is the node that node1-node7 depend on.
groupB: node1
groupC: node2
groupD: node4
groupE: node5
groupF: node6, node7
groupG: node3
```

The line `groupA > (groupF, groupE) > groupD > groupC` says to start up `groupA`. When `groupA` is up, concurrently start up `groupF` and `groupE`. After they are up, start up `groupD`. Finally, when `groupD` is up, start up `groupC`. The line `groupA > groupB` says that `groupB` depends on `groupA`, but can be started concurrently with `groupE` and `groupF`. No other groups have dependencies on `groupB`.

If your command line specifies a node but omits another that is listed in your `/etc/cstartSeq` or `/etc/cshutSeq` files as its leading node, the sequencing rules still apply unless overridden by the `-X` flag.

Sample Shutdown File `/etc/cshutSeq`

Here is one of the sample startup sequence files, interpreted now as a shutdown file:

```
ServerA > Clients0fA
ServerA: node2
Clients0fA: node3, node4, node5, node6
```

The line `Clients0fA: node3, node4, node5, node6` defines nodes 3 through 6 as a group that can be shut down concurrently. The line `ServerA: node2` defines node2 as the group (of one node) called `ServerA`. The line `ServerA > Clients0fA` says that the nodes that are clients of server A (nodes 3 through 6) can be shut down concurrently, but must be shut down before node2 in group `ServerA`.

Sample Subsystem File /etc/subsysSeq

A sample file for subsystem shutdown could be:

```
ssgroupA: /full/pathname/subsystemA
ssgroupB: /local/subsystemB, /local/subsystemC
ssgroupA > ssgroupB
```

The line `ssgroupA: /full/pathname/subsystemA` defines the group `ssgroupA`, consisting of the single subsystem invoked by the `/full/pathname/subsystemA` command. The line `ssgroupB: /local/subsystemB, /local/subsystemC` defines a group consisting of the two subsystems `/local/subsystemB` and `/local/subsystemC`.

The line `ssgroupA > ssgroupB` says to first notify `ssgroupB` before nodes are shut down. When both subsystems in `ssgroupB` have returned, then notify the single subsystem in `ssgroupA`.

Creating a Special Subsystem

Read this section if you write your own special subsystem and want the **cshutdow**n command to notify the subsystem when nodes are being halted or rebooted. The SP system has no predefined special subsystems.

The special subsystem interface is a user-provided command that is invoked during the subsystem phase of SP system shutdown processing. The command is defined in the **/etc/subsysSeq** file and must reside on all nodes of the SP system; the **cshutdow**n command invokes the subsystem interface on a randomly selected node in the current system partition.

The following sections define the interface to a special subsystem. This interface consists of:

- The inputs that the **cshutdow**n command passes to your special subsystem
- The actions your subsystem must perform.

Note: The special subsystem interface is called *after* nonroot processes are killed. If you want your subsystem to take advantage of shutdown notification, some part of it must be running as root.

Input to the Special Subsystem Interface

The **cshutdow**n command invokes the special subsystem interface with the following standard parameters:

-r or -h	Indicates whether the cshutdow n command is halting (-h) or restarting (-r) the target nodes.
-b <i>IPaddress</i>	Specifies, in dotted decimal form, the IP address of the calling program, in this case, the control workstation.
-f <i>/var/adm/SPIlogs/cs/cshut.MMDDhhmmss.pid</i>	Specifies the log file to which the subsystem interface program should append a record of status and activities.
<i>target_nodes</i>	The list of node numbers of the target nodes specified on the cshutdow n command line. The numbers are passed at the end of the argument list.

For example, if the **/etc/subsysSeq** file contains the following line:

```
ssgroupA: /full/pathname/subsystemA
```

and this **cshutdown** command is issued on the control workstation whose IP address is 129.40.64.70:

```
cshutdown -h -N 1 2 3
```

the **cshutdown** command issues the following command on a randomly selected node during the subsystem phase of shutdown processing:

```
/full/pathname/subsystemA -h -b 129.40.64.70 \  
-f /var/adm/SPlogs/cs/cshut.time.pid \  
1 2 3
```

What Your Special Subsystem Must Do

In addition to any processing specific to the subsystem, the special subsystem must call the **csLogger** command to record its activities. Your program must call **csLogger** with the following required parameters:

- b *IPaddress*** Specifies, in dotted decimal form, the IP address of the calling program, in this case, the control workstation.
- f */var/adm/SPlogs/cs/cshut.MMDDhhmmss.pid*** Specifies the log file to which the subsystem interface program should append a log of status and activities.
- logString*** Specifies a string that will be appended to the specified log file. This is the last parameter.

Using the example previously started, if the **cshutdown** command calls your program with this command:

```
/full/pathname/subsystemA -h -b 129.40.64.70 \  
-f /var/adm/SPlogs/cs/cshut.time.pid \  
1 2 3
```

then the **/full/pathname/subsystemA** command could log the start and completion of subsystem shutdown by making calls similar to these:

```
csLogger -b 129.40.64.70 \  
-f /var/adm/SPlogs/cs/cshut.time.pid \  
subsystemA: initiating shutdown at 15:43.55 on 1 Apr 99.
```

```
csLogger -b 129.40.64.70 \  
-f /var/adm/SPlogs/cs/cshut.time.pid \  
subsystemA: shutdown complete at 15:46.23 on 1 Apr 99.
```

Chapter 3. Parallel Management Commands

This chapter describes commands provided with the SP system which are useful for performing common tasks on multiple nodes in parallel.*

These commands, along with the **sysctl** function and authenticated remote commands, provide for management of the SP system from a single point of control.

Using dsh to Execute Parallel Management Commands

You can use **dsh** from a single point of control to execute commands on:

- All nodes in the current system partition (**-a** option)
- All nodes in the SP system (**-aG** option)
- Selected nodes (**-w** option)
- Node groups (**-N** option)

You do not have to limit the target nodes to those nodes on your SP system. **dsh** will execute commands on any host in your network to which you can issue commands through the AIX authenticated remote command **rsh** (see “Security Considerations” on page 33). Although **dsh** uses **rsh** to issue remote commands, **dsh** offers better performance than an **rsh** loop because the remote commands run concurrently.

Inputting to dsh

dsh provides several ways for you to specify input:

1. **dsh** reads lines from standard input and executes each line as a command via **rsh**. Specify commands in **rsh** syntax. Pipes, filters, output redirection, and paths can be used on the remote nodes in the same way they are used in **rsh**.

The following **dsh** command targets three nodes, reads commands from stdin, and filters the output of the **ps** command remotely:

```
$ dsh -w host1,host2,host3
dsh> ps -ef "|" grep root
```

Note: Any input line beginning with an exclamation point is passed directly to the shell on the local host.

2. **dsh** accepts commands specified on the command line. For example, to issue the **ps** command on three nodes and filter the results remotely, specify:

```
dsh -w host1,host2,host3 ps -ef "|" grep root
```

* The parallel management commands described in this chapter were influenced by (but are not an implementation of) the tools described in Gropp, William and Ewing Lusk. 1994. Scalable UNIX tools on parallel processors. In *Proceedings of the Scalable High Performance Computing Conference*, 56-62. IEEE.

Targeting Nodes for dsh

dsh sends commands to a node set called the **working collective**. **dsh** assembles the working collective from the first existence of one of these sources:

1. A list of hostnames you specify on the command line with the **-a**, **-G**, **-w**, **-N** flags.
2. The contents of a file you specify in the **WCOLL** environment variable.

If neither of these exist, an error has occurred and no commands are issued.

You can have commands run on the working collective concurrently or in sequence. Default processing is done concurrently, but you can specify a maximum number of nodes for concurrent execution to prevent system degradation. When this maximum number, or *fanout* is reached, results from outstanding **rshs** are awaited before any further commands are run. These **rsh** results are displayed as soon as they return from the remotely executed commands.

This example specifies a maximum of 8 concurrent commands even though the working collective is defined as the entire system partition.

```
dsh -f 8 -a cat /var/adm/SPlogs/sysman/"*"config.log"* | pg
```

Note: Because **dsh** starts processes for each host in the working collective, there might be cases where a smaller fanout will result in less contention for resources on the local machine, and thus better overall performance, than the greater concurrency of a larger fanout.

Output from dsh

dsh displays returned information grouped by hostname. Stdout of the remotely executed commands goes to the stdout of **dsh**, and stderr of remotely executed commands goes to the stderr of **dsh**.

All **dsh** stderr and stdout lines are prefixed by the name of the host that sent them. You can format **dsh** stderr and stdout lines by piping them to **dshbak**, which strips off the hostnames and displays the lines grouped by host in alphabetic sequence, as shown in this example:

```
dsh -w host1,host2,host3 cat /etc/passwd 2>&1 | dshbak
```

Another example follows using a group of nodes:

```
dsh -N bis_nodes cat /etc/bootptab 2>&1 | dshbak
```

Alternatively, **dshbak** can be specified with the **-c** option. This will cause any identical output from two or more nodes to be shown only once, with the hostnames displayed above the output. Remember, however, that the most efficient way to filter large amounts of output from parallel commands is to filter on the nodes before the output is returned to the workstation from which the parallel command was issued.

Error Handling

No special error recovery is provided for **dsh** command failure on remote hosts. If **dsh** finds that a node in the working collective is down, no further commands will be sent to that node unless you specify the **-c** (continue) flag on the command line. If hosts are down, the underlying **rsh** will time out in approximately 2.5 minutes.

Security Considerations

The **dsh** command uses the authenticated remote command **rsh**. The **rsh** command attempts to authenticate users using Kerberos V4. Although the **rsh** command is capable of supporting other authentication methods, the **sysctl** and **hardmon** services require the use of Kerberos V4. For more information on security considerations, refer to “Installing and Configuring Distributed Authentication Services” on page 144.

The hostlist Command

The **hostlist** command can be used to specify hostnames for parallel command execution via **dsh** or **sysctl**. The command writes hostnames to stdout based on various criteria:

- The contents of a **dsh** working collective file
- The contents of a POE hostname file
- All the nodes in the current system partition (**-a** option)
- All the nodes in the SP system (**-aG** option)
- Ranges of slots on particular frames
- A set of nodes to exclude
- A range of node numbers
- Explicitly listed hostnames

Before the names are written, there is an optional check of whether the nodes are responding.

The output from **hostlist** can be written to a file, which can then be used as a working collective file for **dsh**. Both **dsh** and **sysctl** can read hostnames from stdin, so the output of **hostlist** can be piped directly to **dsh** or **sysctl**.

Examples

To issue a command to the node in the first slot in each of the first 4 frames:

```
hostlist -s 1-4:1 | dsh -w - command
```

To run the program on the nodes on all slots for frame 1 and on slots 1-3 for frame 3, and also on host **otherone**:

```
hostlist -n 1-16,33-35 -w otherone > /tmp/wcoll  
WCOLL=/tmp/wcoll dsh program
```

To determine all the nodes in the SP system on which one is authenticated, and to bypass the nodes that are not responding and “badnode”, issue the command:

```
hostlist -av -e badnode | sysctl -c - -L whoami | dshbak -c
```

Note: The **-L** option on **sysctl** is necessary if the **dshbak** filter is used. It causes the output lines of **sysctl** to be formatted with the hostname preceding each line.

Output might look like this:

```
HOSTS=====
host1          host2          host3          host4
host5          host6          host8
=====
joe.@A.B.C

HOSTS=====
host7
=====
unknown
```

If the **-c** option is not specified, output will not be collapsed:

```
HOST: host1
=====
joe.@A.B.C

HOST: host2
=====
joe.@A.B.C
:
```

List of Parallel Management Commands

The SP system provides several parallel management commands. They all have a common syntax and are based on **dsh**, **sysctl**, **hostlist**, and **dshbak**. More complete descriptions of each is provided in the *PSSP: Command and Technical Reference*.

The syntax of the parallel management commands is:

```
command_name
remote_host_specification cmd_args_flags
```

where

command_name The name of a parallel command

remote_host_specification

Indicates which hosts the command should be executed on.
Can be a range of slots, a list of hostnames, a set of flags to the **hostlist** command, or read from stdin.

cmd_args_flags The flags and arguments to the command

Where appropriate, the output from the parallel management commands is filtered as described previously.

The following parallel management commands are based on **dsh**.

pexec

This application issues a command in parallel on the specified hosts. Any arguments to the commands are passed along with the command name. Results are filtered as described in “Output from dsh” on page 32. For example, to show which users are logged on nodes in slots 5–8, use the command:

```
pexec 5-8 who
```

pexscr

This application executes commands in parallel on different hosts, including the local host. Unlike **pexec**, the commands might differ on different hosts. Lines consisting of *hostname: command* are read from stdin. When the lines have been read, each command is executed on the corresponding host in parallel. Remote commands are executed via **rsh** and local commands are executed directly. For example, you could create file **/etc/doing** on r05n09 containing the following text:

```
r05n09: /usr/lpp/ssp/rcmd/bin/rcp /tmp/bozoid tserv11:/tmp/bozoid
r05n09: /usr/lpp/ssp/rcmd/bin/rcp /tmp/bozoid r05n01:/tmp/bozoid
r05n01: backup_script
r05n15: backup_script
```

Then, to copy the local file **/tmp/bozoid** to tserv11 and r05n01 and simultaneously start backup scripts on r05n01 and r05n15, issue the command:

```
pexscr < /etc/doing
```

pcp

This command distributes a local file or directory to remote hosts in parallel. It uses the SP authenticated remote command **rCP**. To use this command, you must be authorized to issue commands of the form:

```
rcp localfile
remotehost: remotefile
```

for each of the hosts to which the file is to be copied. To recursively copy **/etc/new/** on the local host to **/etc/new1/** on the host in the first slot of each of the first four frames, use the command:

```
pcp "-s 1-4:1" -r /etc/new/ /etc/new1/
```

p_cat

This application issues the **cat** command in parallel on the specified hosts. Any arguments to the command are passed along with the command name. Results are not filtered. For example, the command

```
p_cat "-w host1,host2,host3" /etc/file > localfile
```

copies the contents of the **/etc/file** files on each of the hosts to the **localfile** file on the local host. **localfile** would consist of unmodified lines from **/etc/file** on **host1**, on **host2**, and on **host3** concatenated together.

pls, pfind, prm, pps, pmv

These commands issue the **ls**, **find**, **rm**, **ps**, or **mv** commands in parallel on the specified hosts. Any arguments to the AIX commands are passed along with the command name. Results are filtered. For example,

```
pls "-av -e badnode" /
```

lists the contents of the root directory on all the responding hosts in the SP system with the exception of "badnode". Note that the **-i** flags to **mv** and **rm** are not supported because **dsh** doesn't propagate the stdin of the remote hosts.

Examples

For example, a working collective could be set up with **hostlist**:

```
hostlist -av > /tmp/wcoll  
export WCOLL=/tmp/wcoll
```

```
pps "" -ef
```

would list information about all the processes on the nodes in the working collective.

```
pfind "" /usr/lpp/ssp/bin/ -name "pfind" -print
```

might result in the following:

```
HOSTS =====  
r05n01          r05n03dx          r05n05          r05n07  
r05n09          r05n10            r05n13          r05n14dx  
r05n15          r05n16  
=====
```

Host	Process
r05n01	r05n03dx
r05n09	r05n10
r05n15	r05n16

```
 /usr/lpp/ssp/bin/pfind
```

```
pmv "" /tmp/foo /tmp/newfoo
```

would rename **/tmp/foo** on all the working collective.

```
pls "" /tmp/\\*foo
```

could result in the following output:

```
HOSTS =====  
r05n01          r05n03dx          r05n05          r05n07  
r05n09          r05n10            r05n13          r05n14dx  
r05n15          r05n16  
=====
```

Host	Process
r05n01	r05n03dx
r05n09	r05n10
r05n15	r05n16

```
 /tmp/newfoo
```

Note that the shell metacharacter is escaped twice, once to protect it from the shell, and once to protect it when the **rsh**'s are done by **dsh**.

```
prm "" /tmp/newfoo
```

would remove **/tmp/newfoo** from all the nodes in the working collective.

ppred

This application issues a **ksh** test on each node in parallel and runs commands based on the results of the test. For example,

```
ppred 1-16 '-f /etc/foo' 'echo Here' 'echo Not here'
```

could result in output like the following:

```
HOSTS=====
host1          host2          host3          host4
host5          host6          host8
=====
Here

HOSTS=====
host7          host9          host10         host11
host12         host13         host14         host15
host16
=====
Not here
```

pfps

pfps is a parallel process find application. It takes arguments similar to those of the **find** command, but operates on processes instead of files. In addition, access control list-based permissions to **nice** or **kill** processes is provided. For example, to signal all the “daemon” processes in the SP system partition in parallel, a user that was in the **/etc/sysctl.pfps.acl** file on each node could issue the following:

```
pfps "-a" -tn sysctld -kill HUP
```

For example, to list the **sysctl** daemon processes on nodes **r05n15** and **r05n16**, use the command:

```
pfps "-w r05n15,r05n16" -tn sysctld -print
```

The output is similar to the following:

```
HOSTS =====
r05n15
=====
USER      PID %CPU %MEM  SZ  RSS  TTY STAT   STIME  TIME  COMMAND
root      11172 0.0  1.0  452  420   -  S    11:54:47  0:00
/usr/lpp/ssp/bin/sysctld -l /var/adm/SPlogs/sysctl/sysctld.log
root      12497 0.0  1.0  300  360   -  S    06:24:19  0:00
/usr/lpp/ssp/bin/sysctld -l /var/adm/SPlogs/sysctl/sysctld.log

HOSTS =====
r05n16
=====
USER      PID %CPU %MEM  SZ  RSS  TTY STAT   STIME  TIME  COMMAND
root      7089 0.0  1.0  448  420   -  S    11:54:48  0:00
/usr/lpp/ssp/bin/sysctld -l /var/adm/SPlogs/sysctl/sysctld.log
root      8423 0.0  1.0  296  360   -  S    13:01:31  0:00
/usr/lpp/ssp/bin/sysctld -l /var/adm/SPlogs/sysctl/sysctld.log
root      10418 0.0  1.0  512  504   -  S
```

pdf

pdf is a parallel command that performs function similar to **df**. It provides more information than the **df** command. To get filesystem information from all the nodes on the first slots of each of four frames, use the commands:

```
hostlist -s 1:1-4 > /tmp/wcoll
export WCOLL=/tmp/wcoll
pdf ""
```

To find information on the **/var** filesystem on the nodes in the working collective, use the command:

```
pdf "" /var
```

The output is similar to the following:

Filesystem	Size-KB	Used-KB	Free-KB	%Free	iUsed	iFree	%iFree
=====	=====	=====	=====	=====	=====	=====	=====
HOST: r05n01							
=====							
/var	8192	2452	5740	71%	279	1769	87%
HOST: r05n03dx							
=====							
/var	8192	1988	6204	76%	214	1834	90%
HOST: r05n05							
=====							
/var	8192	2080	6112	75%	223	1825	90%
HOST: r05n07							
=====							
/var	8192	2080	6112	75%	223	1825	90%
HOST: r05n09							
=====							
/var	8192	2088	6104	75%	223	1825	90%
HOST: r05n10							
=====							
/var	8192	2448	5744	71%	286	1762	87%
HOST: r05n13							
=====							
/var	8192	2220	5972	73%	229	1819	89%
HOST: r05n14dx							
=====							
/var	8192	2012	6180	76%	221	1827	90%
HOST: r05n15							
=====							
/var	8192	2368	5824	72%	220	1828	90%
HOST: r05n16							
=====							
/var	8192	2100	6092	75%	225	1823	90%

pfck

pfck displays file system information in parallel based on various usage criteria. For example, to find information on all the file systems on the hosts in POE host list file **hostlist** that have used more than 100000 KB, use the command:

```
MP_HOSTFILE=./hostlist pfps "" -s 100000
```

To display the filesystems on hosts **r05n16** and **r05n15** that have more than 90% of their space used, use the command:

```
pfck "-w r05n15,r05n16" -pu 90
```

The output is similar to the following:

Filesystem	Size-KB	Used-KB	Free-KB	%Free	iUsed	iFree	%iFree
=====	=====	=====	=====	=====	=====	=====	=====
HOST: r05n15							
=====							
/usr	331776	327640	4136	2%	17673	66295	79%
HOST: r05n16							
=====							
/usr	331776	327640	4136	2%	17673	66295	79%

To find all the **/var** filesystems in the SP system that are 95% full, use the command:

```
pfck "" -pu 95 | egrep "(var|HOST)"
```

The output is similar to the following:

Filesystem	Size-KB	Used-KB	Free-KB	%Free	iUsed	iFree	%iFree
=====	=====	=====	=====	=====	=====	=====	=====
HOST: r05n10							
/var	335872	331412	4460	2%	17819	66149	79%
HOST: r05n13							
/var	335872	331632	4240	2%	17820	66148	79%
HOST: r05n14dx							
/var	335872	331388	4484	2%	17817	66151	79%

Chapter 4. Managing File Collections

The SP system installs file collection technology by default, to simplify the task of maintaining duplicate files on multiple machines. In the delivered system, the files that are required on the control workstation, boot/install servers, and processor nodes belong to file collections. The master copy of each collection is on the control workstation. A process checks periodically, perhaps once each hour, if any file has been updated. Each file that was updated since the last time it checked, is propagated to the nodes that also have a copy.

This chapter explains the file collections that are delivered with the system. It shows you how to report on, update, and maintain the system file collections, as well as how to build file collections for your own purposes. It explains the basic concepts you need to understand before working with file collections, such as what file collections are, how they differ from standard directories and files, and how they are used in the SP system.

You can easily maintain the integrity of your files also by grouping them into file collections and using the tools provided to manage them. You can manage updates to your files with consistency and accuracy across the multiple nodes in your SP system. With a master version on the control workstation, you can be assured that any modifications are propagated to the nodes within the set time interval.

You might want to take advantage of file collections to maintain files such as **/etc/environment** and **/.kshrc**. You need to read the entire chapter for complete information but briefly, the steps for a typical scenario might be as follows:

1. Develop a list of files you want to update only on the control workstation and have distributed to the nodes of your SP system.
2. Add that list to the **list** file in the file collection **/var/sysman/sup/user.admin**, one line for each file you want distributed, using the syntax:

```
upgrade <full path name of the file>
```

For example,

```
upgrade ./etc/environment
```

3. Within the set interval, all the files in the **list** file will be distributed to the nodes. If you do not want to wait, you can force an immediate update by using the command:

```
dsh -a supper update user.admin
```

If your PATH environment variable through dsh is not set to access the **supper** command, you might need to use:

```
dsh -a /var/sysman/supper update user.admin
```

There might be cases when you want to distribute some files on the nodes but keep a different version on the control workstation for things you want to start on the nodes but not on the control workstation, for example file **/etc/rc.tcpip**. Use file collection **node.root** in that case, as in this example:

1. To the **list** file in the file collection **/var/sysman/sup/node.root**, add the following:

```
upgrade .
```

2. Put the files you want distributed in subdirectories of **/share/power/system/3.2** on the control workstation. For example **/share/power/system/3.2/etc/rc.tcpip** will be distributed to the nodes as **/etc/rc.tcpip**. After this, any file added to the **/share/power/system/3.2** directory is automatically distributed to the nodes.

3. To force immediate update, use the command:

```
dsh -a supper update node.root
```

The file collection technology provides you with the means to perform both beginning basic tasks and optional advanced tasks. For the basic tasks, you need only understand the fundamental concepts and be familiar with the delivered file collections. The more advanced tasks require an understanding of the master files, what they contain, and how they work.

Information is presented on the following basic file collection tasks:

- Understanding file collections
- Reporting file collection information
- Verifying file collections using **scan**
- Updating files in a file collection
- Granting permission to obtain a file collection
- Adding and deleting files in a file collection

Information is presented on the following advanced file collection tasks:

- Building a file collection
- Installing a file collection
- Refusing files in a file collection
- Modifying the file collection hierarchy
- Removing a file collection

Understanding File Collections

A *file collection* is a set of files and directories that are duplicated on multiple machines in a network and managed by tools that simplify their control and maintenance. Included with the administrative software is a program called **supper**. The **supper** perl program (**/var/sysman/supper**) uses the Software Update Protocol (SUP) to manage the SP file collections and transfer them across the system.

The following terms are used when defining file collections:

Resident A file collection that is installed in its true location and able to be served to other systems

Available A file collection that is not installed in its true location but able to be served to other systems

File collections have unique features:

- A file collection directory does not contain the actual files in the collection. Instead, it contains a set of master files to define the collection. Some master files contain rules to define which files can belong in the collection and others contain control mechanisms, such as time stamps and update locks. These files are explained in “Directory and Master Files” on page 46.

- You handle files in a collection with special procedures and **supper** commands rather than with the standard AIX file commands. The **supper** commands interpret the master files and use the information to install or update the actual files in a collection. You can issue these commands in either batch or interactive mode. The procedures and use of the **supper** commands are detailed throughout this chapter. You can find the command syntax and descriptions in the book *PSSP: Command and Technical Reference*.
- File collections can be either *primary* or *secondary*. Primary file collections can contain other secondary file collections. When a primary collection that contains secondary collections is installed, the secondary collections are *available* but not *resident*. This means they can be served from this location but not actually executed or used there. “File Collection Types” provides further explanation of these concepts.
- File collections require special entries in the file **/var/sysman/file.collections** to define them to the **supper** program. They also require a symbolic link in the **/var/sysman/sup/lists** file pointing to their **list** master file. See “Building a File Collection” on page 58 for an explanation of these entries.
- File collections also require a unique unused user ID for `supfilesrv`, the file collection daemon, which consists of the user name **supman** and its associated default user ID of 102, along with a unique, unused port through which it can communicate. The default installation configures the user ID attribute `supman_uid` to 102 and the port attribute `supfilesrv_port` to 8431. You can change these values using SMIT or the **spsitenv** command.
- The file collection daemon requires read access permission to any files that you want managed by file collections. For example, if you do not use the SP User Management component of PSSP and you want to distribute files that are under the **/etc/security** directory, you must add **supman** to the security group file **/etc/groups**. This gives read access to the `supfilesrv` daemon for files that have security group permission, enabling them to be managed across the SP by file collections.

File Collection Types

File collections can be either primary or secondary. A primary file collection can be a stand-alone collection or it can contain a secondary collection. Having a secondary collection allows you to keep a group of files available on a particular machine to serve to other systems without having those files installed.

For example, if you want to have one **.profile** on all nodes and another **.profile** on the control workstation, consider using the **power_system** collection delivered with the IBM Parallel System Support Programs for AIX. This is a primary collection that contains **node.root** as a secondary collection.

- Copy **.profile** to the **/share/power/system/3.2** directory on the control workstation.
- If you issue **supper install power_system** on the boot/install server, the **power_system** collection is installed in the **/share/power/system/3.2** directory. Because the **node.root** files are in that directory, they cannot be executed on that machine but are available to be served from there. In this case, **.profile** is installed as **/share/power/system/3.2/.profile**.
- If you issue **supper install node.root** on a processor node, the files in **node.root** collection are installed in the **root** directory and, therefore, can be

executed. Here, `/share/power/system/3.2/.profile` is installed from the file collection as `/.profile` on the node.

Predefined File Collections

Looking at the file collections delivered with the SP system can help you to understand file collection concepts. You should become familiar with the delivered file collections before you begin to update or add to them.

The SP system provides four predefined file collections:

- **sup.admin**
- **user.admin**
- **power_system**
- **node.root**

You can display information about each collection on a particular machine using the **supper status** command. This reports whether or not the file collection is resident on that machine. To do this, log in as **root** or issue the command remotely. Enter:

```
/var/sysman/supper status
```

You can see a list of the files in a file collection by looking at the **scan** file. See “Verifying File Collections Using scan” on page 51 for instructions on how to do this. The **scan** file contains an entry for each file in the collection that could be installed on a boot/install server or processor node. Certain files in the collection might not exist on one particular system depending on the contents of the **refuse** file on that system. You can find more information about the **scan** and **refuse** files in “Directory and Master Files” on page 46.

SP File Collection Summary

The following table summarizes where these file collections are *resident* or *available* in the SP default configuration.

Control Workstation		Boot/Install Servers		Processor Nodes	
Resident	Available	Resident	Available	Resident	Available
	sup.admin user.admin power_system node.root	sup.admin user.admin node.root	sup.admin user.admin power_system node.root	sup.admin user.admin node.root	

sup.admin Collection

The **sup.admin** file collection is a primary collection that is *available* from the control workstation, is *resident* (that is, installed) and *available* on the boot/install servers, and *resident* on each processor node.

This file collection is important because it contains the files that define the other file collections. It also contains the file collection programs used to load and manage the collections. Of particular interest in this collection are:

- `/var/sysman/sup` which contains the directories and master files that define all the file collections in the system
- `/var/sysman/supper` which is the Perl code for the **supper** tool
- `/var/sysman/file.collections` which contains entries for each file collection.

user.admin Collection

The **user.admin** file collection is a primary collection that is *available* from the control workstation, *resident* and *available* on the boot/install servers and *resident* on each processor node. This file collection contains files used for user management:

- **/etc/passwd**
- **/etc/group**
- **/etc/security/passwd**
- **/etc/security/group**

The collection also includes the password index files which are used for login performance:

- **/etc/passwd.nm.idx**
- **/etc/passwd.id.idx**
- **/etc/security/passwd.idx**

power_system Collection

The **power_system** file collection is used for files that are system dependent. It is a primary collection that contains one secondary collection called the **node.root** collection. The **power_system** collection contains no files other than those in the **node.root** collection.

The **power_system** collection is *available* from the control workstation and *available* from the boot/install servers. When the **power_system** collection is installed on a boot/install server, the **node.root** file collection is *resident* in the **/share/power/system/3.2** directory and can be served from there.

node.root Collection

This is a secondary file collection under the **power_system** primary collection. The **node.root** collection is *available* from the control workstation, *resident* and *available* on the boot/install servers and *resident* on the processor nodes. It contains key files that are node-specific.

The **node.root** file collection is *available* on the control workstation and the boot/install servers under the **power_system** collection so that it can be served to all the nodes. We do not install **node.root** on the control workstation because the files in this collection might conflict with the control workstation's own **root** files.

Primary and secondary files are defined in the **file.collections** file (see “Step 5: Update the file.collections File” on page 60).

File Collections Organization

When you complete the SP installation and configuration process, the delivered file collections are organized in a hierarchy with the control workstation as the master server for all the collections.

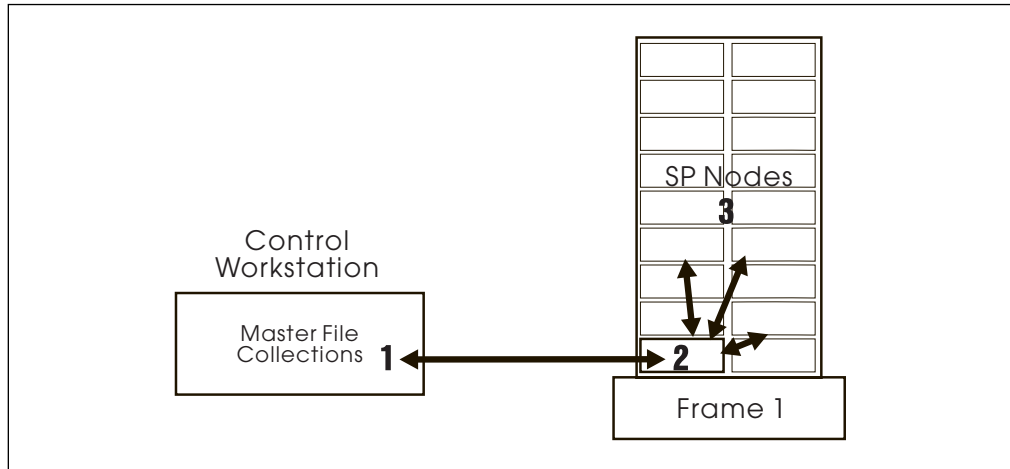


Figure 2. SP Default File Collection Hierarchy

The commands to update the file collections are set up in a **crontabs** file to run hourly in a staggered sequence. Figure 2 shows this organization. Files are updated on the control workstation in the master file collections (1). The boot/install servers use **supper update** to request the changes from the control workstation (2) and the nodes use **supper.update** to request the changes from the boot/install servers (3).

Understanding the File Collection Structure

Before you work with the delivered file collections or build a file collection of your own, you need to understand the directories, master files, what they contain, and how they work.

Directory and Master Files

The **/var/sysman/sup** directory contains master files for all the file collections. Figure 3 shows the structure for this directory.

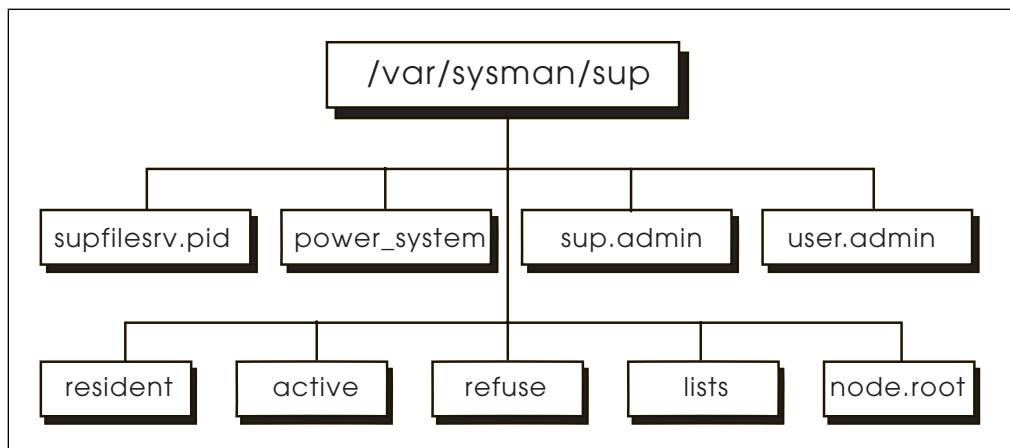


Figure 3. /var/sysman/sup Files and Directories

Following is an explanation of the files and directories.

.active

A file identifying the active volume group. You can modify this file with the **supper activate** command. (This file is not on the control workstation.) This file is present on the nodes depending upon the actions of the administrator.

- .resident** Lists each file collection in the SP system. This file needs to be updated when you are adding a new file collection. (This file is not on the control workstation.)
- lists** A directory that contains links to the **list** files in each file collection.
- node.root** Directory of the master files in the **node.root** collection.
- power_system** Directory of the master files in the **power_system** collection.
- refuse** On a client system, this is a user-defined (you must define the file; it is not supplied) text file containing a list of files to exclude from all the file collections. This allows you to customize the file collections on each system.

You list the files to exclude by their fully qualified names, one per line. You can include directories, but you must also list each file in that directory you want excluded. This file is present on the nodes depending upon the actions of the administrator.
- sup.admin** Directory of the master files in the **sup.admin** collection.
- supfilesrv.pid** The process ID of the SUP **supfilesrv** process.
- user.admin** Directory of the master files in the **user.admin** collection.

Each individual file collection is composed of a directory and several master files. Figure 4 shows the structure for the **sup.admin** file collection in the **/var/sysman/sup/sup.admin** directory. It includes a sample of the file contents. This example shows eight master files. Other file collections might have fewer master files.

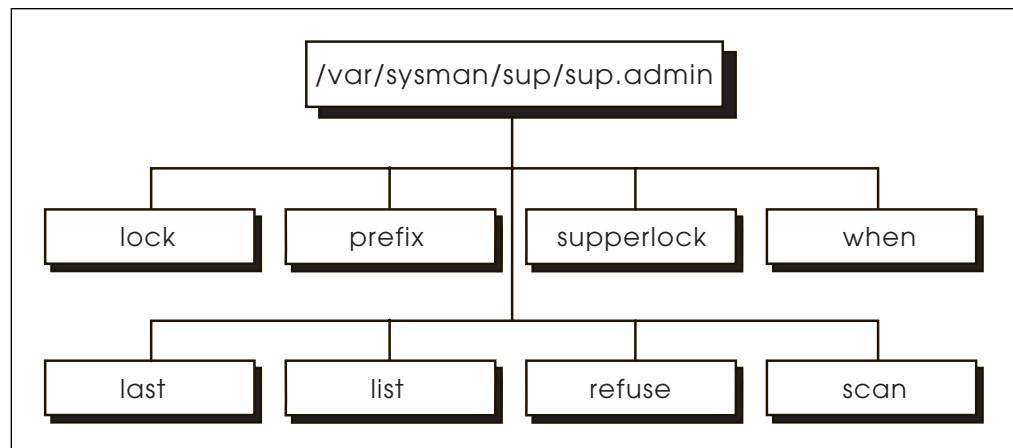


Figure 4. *sup.admin* Master Files

Following is an explanation of the master files shown in Figure 4.

- last** A system-maintained list of files and directories that have been updated. This helps identify files to be deleted on a local machine when they are no longer part of the collection.

The **supper files** command reads the **last** file to obtain a current list of files in the collection. This file is present on the nodes depending upon the actions of the administrator.

list	A user-defined file containing special commands. These commands are interpreted by the supper scan process to define which files belong to the file collection relative to the base directory specified in the prefix file. You can find the specific format for these commands in “supper list File Syntax, Keywords, and Operands” on page 66.
lock	An empty file used by SUP to lock a collection and prevent more than one update of the same collection at the same time.
prefix	A file containing the name of a directory to be used as the base directory for file references and the starting point for the supper scan process. This file is created by the system from data in the file.collections file.
refuse	A system-created file that contains a list of all the files excluded during the update process. If there are no files for this collection listed in the refuse file in the /var/sysman/sup directory, the refuse file in this directory will have no entries.
scan	A system-created file containing a list of files that comprise the collection and their permissions and time stamps. This file is the result of the supper scan process. It is optional but suggested for all collections and can cause maintenance problems if you do not keep it current.
supperlock	Similar to lock , an empty file used by supper to lock a collection during updates.
when	A system-maintained file containing the time of the last file collection update. This file protects against accidental update of a collection with an old version of the files. You can reset it using the supper reset command if you need to override this restriction. This file is present on the nodes depending upon the actions of the administrator.

How the Master Files Work

The **supper** master files play a key role during the install, update, and scan processes.

When you run the **supper scan** command for a file collection, the search begins at the point in the directory specified in the **prefix** file and traverses the directory tree to find all the files that meet the criteria defined in the **list** file. The output of this process is the **scan** file which lists the files in the collection. The **scan** file is optional but it documents the file collection contents and eliminates the need for the install and update processes to do the directory search.

The **supper install** and **update** commands use the **scan** file, if it is present, to identify the files to install or update. If a **scan** file is not present, **supper** performs the same directory search to identify the files as it performs when it creates the **scan** file. These commands also check the **refuse** file in **/var/sysman/sup** at each location and bypass any files it contains. As a result, these commands create a **refuse** file in the file collection's directory, listing the files that were bypassed.

During updates, the **lock** and **supperlock** files prevent update access by other SUP or **supper** commands. **supper** checks the update time in the **when** file against the actual file time stamps to verify that you are not updating with an old

version of the files and, if so, does not update. It uses the **last** file for additional verification when deleting files that no longer exist in the file collection.

Master File Processing Order: Whether invoked with the **scan**, **update**, or **install** commands, **supper** performs the search process in the following manner:

- prefix** Reads this file first to obtain a directory from which it will start the search
- list** Reads this file next and processes its commands in the following order:
 - omit** Marks these files for omission
 - upgrade** Marks these files for upgrade when updating or installing
 - always** Overrides any files marked for omission
- refuse** Reads this file last when it is present on a client (boot/install server or processor node) and does not upgrade any of the specified files on that client.

Automatic Updates: The **supper update** commands are included in the **crontabs** file with the default set to run an update on each of the file collections hourly. You can modify the **crontabs** file to run the **supper update** more or less frequently. Refer to “Updating crontabs on the SP Nodes” on page 127 for more information.

Basic File Collection Tasks

In order to maintain the delivered file collections you will need to perform some basic tasks such as reporting on file collection information, updating files that belong to an existing file collection, and possibly adding or deleting files within an existing file collection. For example:

- To become familiar with the delivered collections you might list their name, access point, and whether or not they are resident. You might also want to verify the complete list of all the files in a collection.
- You might update files such as **/etc/profile** to change the user default profile.
- You might add files such as **/etc/environment** and have them managed by the file collection processes.

You can find basic file collection tasks on pages 42 through 56.

Advanced File Collection Tasks

Once you are familiar with file collection technology, you might want to build your own collections to manage other files on your system that you would like duplicated on multiple machines. You might have a group of files such as application data that you want installed on each node or local tools you want installed on each boot/install server.

The task of building and distributing your own collections is more advanced. It includes creating the directory and master files, adding entries to the **file.collections** and **lists** files, and possibly creating **scan** and **refuse** files. Once you complete these tasks and build your file collection, you can install it on the machines in your system. You can also install a file collection from a server other than its default server.

You can find advanced file collection tasks on pages 63 through 65.

Reporting File Collection Information

supper has subcommands that report information about file collections. These subcommands are useful for verifying information before performing file collection manipulations or for checking results at some point during a procedure.

The following table shows the **supper** subcommand to use, where you would run the command, and the data you can report. See the book *PSSP: Command and Technical Reference* for additional command options.

Use:	On:	To Report This Data:
status	Control workstation, boot/install servers, and nodes	Name, resident status, and access point of all available file collections, plus the name and estimated size of their associated file systems
files	boot/install servers and nodes	All the resident files resulting from a supper update or install command
serve	Control workstation and boot/install servers	All the collections that can be served from your machine
where	Boot/install servers and nodes	Current boot/install servers for collections
when	Boot/install servers and nodes	Last update time of all resident collections
diskinfo	Control workstation and boot/install servers	Available disk space and active volume for your machine
log	Boot/install servers and nodes	Summary of the current or most recent supper session
rlog	Boot/install servers and nodes	Raw output of the current or most recent supper session

The following examples show the output produced by some of these commands.

From the Control Workstation:

```
[tserv11] [/]> supper status
```

```

Collection          Resident  Access Point          Filesystem          Size
=====
node.root           -        /                    -                   -
power_system       -        /share/power/system  -                   -
sup.admin          -        /var/sysman          -                   -
user.admin         -        /                    -                   -
=====

```

From a boot/install server:


```
[r05n01][/]> supper status
```

Collection	Resident	Access Point	Filesystem	Size
node.root	Yes	/	-	-
power_system	Yes	/share/power/system	-	-
sup.admin	Yes	/var/sysman	-	-
user.admin	Yes	/	-	-

From a processor node:

```
[r05n10][/]> supper status
```

Collection	Resident	Access Point	Filesystem	Size
node.root	Yes	/	-	-
power_system	-	/share/power/system	-	-
sup.admin	Yes	/var/sysman	-	-
user.admin	Yes	/	-	-

Example of a **supper where** command:

```
[r05n01][var/sysman/sup/node.root]> supper where
supper: Collection node.root would be updated from server r05cw.
supper: Collection power_system would be updated from server r05cw.
supper: Collection sup.admin would be updated from server r05cw.
supper: Collection user.admin would be updated from server r05cw.
```

Verifying File Collections Using scan

The **scan** file provides an inventory of all the files and directories in the file collection. By looking at the **scan** file you can quickly see which files are in a collection. You can find the **scan** file in the file collection's directory in **/var/sysman/sup**.

Although **scan** files require maintenance whenever files are added, deleted, or changed in the master collection, they save processing time on larger file systems.

Note

You must update the **scan** file each time you modify the file collection or you risk accidentally updating the collection from an outdated master file.

Example

To check for an existing **scan** file in the **user.admin** file collection, enter:

```
ls /var/sysman/sup/user.admin
```

To create a **scan** file for the **user.admin** file collection, enter:

```
/var/sysman/supper scan user.admin
```

Following is a sample of a **scan** file for the **user.admin** collection. It shows the name of each file preceded by three sets of numbers. The first set is the file

permissions, the second set indicates when the collection was installed, and the third set indicates when the collection was last updated.

```
V2
100644 898550069 898291329 etc/amd/amd-maps/amd.u
X/etc/amd/refresh_amd
100644 902502604 902502321 etc/auto.master
100644 902499005 902495923 etc/auto/maps/auto.tstauto
100644 898550069 898291329 etc/auto/maps/auto.u
X/etc/amd/refresh_amd
100664 903622203 903620838 etc/group
100664 903622203 903621192 etc/passwd
100664 903625802 903624000 etc/passwd.id.idx
100664 903625802 903624000 etc/passwd.nm.idx
100640 903622203 902858646 etc/security/group
100644 903622203 903620838 etc/security/passwd
100644 903622203 903620838 etc/security/passwd.idx
```

Note

There are special considerations when using scan files in an HACWS environment. In this environment, a control workstation can be either a file collections client or a server depending on whether it is active. Refer to “Managing File Collections” on page 305 for more information.

Updating Files in a File Collection

The SP file collections can be used to contain system control files that you will need or want to change. When you update files that are part of a file collection, you do not have to make the same update on each replication of the files. Instead, you make the update to the copy on the master server for that collection and run the **supper update** command on all systems you want updated. The **supper** program does the updates for you.

The **supper update** command is included in the **crontabs** file with the default set to run hourly. You can modify the **crontabs** file to run the **supper update** more or less frequently. Refer to Chapter 11, “Maintaining the crontabs File” on page 127 for more information.

Step 1: Make the Changes to the Files

1. Be sure you are working with the master files.

In a file collection there are many replications of the same file. It is possible for copies of a file to exist on the control workstation, on several boot/install servers, and on all the processor nodes.

In the default system, the control workstation is the master server for all file collections. If you have not changed anything in this configuration, this is where you will always make updates.

Note

If you changed the default configuration, be aware of the hierarchy of servers in your system and be sure you are modifying the copy of the file that is on the master server for the collection.

See “File Collections Organization” on page 45 to review file collection hierarchy.

2. Change the files as appropriate.

Example

In our default configuration, the master files for all file collections are on the control workstation.

Log in to the control workstation as **root** and copy a new version of **/.profile** into the **/share/power/system/3.2** directory.

Step 2: Run the supper scan Command

To keep the **scan** file current, you must run the **supper scan** command on the master collection every time you make a change.

Example

Run the **supper scan** command on the control workstation. Enter:

```
/var/sysman/supper scan power_system
```

Step 3: Run the supper update Command

You need to run the **supper update** command in every location that the collection is installed and where you want the change to be effective. For example, if you change the files on a server that is the master for eight processor nodes and you want each node to have the changes, you need to run the update on all eight nodes.

You can accomplish this update in several ways:

1. You can wait for the scheduled update. The **supper update** command is included in the **crontabs** file to be run on a scheduled basis.
2. You can issue the commands immediately, logging in to each machine or remotely issuing the commands with **rsh** or **rexec**.

Example

1. Log in to each server requiring the update. The **.profile** file is in the **node.root** collection as a secondary collection available within the **power_system** collection. This means on the boot/install servers you update the **power_system** collection. Enter:

```
/var/sysman/supper update power_system
```

2. Log in to each of the nodes requiring the update. On the nodes you update the **node.root** collection. Enter:

```
/var/sysman/supper update node.root
```

3. Check the **supper** messages. **supper** returns the number of files updated, removed, and the number of errors produced as shown in this message which

was returned after changing **.profile** and performing the update on a processor node.

```
=====
[r02n07][/]> supper update node.root
Updating collection node.root from server mrCS-r1.
File Changes: 1 updated, 0 removed, 0 errors.
=====
```

supper also writes summary messages to the following file:

/var/adm/SPlogs/filec/sup.<month>.<day>.<year>.<hour >.<min>

Detailed information is written to the following file:

/var/adm/SPlogs/filec/sup.<month>.<day>.<year>.<hour>.<min>.r

File collection log files older than 48 hours are removed at midnight by a **crontab** entry. To see all the messages in the log, you have to display or print the **logs** file. You can check the **supper** log for messages from the last install or update using a **supper** command. The following example shows a **supper update**, followed by a **supper rlog** command:

```
[r05n01][var/adm/SPlogs/filec]> supper update node.root
Updating collection node.root from server r05cw.
File Changes: 2 updated, 0 removed, 0 errors.
```

```
[r05n01][var/adm/SPlogs/filec]> supper rlog
SUP 7.24 (4.3 BSD) for file /tmp/.sf13689 at May 6 12:01:57
SUP Upgrade of node.root at Fri May 6 12:01:57 1994
SUP Fileserver 7.12 (4.3 BSD) 26295 on r05cw.hpssl.kgn.ibm.com
SUP Locked collection node.root for exclusive access
SUP Requesting changes since Fri May 6 12:00:07 1994
SUP Receiving file .kshrc
SUP Receiving file .profile
SUP Upgrade of node.root completed at Fri May 6 12:01:58 1994
```

Granting Permission to Obtain a File Collection

Permission to obtain file collections that are standard with the SP system software is automatically granted to SP nodes. You can choose to grant permission for these file collections to non-SP hosts as well. You can also create your own file collections and control distribution of them.

Automatic Permissions for SP Nodes

During the configuration of file collections, a per-collection **host** file is created to limit access for the IBM-defined collections to the nodes of the SP system. The file resides in the directory **/var/sysman/sup/file_collection/host**, where *file_collection* is the unique name of a file collection. The file collections server, **supfilesrv**, uses this file when checking whether a host has been granted access to obtain the collection in which the **host** file resides. The file collections server runs on the control workstation. If your SP system uses boot/install servers, the file collections server also runs on each node that is a boot/install server. Therefore, the per-collection **host** file is created and maintained on the control workstation and any boot/install servers you have defined.

The **host** file is updated with the hostnames of the adapters in the SP system as found in the SDR. The file is updated during the following events:

- configuration
- reboot of the server host
- addition of node
- deletion of node
- addition of adapter
- deletion of adapter
- definition of boot/install server
- failover of the control workstation by HACWS

The **filec_host** command is run periodically to update the per-collection **host** file. Only the host names of the adapters in the SP system are automatically maintained in the file. They can be distinguished by the special comment:

```
#generated_entry_donor_edit
```

These generated entries should not be deleted. You can add names of other non-SP hosts to the **host** file to permit those hosts access to the SP file collections. It is your responsibility to set up the proper client on the non-SP hosts using the **SUP** public code that comes with the SP software.

The **filec_host** command reads the **/var/sysman/collection.host.list** file to obtain a list of the collections for which the **host** file must be created or updated. The **collection.host.list** file contains the names of the IBM-delivered collections by default. If you have collections in the **/var/sysman** path that you want to define and distribute to SP nodes, you need to add their names to the **collection.host.list** file so that the **host** file gets created and updated for the SP nodes to be granted access to them. The **collection.host.list** file is distributed through the SP **sup.admin** file collection.

The **host** file is not automatically distributed. Distributing the **host** file through the file collections server would place the file on the nodes as well as any boot/install servers. Although this is not a security risk since the file collections server does not normally run on the nodes, whether you want to distribute the **host** file is up to you. If you do want it distributed, see “Distributing the Per-Collection Host File” on page 56.

Granting Permissions to Non-SP Hosts

To grant permission for obtaining a file collection to a non-SP host, edit the **/var/sysman/sup/file_collection/host** file and add the name of the host to which permission is granted. The file should be updated on the host that services the requests for the respective collections. This could be the control workstation or an SP node that is a boot/install server. Depending on the number of host names (one per adapter) on the host being granted permission, you might need to add more than one host name for the host.

Since the **host** file grants permissions to only one collection, the collection of which it is part, you might need to edit the **host** file for each collection to add the non-SP hosts so that the non-SP host can obtain all collections.

Do not remove the entries with the specially generated comments. They are automatically generated to grant the SP nodes access to the IBM-delivered file collections. You can add your own entries anywhere in the per-collection **host** file. However, when the file is automatically updated, all customer entries are placed at the top of the file and are followed by all the SP-generated entries.

Distributing the Per-Collection Host File

If you plan to distribute the per-collection **host** file using the file collections server, you must make all updates to these per-collection **host** files on the control workstation. They get distributed to all boot/install servers (as well as the nodes and other hosts you might have added), overwriting any **host** files that might reside there. If you have edited the **host** file on a boot/install server rather than on the control workstation, you will lose those changes when the file gets distributed from the control workstation.

The per-collection **host** file can be distributed through the file collections server by adding the following to the `/var/sysman/sup/sup.admin/list` file:

```
always ./var/sysman/sup/*/host
```

Adding and Deleting Files in a File Collection

Adding and deleting files in a file collection are a bit more complicated than adding and deleting files in a standard directory. The **prefix**, **list**, and **refuse** master files contain criteria that define which files are in a particular collection. The master files are read and processed by the **scan**, **install**, and **update** commands. When you add or delete files in a file collection, you need to review the contents of these master files. For more information about the master files and how they work, see “Building a File Collection” on page 58.

The examples in this section show how to add the `/etc/environment` file to the **node.root** collection.

Step 1: Add or Delete Files in the File System

1. **Be sure you are working with the master files.**

The same considerations you make to update files also apply here. Verify that you are on the machine that is the master server for the collection you are changing. See “File Collections Organization” on page 45 to review file collection hierarchy.

2. Add or delete files in the file system using standard AIX file commands.

When adding files, you need to consider whether the files are in a primary or secondary collection, what the **prefix** and **list** files in that collection contain, and the position of the new files in the file tree. Remember, the **supper** commands begin their search starting at the point defined in the **prefix** file. All files encountered that pass the criteria defined in the **list** file are considered part of the file collection.

Examples

1. To add a file to the **node.root** collection which is a secondary collection under the **power_system** collection, you would copy the file to the **/share/power/system/3.2** directory. The **/var/sysman/sup/node.root/prefix** file shows that the directory search for this collection begins at **/share/power/system/3.2**. The **/var/sysman/sup/node.root/list** for this example contains the following:

```
symlinkall
omit    ./usr
upgrade .
```

The entry **upgrade .** indicates that **all** files encountered in the **/share/power/system/3.2** directory during the search are to be upgraded.

To add **/etc/environment** in our default configuration, log in to the control workstation and copy **/etc/environment** to the **/share/power/system/3.2** directory:

```
cp /etc/environment /share/power/system/3.2/etc/environment
```

2. To add a file to the **user.admin** collection, you would proceed differently. The **/var/sysman/sup/user.admin/prefix** file shows that the directory search starts at the root of the file tree, therefore, the directory search would find the **/etc/environment** file. The **/var/sysman/sup/user.admin/list** file shows that only those files specifically listed will be upgraded instead of the entire directory.

In this case, you do not need to copy the file. Instead, add a specific entry to the **list** file to upgrade it.

Note

To add a file to a collection, the file must permit others to have read access.

To delete a file from a file collection, either remove it on the master server or delete its entry in the **list** file and proceed with the other steps in this task.

For example, to remove **/etc/environment** from the **node.root** collection, log in to the control workstation and enter:

```
rm /share/power/system/3.2/etc/environment
```

Step 2: Run the supper scan Command (optional)

The **scan** is not a required file and not all file collections have one. Check the file collection directory to see if the collection you are updating has a **scan** file.

- If you have a **scan** file in this collection and you want to maintain it, you must run the **supper scan** command on the master collection every time you make a change. When a **scan** file is present, the **update** command reads it as an inventory of the files in the collection and does not do the directory search.

You also have the option of deleting the **scan** file and allowing **supper** to process the file collection without it.

Note

If you keep the **scan** file and fail to run the **supper scan** command when you add files, the **scan** file will not be current and the new files will not be included in the collection.

- If you do not have a **scan** file in this collection, the **update** command will search the directory, apply the criteria in the master files, and add the new file.

Example

1. Run the **supper scan** command on the control workstation:

```
/var/sysman/supper scan power_system
```

This builds a new **scan** file for the collection.

2. Check the **scan** file to verify that it contains the additional **/etc/environment** file.

Step 3: Run the supper update Command

Run the **supper update** command everywhere you want this change to be effective. See “Step 3: Run the supper update Command” on page 53 for your options in performing this step.

Example

1. Log in to each server requiring the new file. The **/etc/environment** file is in the **node.root** collection as a secondary collection available within the **power_system** collection. Enter:

```
/var/sysman/supper update power_system
```

2. Log in to each of the nodes requiring the new file. On the nodes you update the **node.root** collection. Enter:

```
/var/sysman/supper update node.root
```

3. Check the **supper** messages to verify the updates worked.

Building a File Collection

Once you become familiar with file collections and have worked with the delivered system collections, you might want to create your own. You can build a file collection for any group of files that you want to have identically replicated on nodes and servers in your system. These file collections must all reside on the control workstation or a boot/install server. Some good candidates for file collections are application data files or local tools.

The basic procedure to build a file collections is:

- Begin by logging in as **root** to the server where the files reside.
- Create the directory and create a set of master files by copying them from an existing file collection. The master files define the rules for including and excluding files in the collection.
- Optionally, run the **supper scan** command to search the file tree, apply the rules, and build a **scan** file.
- Set up a link in the **/var/sysman/sup/lists** for the new file collection's **list** file.

- Add an entry to the **file.collections** file to define the file to the **supper** program.

After you build your file collections, you can install them on other servers or nodes.

This section provides examples showing you how to build a primary file collection for a set of local tools in the **/usr/local** directory for the control workstation and boot/install servers.

Related Information

Before you build a file collection of your own, be sure you understand the directory and master files, what they contain, and how they work. Refer to “Understanding File Collections” on page 42 if you need to review these concepts.

Step 1: Identify the Files You Want to Collect

The files you want to collect as a group must all reside on the same system from which you want to serve this collection and their permissions must allow them to be readable by everyone.

Example

In this example, the files will be served from the control workstation so they must all reside there in **/usr/local**.

Step 2: Create a File Collection Directory

Create a directory in **/var/sysman/sup** that will be the name of your file collection and change its owner and group to **bin**.

Example

This file collection will be called *tools*.

```
cd /var/sysman/sup
mkdir tools
chown bin tools
chgrp bin tools
```

Step 3: Create a list File

Create a **list** file to describe the rules for including and excluding files in that directory. Refer to “supper list File Syntax, Keywords, and Operands” on page 66 for details.

Example

The easiest way to create these files is to copy them from an existing file collection directory, delete those you do not need or the system creates, and modify them as required.

```
cp -p sup.admin/* tools
cd tools
rm when last scan
```

Modify the **list**, adding these commands:

```

symlinkall
omitany *
omit ./usr/local/admin
upgrade ./usr/local
always ./usr/local/admin/control_file
execute /usr/local/refresh_fonts (./usr/local/fonts)

```

This instructs **supper** to:

- Copy links rather than resolve them
- Omit files that start with the backup notation
- Omit administrative tools
- Upgrade all files in the local tools directory
- Override an omit instruction and always update the control file
- Execute the script to refresh the font library whenever the **/usr/local/fonts** file is changed

Step 4: Add a Link to the lists File

The **lists** file in the **/var/sysman/sup** directory contains a symbolic link to the **list** file in each file collection. When you create a new file collection you must add a link to this file.

Example

Add a link pointing to the **list** file in the *tools* collection:

```
ln -s /var/sysman/sup/tools/list /var/sysman/sup/lists/tools
```

Step 5: Update the file.collections File

Edit **/var/sysman/file.collections** using your text editor. Add the name of your new file collection as either a primary or secondary file.

Example

The *tools* collection is a primary collection and requires an entry similar to **server.root**. Edit **/var/sysman/file.collections** and add these lines to the end:

```
# tools - boot/install collection to manage local tools files on the file servers
primary tools - / - / EDO power no
```

The nine fields in this entry have specific meaning to **supper**:

Field 1 Defines the file as a primary or secondary collection

Field 2 Specifies the name of the file collection

Field 3 For a primary collection, this specifies the name of the file system associated with the collection. This file system will be created when the file collection is installed. For a secondary collection, this specifies the name of its primary collection. The notation (-) indicates no file system will be created or that it is a secondary collection.

Field 4 Specifies the name of the directory by which the files are normally accessed.

Field 5 If the collection has a file system, this specifies the file system size. The notation (-) indicates no file system is associated with this collection or that it is a secondary collection and the file system is associated with its primary collection.

Field 6 Specifies the prefix directory at which the **scan** process starts.

Field 7 Specifies SUP options. In our example,

- E allows **list** file **execute** statements
- D deletes files that no are longer in the collection.
- O indicates that all files (even those older than the last update) should be evaluated.

Field 8 Specifies the system architecture. **power** indicates RS/6000 architecture.

Field 9 Specifies if this file collection can be installed on a different architecture.

Step 6: Update the .resident File

The **.resident** file contains a list identifying all the SP file collections. Edit the **.resident** file on each node and boot install server on which the file resides and add your new collection.

Example

Add the tools file collection to the **/var/sysman/sup/.resident** file:

```
sup.admin 0
user.admin 0
power_system 0
tools 0
```

Step 7: Build the scan File (optional)

The **scan** file provides you with a list of files in the collection that you can use for verification and eliminates the need for **supper** to do a directory search on each update. If your directory tree is extensive, this can save processing time on large file systems.

Note

You must keep the **scan** file current. When a **scan** file is present, the **update** command reads it as an inventory of the files in the collection and does not do the directory search. If you fail to create a new **scan** file when you add, modify, or delete files in the master collection, the file will not be current and **supper** will not upgrade the collection correctly.

Example

To create a **scan** file for the tools file collection, enter:

```
/var/sysman/supper scan tools
```

Installing a File Collection

During the initialization and customization process, the required SP file collections are first built on the control workstation and then installed on your boot/install servers and processor nodes. If you create your own file collections you have to install them yourself on each server or node.

To install a collection on a boot/install server or processor node, you must be logged in as root or issue remote commands to run **supper** there.

Step 1: Update the sup.admin File Collection

The **sup.admin** file collection contains the files that identify and control all the file collections. It contains the **file.collections** and **.resident** files. Whenever you make changes to these files, you need to update the **sup.admin** collection to distribute these updates.

You can wait for the scheduled update in the **crontabs** file or issue the command yourself on every boot/install server and node that needs the new collection.

Example

Log in to each boot/install server and node or issue the command remotely and enter:

```
/var/sysman/supper update sup.admin
```

Step 2: Run the supper install Command

Run **supper install** on each boot/install server or node that needs this collection. The **install** command assumes the dedicated server to be the default.

Example

In this example, the tools file collection will only be installed on the boot/install servers and not the processor nodes.

Log in to each boot/install server or issue the command remotely and enter:

```
/var/sysman/supper install tools
```

Step 3: Add the supper update to crontabs

You need to add the **supper update** command to the **crontabs** file on each server or node that has this collection. This insures that the new collection will be updated on schedule with the other file collections.

Example

To add the command so it runs hourly, edit **crontabs** and add this line:

```
0 * * * * /var/sysman/supper update tools 1>/dev/null 2>/dev/null
```

Until the update runs, the files will not be available at each location. If you need the files on these systems before the scheduled update, you can issue the update command on each server and processor node yourself.

Example

Log in to each boot/install server or issue the command remotely, and enter:

```
/var/sysman/supper update tools
```

Refusing Files in a File Collection

The **refuse** file allows you to customize the file collection at different locations. It is possible for you to create a file collection with one group of files and have different subsets of that group installed on the boot/install servers and the processor nodes.

You do this by creating a **refuse** file in the `/var/sysman/sup` directory on the boot/install server or processor node. This master file lists all the files in every file collection that you do not want installed or updated on that system.

Example

In this example, we want one boot/install server to have the full file collection including the test files and another to have the boot/install collection minus the test files. Log in to the boot/install server that should not have the test files and create a **refuse** file with the following entries:

```
/usr/local/draw.test
/usr/local/crunch.test
```

This instructs **supper** to never include these test files when installing or updating this file collection *at this location*.

Note

There are two kinds of **refuse** files. One is in the `/var/sysman/sup` directory and contains the files that will not be installed on that server or node. These **refuse** files are optional, are user-created, and can vary at each location. The other **refuse** file is in the file collection's directory and contains a list of all the files refused when the file collection is installed or updated. This **refuse** file is system-created and might be empty if you always install the entire collection.

Modifying the File Collection Hierarchy

The delivered file collections have a default hierarchy shown in Figure 2 on page 46. You can modify the hierarchy in several ways. One way is to make a given boot/install server function as the master server for a collection. This enables you to maintain a different set of files on a group of nodes that request their **supper** updates from that boot/install server. You do this by eliminating the logical path from the control workstation to the boot/install server for one or more boot/install collections. This is called taking the boot/install server *offline* and you do it with the **supper offline** command. Figure 5 on page 64 shows a boot/install server that is offline for the **power_system** collection.

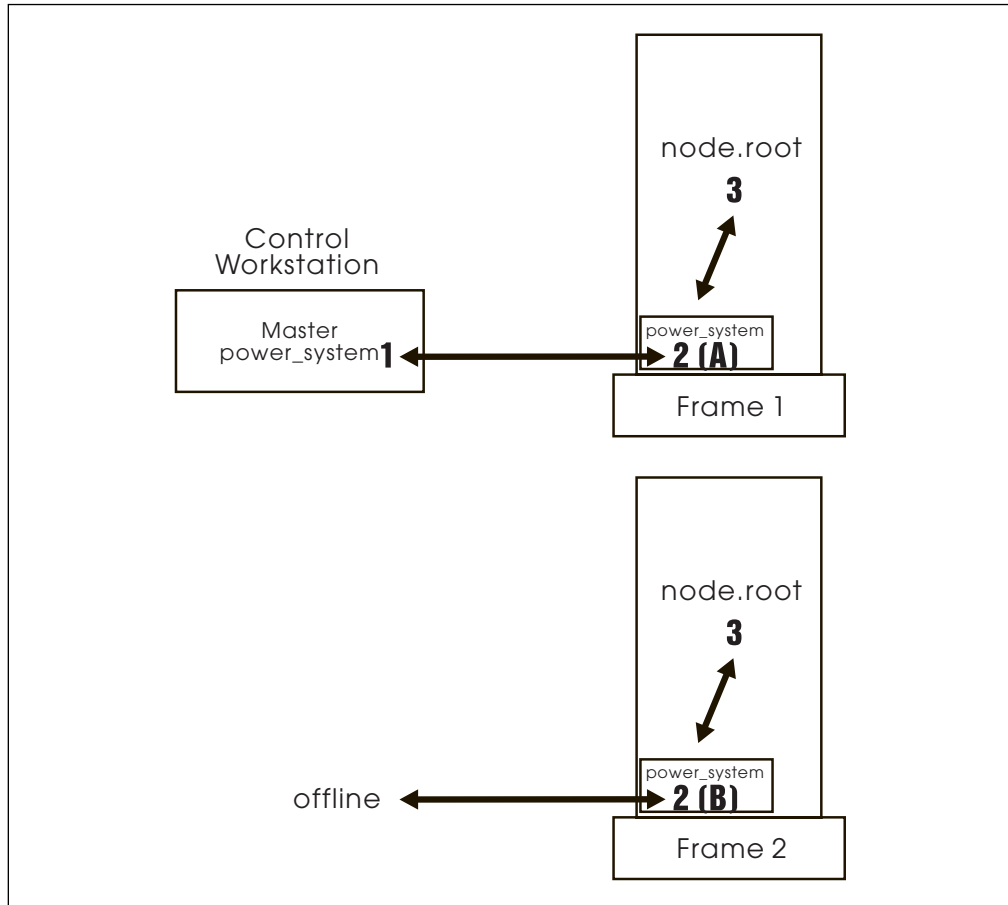


Figure 5. SP Hierarchy of an Offline Boot/install Server

In this case, Boot/install Server A and Frame 1 get any changes made on the control workstation; Boot/install Server B and Frame 2 do not. Only changes made directly to the files on Boot/install Server B affect the files on Frame 2.

Note

Whenever you change the hierarchy, you must be aware of the location of the master files for every file collection. In Figure 5, there are two master file collections for **power_system**. The master files for the **node.root** collection on Frame 2 reside on Boot/install Server B and the master files for the **node.root** collection on Frame 1 reside on the control workstation.

- If you want to change a file in the **node.root** collection for every node, you must make that change in two places, on Boot/install Server B and on the control workstation, and then run the **supper update** command on each node.
- If you want to change a file in the **node.root** collection for just one frame of nodes, you must make that change in the master file collection for that frame, either Boot/install Server B or the control workstation, and then run the **supper update** command on each node in that one frame.

The following example shows you the procedure and commands to change the default hierarchy as represented in Figure 2 on page 46. It shows you how to take

the **power_system** collection offline, change a file in the **node.root** file collection, and distribute that update to the nodes on Frame 2.

Example

This example assumes that **.profile** is in the **node.root** file collection.

1. To take the boot/install server offline from the control workstation, log in to Boot/install Server B as **root** and enter:

```
/var/sysman/supper offline power_system
```

2. On Boot/install Server B, modify the **.profile** in the **/share/power/system/3.2** directory.

Remember, any changes made on the control workstation will not be updated on Boot/install Server B because it is offline. You must make the changes on Boot/install Server B.

3. Issue remote commands to update the **node.root** file collection on each of the nodes.

```
/var/sysman/supper update node.root
```

4. Boot/install Server B no longer gets updated files from the control workstation. Unless you remove the **supper** command to update the **power_system** from the **crontabs** file on Boot/install Server B, the command will execute every hour and fail. Edit the **crontabs** file and remove:

```
0 * * * * /var/sysman/supper update power_system 1>/dev/null 2>/dev/null
```

Removing a File Collection

Warning: Never remove any of the file collections that come with the SP system!

The file collections that come with the SP system are required. If you remove them, the System Support Programs cannot run properly.

Removing a file collection does not delete it. It removes it from the place it was installed. To completely delete a file collection, you must remove it from every place it was installed. Be sure you do not need any files in a file collection before you remove it.

You can use the **supper scan** command to build a **scan** file for the file collection you are going to remove. The **scan** file shows you a list of all the files in a collection. You can check it to verify that you do not need any of these files before you remove the collection. See “Verifying File Collections Using scan” on page 51 for more information about the **scan** file.

Example

To build a **scan** file for verification, enter this command on the boot/install server where the **tools** collection is installed:

```
/var/sysman/supper scan tools
```

If you are sure you do not need any of the files, continue with the next command.

To remove the **tools** boot/install collection, on the same boot/install server enter:

```
/var/sysman/supper remove tools
```

This removes all the files in the collection only from that machine. If you also issue this command on the control workstation or the master server for the collection, you will no longer be able to propagate these files in your system.

supper list File Syntax, Keywords, and Operands

The following commands are used in the **supper list** file to define which files in a directory to include or exclude from a file collection. Refer to “Directory and Master Files” on page 46 for information about the **supper list**.

Syntax Rules

- Each command contains a keyword and a number of operands separated by spaces.
- The order of the commands does not matter and you can include blank lines between the commands.
- *filename* can be a single file, directory, or a list of files or directories separated by blanks. The names can include wild cards and metacharacters as used by **csh**, including *, ?, [...], and {...}. The value for *filename* must be specified as a relative path, such as **./etc/passwd** or **./var/sysman**.

Command Keywords and Operands

- **upgrade** *filename*

The specified files or directories are included in the list of files to be upgraded unless they are also specified by an **omit** or **omitany** command or are in the **refuse** file. If a directory name is given, it recursively includes all subdirectories and files within that directory.

- **always** *filename*

The specified files or directories are included in the list of files to be upgraded, regardless of the **omit** or **omitany** commands. They will not be included if they are in the **refuse** file.

- **omit** *filename*

The specified files or directories will be excluded from the list of files to be upgraded. For example, by specifying **upgrade** **./usr/vision** and **omit** **./usr/vision/exp**, the generated list of files would include all subdirectories and files of **./usr/vision** except **./usr/vision/exp** and its subdirectories and files.

- **omitany** *pattern*

The specified patterns are compared against the files specified in the **upgrade** entry. If a pattern matches, the file is omitted. The **omitany** command currently supports all wild card patterns except {...}. Also, the pattern must match the entire file name, so a leading */, or a trailing /*, might be necessary in the pattern.

- **backup** *filename*

The specified files are marked for backup. If they are upgraded, backup copies are created. Directories cannot be specified and no recursive file name construction is performed; you must specify the names of the specific files to be backed up before upgrading.

- **noaccount** *filename*

The accounting information (permissions and time stamps) of the specified files is not preserved.

- **symlink** *filename*

The specified files are to be treated as symbolic links. They are transferred as links and not followed. By default, symbolic links are followed. **symlinkall** transfers all symbolic links.

- **execute** *exec-command (filename)*

The command specified is executed on the client process whenever any of the files listed in parentheses are upgraded. A special token, %s, might be specified in the command and is replaced by the name of the file that was upgraded. For example, if you use `execute ranlib %s (libc.a)`, whenever `libc.a` is upgraded, the client machine will execute `ranlib libc.a..`

- **include** *listfile*

The specified *listfile* is read at this point. This is useful when one collection subsumes other collections; the larger collection can simply specify the files for the smaller collections contained within it.

Related Information

The *PSSP: Command and Technical Reference* presents the full **supper** command syntax with all the parameters and options. It also explains how to use **supper** in batch or interactive mode.

You can find additional information about **supper** and SUP in their manual pages.

Chapter 5. Managing User Accounts

The SP system supplies user administration commands, **spmkuser**, **spchuser**, **sprpmuser**, and **spluser**. This chapter shows you how to add, change, delete, or list user accounts using both the SP commands and the SMIT interface.

Task information included in this chapter includes:

- Understanding SP user management
- Adding user accounts
- Changing user accounts
- Listing a user account
- Removing user accounts
- Changing passwords
- Controlling user login

Understanding SP User Management

The SP system provides the ability for you to manage your user accounts by adding new users, deleting existing users, or changing the user account information from a single point of control. SP User Management ensures that users have the same account, home directory, and environment across all the nodes in the SP system. SP User Management is optional, and it is designed to fit into your existing user management system.

The objective of any user management system is to ensure that the user account files, namely **/etc/passwd**, **/etc/group**, **/etc/security/passwd**, and **/etc/security/group** and optional password index files are consistent across all of your nodes. You can accomplish this by using Network Information Service (NIS), by distributing these files using file collections, or by some other means. The password index files **/etc/passwd.nm.idx**, **/etc/passwd.id.idx**, and **/etc/security/passwd.idx** are used for login performance.

If you use NIS, the SP system references the NIS maps that are located on the NIS master machine. The NIS maps provide each user with a single image of their user account information in the NIS domain. This information includes the user's login name, password, group information, default shell, and the name of the host where the home directory resides. If you use SP automounter support, the system also maintains automounter maps of mount points for the home directories. As the system administrator, you need to create and maintain the NIS environment on the NIS master machine in the **/var/yp** directory. The SP User Management commands do not interface with the NIS environment. "Brief Review of NIS" on page 70 provides a more detailed discussion of using NIS in an SP environment. For a more complete discussion, refer to the appropriate NIS documentation or the book *IBM International Technical Support Centers RS/6000 SP System Management: Easy, Lean and Mean*.

If you are not using NIS, the SP User Management commands use the **user.admin** file collection to propagate the user administration files to the nodes of the system. You may wish to use this file collection to keep other user management related files consistent across the nodes. For more information refer to the "user.admin Collection" on page 45.

In NIS, users can change their own passwords, as needed, while logged in any SP node. In non-NIS environments users must log into the host where the password file resides. The full pathname for the password file is set with the **passwd_file** parameter of the **spsitenv** command, or through SMIT panels during installation. Administrators can change passwords for any user account from the NIS master.

After you complete the installation and customization procedures, you establish an initial set of users from the existing user management files in your network. The SP system provides the ability for you to manage your user accounts by adding new users, deleting existing users, or changing the user account information.

Brief Review of NIS

This section briefly describes the use of the Network Information Service (NIS). For a more complete discussion, refer to appropriate NIS documentation or the *IBM International Technical Support Centers RS/6000 SP System Management: Easy, Lean and Mean (GG24–2563)*. Although an SP is a machine containing multiple RISC System/6000 nodes, you do not want to maintain an SP as multiple computers, but as one system. NIS is one of the tools that can make the daily operations of an SP simple and easy.

NIS is a distributed system which contains system configuration files. By using NIS, these files will look the same throughout a network, or in this case, throughout your SP machine. NFS and NIS are packaged together. Since the SP install image includes NFS, NIS comes along as well. The most commonly used implementations of NIS are based upon the distribution maps containing the information from the `/etc/hosts` file and their user-related files: `/etc/passwd`, `/etc/group`, and `/etc/security/passwd`.

NIS allows a system administrator to maintain system configuration files in one place. These files need only be changed once, then propagated to the other nodes. From a user's point of view, the password and user credentials are the same throughout the network. This means that the user only needs to maintain one password. When the user's home directory is maintained on one machine and made available through NFS, the user's environment is also easier to maintain.

From an SP point of view, an NIS solution removes the SP restriction of changing user's passwords on the control workstation. When you would use File Collections for system configuration file distribution, users have to change their password on the control workstation. When using NIS, you can control user password management across the SP, from any given node.

Note: When configuring User Management to true in the Site Environment, the configuration process renames the `/usr/bin/passwd` executable to `/usr/bin/passwd.orig`. In its place, a new `/usr/bin/passwd` file is created requesting you to change your password on the control workstation, when executing `/usr/bin/passwd`. Renaming `/usr/bin/passwd.orig` to `/usr/bin/passwd` would bring back the original executable.

Using NIS and File Collections

The SP is provided with another tool to ensure that system configuration files look the same throughout your SP network. This tool is called File Collection Management. When configuring the SDR, you are asked if you want to use this facility. When answered affirmatively, the control workstation configures a mechanism for you that will periodically update the system configuration files (you specify the interval). The files included in that configuration are:

- All files in the directory **/share/power/system/3.2**
- Some of the supper files
- The Amd files
- The user administration files (**/etc/group**, **/etc/passwd**, **/etc/security/group**, and **/etc/security/passwd**)

This setting also causes the replacement of the `/usr/bin/passwd` command by a dummy command, indicating that you must update your user password on the control workstation. In terms of user administration, the File Collection Management system is an alternative to using NIS for users who are not familiar with NIS or do not want to use it.

However, even though file collections are implemented, NIS can still be used in parallel for user administration. When doing so, the File Collections should exclude the user administration files and the `/usr/bin/passwd` command should be restored (the original `passwd` command is stored as `/usr/bin/passwd.orig`).

Using NIS Netgroups and Interactive Session Support

NIS provides the ability to hierarchically group users and hosts. These groups can be used as classifications in the `/etc/passwd` and `/etc/exports` files to define login permissions to to mount remote file systems. The file that contains the definitions is stored in the `/etc/netgroup` file.

Interactive Session Support (ISS) does the same where assigning a login host is concerned. Based on certain criteria, ISS determines the node to which the next login request should be directed. ISS can work together with NIS.

Setting UP NIS

Using the SMIT panels, you can set up NIS, manage it and control the NIS daemons. The NIS-associated SMIT panels can be found by selecting the following from SMIT:

SELECT Communications Applications and Services

SELECT NFS

SELECT Network Information Service (NIS)

- The Network Information Service (NIS) menu appears.

Each node in a NIS environment must be a member of a domain and each node can only belong to one domain at a time. So, you must change or set the domain name on each node. You can use SMIT to do this or use the `/usr/bin/domainname <cluster-name>` command. This will take effect immediately, or at system reboot, or at both. This configuration is set in the `/etc/rc.nfs` file, where the comment characters are removed from the lines to set a NIS domain.

Configuring a Master Server: In your planning, you must decide whether (1) you will have slave servers and (2) you will allow users to change their passwords anywhere in the network. With these decisions made, you can now configure the master server using SMIT:

SELECT Configure / Modify NIS

SELECT Configure this Host as a NIS Master Server

Specify the hosts that will be configured as slave server, and, if any, specify yes to start the yupdated. In most cases, the yppasswd is started as well. These configuration settings can also be found in the /etc/rc.nfs file.

By default, the NIS master server maintains the following files:

- /etc/ethers
- /etc/group
- /etc/hosts
- /etc/netgroup
- /etc/networks
- /etc/passwd
- /etc/protocols
- /etc/publickey
- /etc/rpc
- /etc/security/group
- /etc/security/passwd
- /etc/services

Any changes to these files must be propagated to clients and slave servers using SMIT:

SELECT Manage NIS Maps

SELECT Build / Rebuild Maps for this Master Server

Either specify a particular NIS map by entering the name representing the filename or leave the default value of all then press Enter. You can also do this manually by changing to the directory /etc/yp and entering the command make all or make <map-name>. This propagates the maps to all NIS clients and transfers all maps to the slave servers.

Configuring a Slave Server: A slave server is the same as the master server except that it is a read-only server. Therefore, it cannot update any NIS maps. Making a slave server implies that all NIS maps will be physically present on the node configured as the slave server. As with a master server, the NIS map files on a slave server can be found in /etc/yp/<domainname>. To configure a slave server using SMIT:

SELECT Configure / Modify NIS

SELECT Configure this Host as a NIS Slave Server

Configuring a slaver server starts the ypbind daemon which searches for a server in the network running ypserv. Shortly afterwards, the ypserv daemon of the slave server itself will start.

In a lot of situations, the slave server must also be able to receive and serve login requests. This that is the case, the slave server must also be configured as an NIS client. Refer to the next section for this information.

Configuring an NIS Client: An NIS client retrieves its information from the first server it contacts. The process responsible for establishing the contact with a server is ypbind. To set up an NIS client using SMIT:

SELECT Configure / Modify NIS

SELECT Configure this Host as a NIS Client

Managing NIS

As previously discussed, setting up NIS is a fairly simple task. Most of the work is related to the design on your domain, the naming conventions you will use, and deciding which nodes will be NIS clients or slave servers. Once these definitions are clear, configuring NIS is easy. There are, however, a few points to keep in mind.

- Configure NIS in stages. Make sure that you define your master server first, then your slave servers, and finally, the clients. Also, when going over routers, first configure the master segment, then the slave servers in the other segments and finish by configuring the clients.
- Any changes to the system configuration files managed by NIS will only take effect after an enforced update of the NIS databases. It is not automatic, unless you add an update command to cron.

Use of Netgroups: By default, NIS is not provided with netgroups. Netgroups are logically-bundled nodes or users that can be granted or denied access to certain AIX services. The files used to define a netgroup is /etc/netgroup. The /etc/netgroup syntax is (-,user,domain) or (host,-,domain) where domain can be blank (commas are required). This results in having host-netgroups and user-netgroups.

Host-netgroups can be used, for example, in the /etc/exports file, where access can be granted to a netgroup.

User-netgroups may be used in the /etc/passwd file where you can specify that only netgroup team can log in.

Domain Name Server (DNS) and NIS: Since NIS replicates the /etc/hosts file, host resolution may also be done using NIS. There are four ways AIX can do host resolution:

- No DNS, No NIS
In this case, the /etc/hosts file must provide the answer to a request of a node in the network.
- No DNS, but NIS
Only NIS host maps are referenced. If NIS does not have the answer, the /etc/hosts file is not referenced.

- DNS, No NIS

For using DNS, the `/etc/resolv.conf` file must be configured. A request to a foreign host invokes TCP/IP to resolve the host on the nameserver. If the nameserver cannot resolve the hostname, TCP/IP will refer to the `/etc/hosts` file, possibly with a short delay.

- DNS and NIS

As with the previous point, if DNS cannot resolve the host, NIS is referred to for hostname or hostaddress resolution. If NIS cannot resolve the request, the `/etc/hosts` file is not referenced.

SP User Management Options

Using the SP user management commands, you can add and delete users, change account information, and set defaults for your users' home directories. Specify the user management options you wish to use in your site environment during the installation process, or change them later, either through SMIT panels or by using the `spsitenv` command. The *PSSP: Installation and Migration Guide* contains detailed instructions for entering site environment information.

From SMIT

ENTER `smit enter_data`

- The Enter Database Information menu appears.

SELECT `Site Environment Information`

- The Site Environment Information menu appears.

The following options, displayed in the SMIT Panel for Site Environment Information, pertain to user management:

User Administration Interface

Whether or not you want to use SP User Management to add, delete, and change user account information.

Password File

The path of your password file. The user commands `add`, `modify`, and `delete` entries in this file. If you modify the default path of `/etc/passwd` and you are using NIS, ensure that you also modify your NIS makefile to build the password map from the new password file.

Password File Server Hostname

The hostname of the machine where your master password file resides. The initial value is the control workstation. The master password file location cannot be one of the nodes in the SP system.

Home Directory Server Hostname

The name or IP address of the machine where the user home directories reside.

You can specify a default host for users' home directories. If you use the SP automounter support the user management commands will use this host when building automounter maps to mount and link the home directories. If you do not specify a default, the initial value of the control workstation is assumed.

You can override the value in Hostname Home Directory Server when adding or modifying a user account with the **spmkuser** and **spchuser** commands.

If the users' home directories are served by a machine within the SP system and you follow the authentication setup described in Chapter 12, "Security Features of the SP System" on page 131, then you need not do anything more to have the SP user management commands run correctly.

However, if the home directory server is outside the SP system, there are two cases:

1. The machine that serves the home directory is in the same authentication realm as the SP system. In this case, the administrator's principal name must be added to the **.klogin** file of that machine.
2. The machine that serves the home directory is outside the realm and nonauthenticated. In this case, add a line to the **.rhosts** of the machine with the following data:
 - The hostname where the commands run
 - The user name **root**

In this case, when the SP user management commands are run, informational messages are displayed to show that authenticated commands could not be run. However, the user management task will be performed.

Home Directory Path

This is where you specify the default path of users' home directories. The default is **/home/control_workstation_name/**.

Enter another value to set another path as the default for your site. You can override the default path with the **home** attribute, **spmkuser**, and **spchuser** commands.

Any host that is serving user directories must have its **/etc/exports** file modified to grant access to all hosts that need to mount these directories. Export the file systems using **exportfs -a** after modifying the **/etc/exports** file.

SP User Management Commands

The SP User Management commands reside in the **/usr/lpp/spp/bin** directory on the control workstation. If you wish to use these commands in an NIS environment and your control workstation is not your NIS master, you must mount these commands on the NIS master where you will be executing them. (Your NIS master must be an RS/6000.)

You can add, change, delete, or list user accounts using either the SP commands or, optionally, the SMIT interface. For example, you can add new users to the SP system in any of the following ways:

- Enter the **spmkuser** command with the appropriate attributes and values.
- Enter **smit spmkuser**.
- Enter **smit**, select *Security & Users*, and then select *RS/6000 SP Users*.

- Enter **smit**, select *RS/6000 SP System Administration*, and then select *RS/6000 SP Users*.

The examples in this section show the first two methods. You can find the complete syntax for the commands in the *PSSP: Command and Technical Reference*.

Adding User Accounts

To add new users to the SP system with the SMIT interface:

TYPE **smit spmkuser**

- The Add a User window appears.

Only the *User name* is required. You can accept the defaults or leave the other fields blank. The defaults for primary group, secondary groups, and the initial program are in **/usr/lpp/ssp/bin/spmkuser.default**. The default for the home directory is stored in the System Data Repository. The following table shows how the fields in this window relate to the parameters in the **spmkuser** command.

SMIT Field	Command Parameter	Description
User name	Name	User login name
User ID	id	ID of the user
LOGIN user?	login	Indicates whether the user can login to the system with the login command. This option does not change the /etc/security/user file but instead alters the user's password field in /etc/security/passwd .
Primary GROUP	pgrp	Principle group of the user
Secondary GROUPS	groups	The groups to which the user belongs
HOME Directory	home	Host name of the file server where the home directory resides and the full path name of the directory separated by a colon
Initial PROGRAM	shell	Shell program run when the user logs in
User Information	gecos	General information about the user

When you add a new user, the system generates a random password for the user and stores it in **/usr/lpp/ssp/config/admin/newpass.log**. The **root** user has read and write permission to this file. It is the administrator's responsibility to communicate this password to the new user and periodically delete the contents of this file.

Example

The following example shows how to enter the information for a user named **joe** using SMIT:

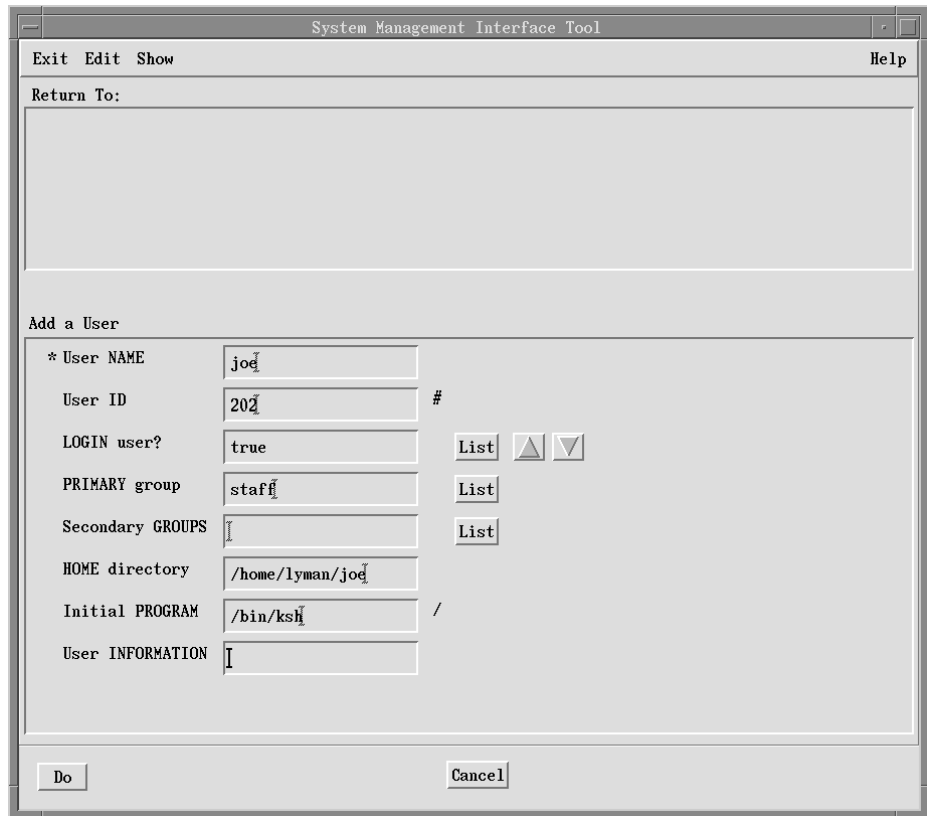


Figure 6. Example of Adding an SP User

TYPE The appropriate information in each field

PRESS **Do** to store the data.

To store the same information using the **spmkuser** command, enter:

```
spmkuser id=202 pgrp=staff home=lyman:/home/lyman shell=/bin/ksh joe
```

Changing User Accounts

To change user information with the SMIT interface:

TYPE **smit spchuser**

TYPE The name of the user whose information you want to change

- The Change/Show Characteristics window appears.

To change information, type the new values in the appropriate fields. The only field you cannot change is *User name*. You can scroll to see the information in the home directory field.

Example

To change information using SMIT:

TYPE The new values

PRESS **Do** to store the data.

To change the home directory for **joe** using the **spchuser** command, enter:

```
spchuser home=lyman:/home/lyman/joe joe
```

Removing User Accounts

To remove an existing user account from the SP system with the SMIT interface:

TYPE **smit sprmuser**

TYPE The name of the user whose information you want to remove

- The Remove a User window appears.

You can choose to remove the user's home directory and the authentication data when **/etc/security/passwd** is utilized.

Example

To remove the **joe** user account with the **spruser** command but leave the authentication data and home directory intact, enter:

```
spruser joe
```

Listing a User Account

You can display the information for a user with the SMIT interface using **smit spchuser**.

TYPE **smit spchuser**

TYPE The name of the user whose information you want to view

- The SP Change/Show Characteristics window appears.

Be sure to cancel after viewing the information.

You can also list user information using the **splsuser** command. For example to list information for a user named **ken** in colon-separated records, enter:

```
splsuser -c ken
```

Changing Passwords

Your method of account management determines which steps are necessary to change passwords. The procedure for NIS is different than for a system with file collections. If neither NIS nor file collections is used, you will probably use **rdist**.

Changing Passwords on an NIS System

NIS users can modify their passwords using either the **yppasswd** or **passwd** command. They must run these commands within the NIS domain that serves their user account.

Note

If you maintain an NIS password file outside of an NIS environment, you should continue your current process for changing passwords and disable the **yppasswdd** daemon on the NIS master:

1. Enter **smit chmaster**.
2. Set *START the yppasswdd daemon?* to **no**.
3. Set *CHANGE server now, at system restart or both* to **both**.

As a result, whenever users try to change passwords, they receive a message stating that the master is not running.

Changing Passwords for a File Collection System

If you use file collections rather than NIS, you must change the password in the master password file. Although this file may not be maintained on the control workstation, any changes to passwords must be made known to the control workstation.

1. Change password on host where master password file resides.
2. If the master password server is not the control workstation, copy the files **/etc/passwd** and **/etc/security/passwd** from the password server machine to the control workstation.
3. New password will be propagated throughout SP system within one hour.

Restricting the Use of the Control Workstation for General Users

If you use file collections on your system, users must be able to log in to the machine designated as the password file server to change their passwords. If you use the control workstation for the password server, you must allow users to log in to it to change their passwords. However, you may wish to have only administrative tasks performed on the control workstation, and therefore, do not want general users to be able to do anything other than change their passwords.

A script is provided that allows full login on the control workstation to the users you designate and restricts full login to all others. Users not designated can only log in to change their passwords and then are logged out. The script can be incorporated into the profile that is run during every login to the control workstation.

The script is located in **/usr/lpp/ssp/config/admin/cw_restrict_login**. To use the script, take the following steps:

1. Edit **/usr/lpp/ssp/config/admin/cw_allowed**

The **cw_allowed** file defines to the **cw_restrict_login** script which user names are allowed to fully log in to the control workstation. **cw_allowed** is shipped containing three administrators' user names to show how the file should be formatted. These sample names should be removed from the file.

Start each user name in the left most column and place one user name per line. Place no comments in the file. You do not need to list the root user in the file; the script is written to allow the root user to fully log in to the control workstation.

2. Integrate the restricted login script into the control workstation's login process by adding the following lines to the beginning (or at the most appropriate place) of the control workstation's **/etc/profile** file:

```
# Allow general users to login to control workstation to only change
# their password then log them out.
/usr/lpp/ssp/config/admin/cw_restrict_login
```

If you are using the AIX Common Desktop Environment (CDE) on the control workstation, you will also need to make a link from the restricted login script for CDE to the appropriate CDE directory:

```
ln -s /usr/lpp/ssp/config/admin/cde_cw_restrict_login \
/etc/dt/config/Xsession.d/cde_cw_restrict_login
```

3. Verify that you have set up correctly:
 - a. Log in to the control workstation as root and make sure you can fully log in.
 - b. Log in as a user that should be allowed (listed in **cw_allowed**) to log in to the control workstation. Make sure this user can fully log in.
 - c. Log in as a user that is not allowed (not listed in **cw_allowed**) to fully log in to the control workstation. Make sure this user can change his password and then is logged out.

At any time you wish to remove this restrictive login, you can do so simply by commenting out (adding a # at the beginning of the script execution line) or by deleting the lines you added from the control workstation's **/etc/profile** file.

If you are using CDE and wish to remove its restrictive login, simply remove the **cde_cw_restrict_login** link you created in the **/etc/dt/config/Xsession.d** directory.

Changing Passwords on Systems without NIS or File Collection

If neither NIS nor file collection is used, you must distribute these updated password files across the SP system after you change the password.

1. **/etc/passwd**
2. **/etc/security/passwd**

Changing Passwords when SP User Administration Interface is Configured

If you have configured the SP USER ADMINISTRATION INTERFACE to **true** and you are not using NIS, the following base AIX commands are linked to SP commands on the nodes:

- **/bin/chfn** → **/usr/lpp/ssp/config/sp_chfn**
- **/bin/chsh** → **/usr/lpp/ssp/config/sp_chsh**
- **/bin/passwd** → **/usr/lpp/ssp/config/sp_passwd**

The original AIX files have been saved in their original location with extension **.aix** added. For example, the original AIX **passwd** command is in **/bin/passwd.aix**.

The linked **sp_** commands inform users what machine to log in to when, or if, they need to change password, gecos, or shell information.

Important Notes

1. Under certain circumstances, the user is prompted to change passwords during the login process. If this occurs, the user should log in to the machine serving the password file and make the change from there. If this is not done, the password change will be only local to the node; in addition, if File Collections is being used, the password change will be only temporary. To avoid this problem, communicate to users what machine to log in to when a password change is required during the login process. IBM suggests placing this information in the message of the day (MOTD).
2. If you de-install an AIX modification level after upgrading or rejecting an AIX PTF, the **lppchk** command will get run automatically. This command will show that the system is "broken" when it discovers the mismatch in file information for the AIX linked commands. This problem can be alleviated by following the proper procedure documented in the *PSSP Installation and Migration Guide*, section on Recovering from a Node Migration Failure, particularly steps 3 and 6.

Controlling User Login

Login Control is used to dynamically prevent interactive login of users on a node basis. Preventing interactive login of users on nodes running parallel jobs is desirable for performance purposes.

You may want to use Login Control to temporarily restrict all but a few users on a node for a specific purpose.

You can issue commands to temporarily restrict or unrestrict users' interactive access to nodes running parallel jobs. The Login Control utility will not prevent LoadLeveler from running jobs submitted by blocked users because LoadLeveler logs in as **root** and then switches to the user. Root is never blocked on a node.

Understanding Login Control

User login completes successfully under AIX after checking attributes in the **/etc/security/user** file. The attributes are *login* and *rlogin*. If these values are set to true, the user is allowed to log in. Conversely, if these values are set to false, the user cannot log in. Login Control updates this file dynamically for a user or several users when a request is received.

The **spacs_cntrl** command takes four state keywords that manage user access to a parallel node:

1. **block**
2. **unblock**
3. **allow**
4. **deny**

Block and Unblock

These keywords are used by a System Administrator to dynamically restrict and unrestrict access to a node for a user or a group of users. These keywords are intended to set the user to a known state (either restricted or unrestricted) in the **/etc/security/user** file and in the **spacs_data** file.

Allow and Deny

These keywords can be used by a job submission system on a transactional basis to update the user state. Multiple usage of **spacs_cntrl** with one of these keywords for a user results in the request state being updated in **spacs_data**.

A user can be in one of four states when these keywords are used:

1. Allowed interactive login
2. Denied interactive login
3. Allowed login after a deny request
4. Denied login after an allow request

State 3 can occur when other jobs by this user are still running. A deny request removes one allow request: in this case, the first allow from the first job.

State 4 can occur when more than one deny request is issued for a user. A deny request removes one allow request.

A special file, **spacs_data**, is created when a user access change request is made more than once by a job submissions system using the keywords **allow** or **deny**. It can be said that this file holds the state of requests for a user. After the initial allow or deny request has been processed in **/etc/security/user**, the next allow or deny request sets a request count for the user. The same number of subsequent opposite requests will check the count, decrement it and remove the user, if present, from **spacs_data**. The use of the block and unblock keywords by the System Administrator updates **spacs_data**, clearing all multiple job submission system allow/deny requests for a user before changing the state in **/etc/security/user**. These keywords are intended to set the user to a known state (either restricted or unrestricted) in the **/etc/security/user** file and in the **spacs_data** file. The **spacs_data** file is owned by root and is located in the **/var/spacs** directory.

Login Control recognizes requests made with the **spacs_cntrl** command. (See the *PSSP: Command and Technical Reference* for syntax.) This command is run on each parallel node and can be used with **dsh** for this purpose.

Login Control uses a lock on the **spacs_cntrl.lock** file to prevent more than one instance of Login Control from updating the files. This file is created the first time the **spacs_cntrl** command is executed. This file also resides in the **/var/spacs** directory.

Types of Access Disallowed

When **spacs_cntrl** block or deny is used, all of the following types of interactive access are disallowed:

- **login**
- **rlogin**
- **rsh**
- **rcp**

- **rexec**
- **rsh**
- **rcp**

Disallowing ftp on a Node

As system administrator, you can disallow users from using **ftp** on a node by placing users in the **/etc/ftpusers** file. This file can be kept in a file collection and distributed to the appropriate nodes. The **ftp** daemon can also be disabled to prevent all ftp access.

You can also use the **ruser** command to manage **ftp** usage by a user on a node:

```
ruser -a -f username
```

Using this command with **dsh** to the appropriate nodes allows manipulation of the **/etc/ftpusers** file on a per user basis.

Using the Login Control Sample File

A sample Perl script called **block_usr_sample** is located in the **/usr/lpp/spp/samples** directory. This script creates a file of user IDs, **/tmp/usr.input**, that can be used with **spacs_cntrl -f** to restrict user access. It serves as a model for formatting other files for use with the **spacs_cntrl** command:

```
userid1  
userid2  
userid3
```

Once you are familiar with Login Control, **spacs_cntrl** can be run within this script by removing the comments surrounding the command. A copy of this file is located in Appendix C, "Sample Files" on page 443.

Before using this script you must edit the **threshold UID (\$uidstart)** in the script to be large enough so that system IDs and SP system IDs are not restricted. These IDs include **bin**, **admin**, **lpd**, **supman**, and **prtld**. The default threshold UID is set at 125. The **block_usr_sample** script can be added to the **script.cust** file or to the node's **inittab** file.

Adding the script to run from the **script.cust** file will create the **/tmp/user.input** file and run **spacs_cntrl block** when you install the node to initially restrict interactive access for all users. Users would remain in this state across a reboot unless specifically unrestricted by the System Administrator or by the job submission system. New users that are added to the system would not be included until the **block_usr_sample** script was run again at some later time.

Adding the **block_usr_sample** script to the **inittab** file will run this script at node boot time. This will include any new users on the node but may also increase the time to boot the node. This script must be added **AFTER** the **rc.sp** script in the **inittab** file. **rc.sp** removes the login control lock file, **spacs_cntrl.lock**, if present, when booting a node to ensure login control will run if a node is rebooted after a crash.

Using Login Control

There are many ways you can use Login Control on your SP system. You can restrict all or a subset of users on a parallel node to prevent the users from gaining interactive access. The restriction remains in effect until you specifically use the unblock keyword to unrestrict access for the restricted users.

This enables you to set aside a parallel node for the use of select users for interactive and parallel jobs. You may list the users on the command line or use a file containing the user names in a column. Note that this file must reside on the parallel node in order to be accessible to the login control command.

Restricting a User on a Single Node

To restrict user Betty on a particular node, on that node issue:

```
spacs_cntrl block betty
```

Restricting Users on Multiple Nodes

1. Run the **block_usr_sample** file after adjusting threshold UID.
2. Send file to the node(s). Note this example would require **rsh** privileges on the nodes.

```
dsh -a rcp root@mynode:/tmp/usr.input /tmp/usr.input
```

3. Issue login command to restrict interactive access for users to all the nodes.

```
dsh -a spacs_cntrl -f /tmp/usr.input block
```

Maintaining Login Control

The following information is important for Login Control to function properly.

Since the **/etc/security/user** file is used by Login Control for state information, and by the AIX system for restricting and unrestricting login for users, this file is machine dependent and should not be overwritten. The **/etc/security/user** file should not be distributed through file collections or any other mechanism throughout the SP system.

The **spacs_cntrl** command may generate a large log file, depending on which flags you specify. The default for trimming the log file is 400 lines and is done every night through a **cron** job. If you wish to save the log for some reason, you can rename the file and a new log file will be created.

Usage Notes

You must issue the **spacs_cntrl** command to block new users added to a node if you have previously blocked all users on that node. Users denied login will see the standard message, "You have entered an invalid login name or password."

Each time the log is opened, a timestamp is written. This may or may not be followed by messages depending on the options chosen on the command line. The following **spacs.log** example shows the output when using the **-d** and **-l** flags on the **spacs_cntrl** command line.

Sample Log

```
"Tue Apr 26 10:11:47 EDT 1994"
spacs_cntrl: Command Invocation: spacs_cntrl -d -l deny nancy.
spacs_cntrl: CPROFILE called for /etc/security/user and /etc/security/user.17008.
spacs_cntrl: ACS_FILE/lock called for /var/spacs/spacs_data.lck.
spacs_cntrl: List of users in deny list: .
spacs_cntrl: List of users in allow list: ,mroz.
spacs_cntrl: Deny processing.
spacs_cntrl: List of users in send list: nancy.
spacs_cntrl: MUSRFILE called.
spacs_user: Called with parameters: /usr/lpp/ssp/bin/spacs_user -l -d deny nancy.
spacs_user: Attribute for user nancy changed to requested value.
spacs_cntrl: List of users in return list: nancy:0.
spacs_cntrl: CPROFILE called for /var/spacs/spacs_data and /var/spacs/spacs_data.17008.
spacs_cntrl: ACS_FILE/unlock called for /var/spacs/spacs_data.lck.
"Tue Apr 26 10:11:52 EDT 1994"
spacs_cntrl: Command Invocation: spacs_cntrl -d -l deny pepper.
spacs_cntrl: CPROFILE called for /etc/security/user and /etc/security/user.20102.
spacs_cntrl: ACS_FILE/lock called for /var/spacs/spacs_data.lck.
spacs_cntrl: List of users in deny list: .
spacs_cntrl: List of users in allow list: ,mroz.
spacs_cntrl: Deny processing.
spacs_cntrl: List of users in send list: pepper.
spacs_cntrl: MUSRFILE called.
spacs_user: Called with parameters: /usr/lpp/ssp/bin/spacs_user -l -d deny pepper.
spacs_user: Attribute for user pepper changed to requested value.
spacs_cntrl: List of users in return list: pepper:0.
spacs_cntrl: CPROFILE called for /var/spacs/spacs_data and /var/spacs/spacs_data.20102.
spacs_cntrl: ACS_FILE/unlock called for /var/spacs/spacs_data.lck.
```


Chapter 6. Managing SP Resources

As of PSSP 3.1, the Resource Manager function previously available with PSSP has been moved to the IBM LoadLeveler product. For more information, see the book *LoadLeveler: Using and Administering*.

The `ssp.jm` fileset contains the commands and libraries necessary on the control workstation to support system partitions that have back levels of PSSP. No Resource Manager daemons are available.

The following table lists functions that were previously part of the PSSP Resource Manager and are now part of the LoadLeveler product.

Resource Manager Function	LoadLeveler Function
Support for pools	The pool_list keyword in the machine stanza; the Pool specification on the requirements statement in the job command file.
Specifying batch, interactive, or general use for nodes	The machine_mode keyword in the machine stanza.
Enabling SP exclusive use accounting for parallel jobs	The sp_exclude_acct keyword in the machine stanza.
Controlling user logins	LoadLeveler does not directly interact with the Login Control Facility. LoadLeveler logs into nodes as root and switches to the user's ID. root is never blocked on a node.
Providing node and adapter information for SP nodes	The llextrSDR extracts information from the SDR that you can use in administration file stanzas.
Requesting dedicated use of nodes	The node_usage keyword in the job command file.
Requesting dedicated use of adapters	The <i>usage</i> operand on the network keyword in the job command file.
Displaying job information with the jm_status -j command	The llq command.
Displaying pool information with the jm_status -P command	The llstatus command.

The following table lists switch table management functions for user space parallel jobs that were previously part of the PSSP Resource Manager. They have been replaced by a set of application program interfaces (APIs) in PSSP called the Job Switch Resource Table (JSRT) Services. The JSRT Services are used to load, unload, clean, and query Job Switch Resource Tables and are available to you also for application development. These tables are needed by parallel jobs in order to run user space tasks over the SP Switch. See "Switch Communication Using User Space Message Passing" on page 205 for more information. The JSRT Services API is installed using the **ssp.st** fileset that is part of the PSSP installation image.

Table 4. New Job Switch Resource Table Functions Previously Part of the Resource Manager

Resource Manager Function	Job Switch Resource Table
Loading the job switch resource table for user space parallel jobs.	The swtbl_load_table API is now called by LoadLeveler to load the job switch resource tables.
Monitoring switch_responds for the current state of the switch.	The swtbl_query_adapter API is now called by LoadLeveler to get the current state of the switch.

Chapter 7. Managing Time Synchronization

There are several ways you may currently be handling synchronizing time-of-day clocks on your control workstation and processor nodes. You may already be using Network Time Protocol (NTP) either locally or through the Internet, or you may be using some other time service software. It is also possible that you have no established method for synchronizing the system clocks in your computing environment.

The SP system gives you several options for time-keeping:

- If you have an established NTP time server, you can use it to synchronize and manage time on the SP system.
- You can choose an NTP time server from the Internet.
- You can run NTP locally on the SP system to generate a consensus time.
- You can choose not to use NTP at all, relying on other methods at your site.

The SP machines do not have system batteries. If you choose not to use NTP, you must have another way to manage clock synchronization.

Managing NTP

There are three attributes to set for NTP. *ntp_version* defaults to a value of **3**, the version shipped with the SP System Support Programs. If your installation is using an earlier version of NTP, change this value. The other two attributes are described in Table 5.

	Worksheet Entries to Be Filled In	
To do this....	<i>ntp_config</i>	<i>ntp_server</i>
Use your site's existing NTP time server to synchronize the SP system clocks.	timemaster	<i>hostname</i> of your current NTP time server
Use an NTP time service from the Internet to synchronize the SP system clocks.	Internet	<i>hostnames</i> of time servers on the Internet*
Run NTP locally on the SP to generate a consensus time.	consensus (default)	
Do not use NTP on the SP; instead, use some other method to synchronize system clocks.	none	
Note: <ul style="list-style-type: none">• Change default attribute values to suit your environment.• Blank entries imply that you make no substitutions for these values.• *Reference /usr/lpp/ssp/ssp.public.README for information on Internet time servers.		

When you choose any of the options that involve using NTP, the installation scripts determine the IP address of the host specified as **ntp_server** and creates the **/etc/ntp.conf** file on each of the control workstation, boot/install server, and processor nodes. This file indicates which hosts can provide time service to which other hosts using the designations of *server* and *peer*, where

server Indicates a host from which you can request time.

peer Indicates a host with which this host can synchronize time. either peer host can request time from the other.

The following example shows a sample **/etc/ntp.conf** file:

```
# Server ip_addr version <ntp version>
server 9.8.77.66 version 3      # server is host_one.conrad.ibm.com
server 123.45.678.9 version 3  # server is host_two.ursula.ibm.com
peer 9.8.77.55 version 3      # server is host_x.conrad.ibm.com
peer 9.8.77.44 version 3      # server is host_y.conrad.ibm.com
peer 9.8.77.33 version 3      # server is host_z.conrad.ibm.com
```

Assuming this is the **/etc/ntp.conf** file on *your_host*, the following is true:

- *your_host* can request time from *host_one*, *host_two*, *host_x*, *host_y*, and *host_z*
- *your_host* can provide time, on request, to *host_x*, *host_y*, and *host_z*
- *your_host* cannot provide time to *host_one* or *host_two*.

The following table shows which host addresses will be used for servers and peers in the **/etc/ntp.conf** files created on the control workstation, file servers, and processor nodes for each **ntp_config** choice.

ntp_config	Control Workstation	Boot/Install Servers	Processor Nodes
timemaster	Server: Your time server Peer: All boot/install servers	Server: Your time server Peer: Control workstation Peer: All boot file servers	Server: Your time server Server: Control workstation Server: All boot file servers
Internet	Server: Internet server	Server: Control workstation Peer: Other boot file servers	Server: Control workstation Server: All boot file servers
consensus	Server: 127.127.1.10*	Server: Control workstation Peer: Other boot file servers	Server: Control workstation Server: All boot file servers
None	No file created	No file created	No file created
* This address signifies local time service to NTP and makes the control workstation a stratum 10 time server allowing you to hook into a higher stratum Internet time server at a later date.			

Related Information

The manual pages for NTP provide additional information on how NTP synchronizes time across a network. They are available in **/usr/lpp/ssp/man/man8**. The man pages are `xntpd` and `xntpd.c`.

For more information on the installation scripts, or to update the site environment, see the *PSSP: Installation and Migration Guide*.

Time Setting Options

To change time zones, use the **chtz** *timezone_info* command. For example:

```
chtz EST5EDT
```

sets the time zone to Eastern Standard Time and 5 hours from Greenwich Mean Time and sets the machine to know about Daylight Savings Time.

Nodes and Control Workstations Out of Time Synchronization

Generally, time services will adjust clocks incrementally to keep the clocks synchronized with the master clock. However, when time is off by a large amount, for example, an hour, it becomes difficult and time-consuming to incrementally synchronize with the time master.

To ensure that clocks stay synchronized, set nodes and the control workstation to understand Daylight Savings Time. Refer to the **script.cust** sample file in the *PSSP: Installation and Migration Guide* for information on setting time zones.

Chapter 8. Managing the Automounter

This chapter provides information on using an automounter on an SP system to manage the mounting of user home directories and other directories. Topics include:

- Understanding the automounter implementation
- Establishing an automount map for users' directories
- Using automount for other file systems
- Customizing the SP automounter function

For more information on the AIX automount daemon that is shipped as part of NFS in the AIX Network Support Facilities of the Base Operating System (BOS) Runtime, refer to *IBM AIX Version 4 System Management Guide: Communications and Networks*.

Understanding the Automounter Implementation

An automounter is a facility used to manage the mounting activity of a file system. When you access a file or directory under automounter control, the automounter transparently mounts the required file system. When there has been no activity to that file system for some pre-determined amount of time, the automounter unmounts the file system.

Automounters are typically used with the Network File System (NFS). NFS is a distributed file system that allows you to access files and directories located on remote systems and treat those files and directories as if they were local. When performing its mounting activity for an NFS file system, the automounter uses the NFS mounting facilities. The automounter reduces the period of time that a file system is actively mounted, thereby minimizing local system hangs due to NFS server outages.

On an SP system, the automounter is optionally used to manage mounting of user home directories and other directories. When configured, an automounter daemon runs on each node and is started at the time the node is booted. It mounts directories on demand and unmounts them after a period of inactivity. The mounted directories can come from SP boot/install servers or any workstation or server in the network, including local directories on that node.

The automounter manages directories specifically defined in automounter map files. Typically, there is one map file for each file system to be controlled by the automounter. The map files contain entries for each directory supported within the file system and the specific mount information for that directory. If SP User Management services have been configured, the SP creates and maintains a map file to control user home directories in the **/u** file system.

Requesting SP Automounter Support

You can choose to allow the SP to use an automounter to provide remote access to users' home directories or you can rely on other methods of automounting or file system management established at your site. You specify your choice by setting the automounter configuration site environment variable **amd_config** when defining your site environment. Specify your choice for the **Automount Configuration** variable on the Site Environment Information SMIT panel or use the **spsitenv** command on the control workstation. If **amd_config** is **true** (the default setting), the SP does the necessary directory and file configuration, and starts the automounter daemon. If SP User Management services have also been configured (the **usermgmt_config** site environment variable is **true**), the SP adds, removes, and changes entries in the automounter map file for the **/u** file system to maintain user home directory information when SP users are added, removed, or changed.

If you set **amd_config** to **false** when you initially install the SP system, it will not configure or start the automounter daemon and it will not maintain user home directory information in automounter map files. You can change the initial setting of **amd_config** at a later time using SMIT or the **spsitenv** command. However, this change will not affect the running state of the automounter daemon until the next time you reboot your nodes or control workstation or take other actions to start or kill the daemon. Note that if **amd_config** is initially **false** and then later set to **true**, all user management changes that had occurred while **amd_config** was **false** will not be reflected in the automounter map file for user home directories and that information may need to be manually added. If you choose to unconfigure the automounter by changing **amd_config** from **true** to **false**, make sure that all user home directories, as well as other directories previously served by the automounter, are distributed and made available by some other means.

When **amd_config** is **true**, the SP system uses the AIX automount daemon that is shipped as part of NFS in the AIX Network Support Facilities of the Base Operating System (BOS) Runtime. In systems with AIX 4.3.1 or later, this is the AutoFS automounter. In previous levels of AIX, this is the Automount automounter. *IBM AIX Version 4 System Management Guide: Communications and Networks* describes the operation and use of this automounter and the format of the map files. In addition to the map file for the **/u** file system maintained by the SP system, you may provide your own automount map files to enable the **automount** daemon to control similar access to directories in other file systems that you would like to have automatically mounted on demand.

If you choose to install and use a different automounter, the SP system allows you to provide a set of user customization scripts that the system will use to replace its functions for configuring and starting the automounter daemon and managing the map file for the user home directory information. **amd_config** must still be set to **true**.

If your SP system contains nodes that are running PSSP 2.2 or previous releases, those nodes use Amd, the BSD automounter. The **amd_config** site environment variable specifies your choice for SP automounter configuration for all nodes, including these older nodes running Amd. The control workstation still maintains the Amd map file, in addition to the Automount map file, to support those nodes running Amd.

Automount Map Files

The AIX **automount** daemon reads automount map files to determine which directories to support. Typically, there is one map file for each file system to be controlled by the automounter. The map file contains entries for each directory supported within the file system, the host name where the directory resides, and the specific mount information for that directory. The automount map files are kept in the **/etc/auto/maps** directory, by default. The list of all map files to be used by the **automount** daemon is specified in the master map file **/etc/auto.master**. This master map file contains entries for each file system to be controlled by the automounter, the name of the map file containing the directory information, and optional default mount information.

If automounter support and the SP User Management interface are both configured for your site environment, **/u** is automatically controlled by the automounter. The SP-managed automount map file for the **/u** file system is **/etc/auto/maps/auto.u**. This map file contains a stanza for each user added using SP User Administration (either the **spmuser** command or SMIT panels), containing the server name and directory location of the user's home directory. Existing users must be added to the map file using the **mkamdent** command.

An entry in the automount map file has the following format:

```
key -mount_options server_name:mount_directory:sub_directory
```

where:

<i>key</i>	The directory of interest within the file system
<i>mount_options</i>	Options used during the mount operation
<i>server_name</i>	The name of the machine that contains the directory
<i>mount_directory</i>	The exported directory from that machine
<i>sub_directory</i>	An optional sub-path under the exported directory

A special substitution value of **&** can be used to substitute the *key* value in the entry.

Note: An entry is added to **auto.u** for the net install ID's home directory. This **netinst** user ID is required by AIX for a net install.

Note for Users of AIX 4.3.0 or Earlier

The **automount** daemon will mount file systems in the **/tmp_mnt** directory using the following naming convention:

```
/tmp_mnt/filesystem/first-accessed-sub_directory
```

where *filesystem* is the name of the file system under automount control and *first-accessed-sub_directory* is the name of the first sub-directory accessed within a mounted directory. A symbolic link is made from the directory the user accessed to the appropriate sub-directory within that mount point:

```
/filesystem/sub_directory→/tmp_mnt/filesystem \  
first-accessed-sub-directory/sub_directory
```

where *sub_directory* is the name of the subpath currently being accessed by a user. This naming convention can be quite confusing because the physical directory path is dependent on some arbitrary name (that is, the first sub-directory accessed within a mounted directory). Because this name can change from one access period to another, you should not make any assumptions about this name or directly use it in any scripts or programs. Always access the directory through the automounter-supported path name. This naming convention may cause problems for C-shell users because the C-shell **pwd** built-in command returns the actual physical path and not the automounter-supported path, which is a symbolic link. The results of the C-shell command should not be stored for later use by the same program or another program or shared across a parallel execution environment; there is no guarantee the physical path name will exist.

Automount Daemon Example and AutoFS Example

The following examples illustrate how an automount map entry is used by the **automount** daemon and the **AutoFS** daemon to provide access to a user's home directory.

In the SP system, by default, the users' directories reside in **/home/hostname**, where *hostname* is the short name for the host system. The following stanza shows the entry in the automount map **/etc/auto/maps/auto.u** for user **jws** whose home directory resides on **luna**:

```
jws luna:/home/luna:&
```

In this case, if a user is on **luna** and changes directories to **/u/jws**, automount will create a symbolic link to the local directory:

```
/u/jws→/home/luna/jws
```

AutoFS locally mounts the **/home/luna/jws** JFS directory at **/u/jws**. Issuing the **mount** command will show an entry similar to:

```
/home/luna/jws /u/jws jfs Aug 07 09:07
```

However, if the user is on any machine other than **luna** and changes directories to **/u/jws**, automount mounts **luna:/home/luna** in the local **/tmp_mnt** directory at **/tmp_mnt/u/jws**. Automount then creates a symbolic link to the **jws** sub-directory under that mount point:

```
/u/jws→/tmp_mnt/u/jws/jws
```

AutoFS mounts the **luna:/home/luna/jws** NFS directory at **/u/jws**. Issuing the **mount** command will show an entry similar to:

```
luna  luna:/home/luna/jws  nfs  Aug 07 09:07
```

In the automount environment, because **jws** was the first user to access a sub-directory in **luna:/home/luna**, the **jws** sub-directory name appears twice in the physical path name. If a second user, **kcb**, also accesses a sub-directory in **luna:/home/luna**, automount creates the following symbolic link:

```
/u/kcb→/tmp_mnt/u/jws/kcb
```

In the automount environment, note the presence of **jws** in the physical path name. This is the name of the mount point the automounter used when mounting **luna:/home/luna**.

If a user's home directory resides in some other type of file system that already appears locally on the machine (for example, AFS), the server name should be specified as the local host. A map entry for a user with an AFS home directory would look like this:

```
tberry $HOST:/afs/kgn.ibm.com/usr2:&
```

In this case, when a user changes directories to **/u/tberry**, automount creates a symbolic link to the AFS directory:

```
/u/tberry→/afs/kgn.ibm.com/usr2/tberry
```

AutoFS mounts the **/afs/kgn.ibm.com/usr2/tberry** AFS directory at **/u/tberry**.

Note: The use of the **\$HOST** substitution variable is specific to the SP invocation of the **automount** daemon. The AIX **automount** command allows you to specify substitution variables and values to be used in the map files. The SP invocation of the **automount** daemon includes the parameter **-D HOST='uname -n'** which sets **\$HOST** to the name of the local machine as returned by the **uname** command.

See *IBM AIX 4 System Management Guide: Communications and Networks* for more information on the **automount**. daemon and map file format.

Automount Master Map Files

The SP invocation of the **automount** daemon uses the master map file **/etc/auto.master**. This master map file contains entries for each file system to be controlled by automount, the name of the map file containing the directory information, and optional default mount information. If you add additional file systems to be controlled by the **automount** daemon, you must edit this file and add an entry indicating the name of the file system and the full directory path of the associated map file. See "Using Automount for Other File Systems" on page 99.

If you wish to change the default mount options for mounting directories for the **/u** file system or any other file system controlled by the **automount** daemon, edit the **/etc/auto.master** file to add the mount options to the end of the appropriate file system entry. For example, to add mount options to the **/u** file system, your **/etc/auto.master** entry may be modified to appear as follows:

```
/u /etc/auto/maps/auto.u -rw,hard,retry=3,timeo=40,rsize=4096,wsiz=4096
```

The master map file can also specify an NIS database as a map file. The Network Information Service (NIS) is a distributed database that allows you to maintain consistent configuration files throughout a network. By default, the SP invocation of the **automount** daemon turns off the use of the **auto.master** NIS database. If you are using NIS to maintain and distribute your automounter map files, you will need to edit the **/etc/auto.master** file to add as the first entry **+auto.master** which includes the NIS **auto.master** database.

See *IBM AIX 4 System Management Guide: Communications and Networks* for more information on the automount master map file format and on NIS.

File Collections and Automounter Files

If you have configured the SP to use file collections by setting the site environment variable **filecoll_config** to **true**, automount map files will be automatically distributed to all nodes. The automount map files are part of the **user.admin** file collection. The **/var/sysman/sup/user.admin/list** file contains entries to distribute the **/etc/auto.master** file, all files in the **/etc/auto/maps** directory, and all files in the **/etc/auto/cust** directory.

If you decide to replace some or all of the SP automounter function with your own customizations, you may need to edit the **/var/sysman/sup/user.admin/list** file to distribute your own files or to remove the currently distributed SP files. In the list file, there exists a comment mentioning that SP automounter configuration has been added. Do **NOT** delete or change this comment. You may change or remove any other entries in the file as necessary for your automounter installation. However, if this comment is removed or altered on the control workstation, the next time your system is booted or the **services_config** command is run, the SP configuration will add the automount entries back again. See “Customizing the SP Automounter Function” on page 103 for more details on providing your own automounter customizations.

If you are not using file collections on your SP, you must use some other local means of distributing the automount map file, master map files, and user customization scripts to all of the nodes on your system. The SP will not copy your map files to the nodes for you.

See Chapter 4, “Managing File Collections” on page 41 for more information on the file collection processes.

Automounter Diagnostics

The SP automounter configuration and execution functions write error messages to an error log file, **/var/adm/SPlogs/auto/auto.log**. Also, the **automount** daemon uses the daemon facility of the BSD syslog subsystem to record errors. The control workstation and nodes have been configured to write all **daemon.notice** and greater messages to **/var/adm/SPlogs/SPdaemon.log**. If you are having problems with the automounter, you may need to stop the daemon, check the diagnostic information recorded in these logs, and restart the **automount** daemon. For more information on error logging, refer to Chapter 28, “Managing Error Logs” on page 383. You can find a description of the automounter error log, messages, and instructions for stopping and starting the **automount** daemon in *PSSP: Diagnosis Guide*.

Establishing an Automount Map for Users' Directories

If you have configured your SP to include both automounter and user management support, the SP will create and maintain an automount map file for users' home directories. When you add a new SP user to your system, a map entry will be added to **/etc/auto/maps/auto.u** associating the user's **/u** home directory with the actual physical directory located either on the current machine or on a remote file server that can be accessed through a network connection using NFS. This map entry will be changed or removed when SP User Management functions are used to change or remove the user from the system.

If you have existing users that do not have entries in the **auto.u** map file, you can use the **mkamdent** command to add home directory entries for them. The **mkamdent** command allows you to specify a list of users to add to the **auto.u** map. The command builds a stanza for each user in the list using the SP site environment values for **homedir_server** and **homedir_path** as the defaults (these can be set using either SMIT panels or the **spsitenv** command). **homedir_server** is the default file server and **homedir_path** is the default base path for the home directory (not including the individual user's sub-directory). You can override these defaults with the **-s** flag. See *PSSP: Command and Technical Reference* for more information on the **mkamdent** and **spsitenv** commands.

You must make sure that the **/u** directory is not a symbolic link to some other directory. This will prevent the **automount** daemon from starting successfully. **/u** must exist as a local directory on each processor.

If you are not using SP User Management services, but would still like to have the **automount** daemon control access to directories in the **/u** file system, you can create and maintain your own **auto.u** map file. See "Using Automount for Other File Systems" for details on how to add the map to the SP automounter support.

Changing User Home Directory Information

It may become necessary to change the user home directory information stored in the automounter map file. If, for example, the home directory is moved to a new server machine or to a different directory, the existing entry in the automounter map file will need to be updated with the new information. IBM suggests changing this information by using the **spchuser** command or the equivalent SMIT panels.

After updating the map file, you may need to wait up to 5 minutes with no access attempts to the user's home directory through the **/u** file system. This allows the automounter to time out any previous access attempts and unmount any existing mounts that may be referencing the old information. Do not attempt to force the unmount by manually issuing the **umount** command as that will put the automounter in an inconsistent state. You would then need to stop and restart the **automount** daemon to recover access to that user's home directory.

Using Automount for Other File Systems

If you have configured your SP with automounter support, you can use the **automount** daemon to provide automounting control for other file systems in your network. Each file system you want managed by automount requires an automount map file and an entry in the **/etc/auto.master** map file identifying the file system and its map file.

You must make sure the directory you are controlling with the **automount** daemon exists as a local directory on each processor and is not a symbolic link to another directory. Otherwise, the **automount** daemon will not start successfully.

The process to add additional file systems requires you to stop and start the **automount** daemon on all nodes. To do so, all activity with files and directories currently controlled by the **automount** daemon must be stopped. This includes all activity in users' home directories if the **automount** daemon is controlling the **/u** file system. You should plan to add all additional file systems to the automounter at a time when this would not adversely affect your system schedule or disrupt users.

The examples in this section illustrate the process for adding additional file systems and show how to provide automount support for automounting the **/vol** file system.

The steps in this process are:

1. Create an automount map file
2. Update the master map file
3. Add the map to the file collections
4. Distribute the map to the boot/install servers and nodes
5. Stop and restart the **automount** daemon

Step 1: Create an Automount Map File

The easiest way to create an automount map file is to copy an existing one and modify it for the new file system. The map can reside in any directory you like. The map for the **/u** file system is **/etc/auto/maps/auto.u**. It may be convenient for you to keep all of your automount map files in this **maps** directory.

Example

1. On the control workstation, copy the existing **auto.u** map and name it **auto.vol**, keeping it in the same directory. Enter:

```
cp /etc/auto/maps/auto.u /etc/auto/maps/auto.vol
```

2. Replace the map entries with the directory and host information for **/vol** as shown here:

```
inst.images images1:/inst.images
techlib     filesrv1:/share/&
local      $HOST:/afs/site.edu/share/local
```

This example results in the following mounts for AIX 4.3.0 and earlier levels:

- If a user issues **cd /vol/inst.images**, **inst.images** is mounted from **images1** linking **/vol/inst.images**→**/tmp_mnt/vol/inst.images**
- If a user issues **cd /vol/techlib**, **/share/techlib** is mounted from **filesrv1** linking **/vol/techlib**→**/tmp_mnt/vol/techlib**
- If a user issues **cd /vol/local**, automount simply creates a symbolic link because the directory is accessible from the local host. The result is **/vol/local**→**/afs/site.edu/share/local**.

This example results in the following mounts for AutoFS systems:

- If a user issues `cd /vol/inst.images`, `inst.images` is mounted from `images1` at the local mount point `/vol/inst.images`
- If a user issues `cd /vol/techlib`, `/share/techlib` is mounted from `filesrv1` at the local mount point `/vol/techlib`
- If a user issues `cd /vol/local`, a local mount is made from `/afs/site.edue/share/local` to the mount point `/vol/local`.

Step 2: Update the Master Map File

Add an entry to the automount master map file `/etc/auto.master` so that when the **automount** daemon is restarted it will also control the file system being added. The entry may contain default mount options for all the directories listed in the automount map file.

Example

Add the following entry to `/etc/auto.master`:

```
/vol /etc/auto/maps/auto.vol -ro,soft,rsize=4096,wspace=4096
```

This will tell the **automount** daemon to use the `/etc/auto/maps/auto.vol` file when controlling access to directories within the `/vol` file system. The specified mount options will be used for each directory mount.

Step 3: Add the Map to the File Collections

If you are using file collections to distribute your map files, add the new map so that it can be distributed and managed by the file collection programs. If you have placed your map file in the `/etc/auto/maps` directory, you do not need to do anything for this step. The **user.admin** file collection already distributes all files in the `/etc/auto/maps` directory. However, if you have not placed your map file in the `/etc/auto/maps` directory, you will need to add the map to a file collection that is resident on all the systems you require. The **user.admin** collection is a good choice because it is resident on the control workstation, boot/install servers, and all processor nodes. See “SP File Collection Summary” on page 44 for a complete list of file collections provided by the SP.

Example

Add `auto.vol` to the **user.admin** file collection so that it can be distributed to all boot/install servers and nodes. To do this, edit the file collection list file `/var/sysman/sup/user.admin/list` and add the following entries to request an upgrade of this new map and to refresh the daemon after the upgrade:

```
upgrade ./etc/auto/maps/auto.vol
execute /etc/amd/refresh_amd (./etc/auto/maps/auto.vol)
```

Note: This step is not necessary for this example since the map file resides in the `/etc/auto/maps` directory and will already be automatically distributed. The details are provided here to illustrate how to update the file collection if the map file resides in some other directory.

Step 4: Distribute the Map to the Boot/Install Servers and Nodes

If you are using file collections to distribute your map files, check to see if you have a **scan** file for the file collection containing the new map.

Example

If you are using the **user.admin** file collection to distribute your map, check if the file **/var/sysman/sup/user.admin/scan** exists on the control workstation or any boot/install servers. If the **scan** file does exist, you will need to update it. On each processor that has a **scan** file for the collection, enter:

```
supper scan user.admin
```

You can wait for the scheduled **supper update** to update the file collection as specified in the **crontabs** file on each processor node. Or, if you want to distribute the map immediately, use a remote command to run the **supper update** command first on each boot/install server and then on each node.

Example

To distribute the **user.admin** file collection with the new map file immediately, run the following command first on each boot/install server and then on each node:

```
supper update user.admin
```

Step 5: Stop and Restart the automount Daemon

In order for the automounter to recognize the update to the **/etc/auto.master** file and handle the new file system, the automounter must be stopped and restarted. The daemon must be stopped at a time when no users are accessing files or directories under automount control. Plan this step for a time that is not disruptive to your users.

The cleanest and safest way to stop and restart the automounter is to reboot all the systems where this new map will reside.

If it is not possible to reboot the systems, then do the following to stop the **automount** daemon on AIX older systems:

1. Issue the **mount** command with no parameters to get the process ID of the **automount** daemon and to see if there are any active mounts on the **/tmp_mnt** directory. This is the directory the **automount** daemon uses as its mount point.
2. Stop all processes that are accessing files or directories controlled by the **automount** daemon.
3. Stop the **automount** daemon with the following command using the process ID returned by the mount command:

```
kill -term process_id
```

Note: It is important to issue the **kill -term** command (**kill -15**) and not **kill -kill** (**kill -9**) to allow the **automount** daemon to properly clean up its mounts and directories.

To stop AutoFS on systems with AIX 4.3.1 or later:

1. Issue the **mount** command with no parameters to see if there are any active mounts for automounter controlled directories.

2. Stop all processes that are accessing files or directories controlled by the automounter.
3. The automounter daemon is controlled by the System Resource Controller (SRC). Issue the following command to stop the AutoFS system:

```
stopsrc -g autofs
```

Restart the **automount** daemon by running the following command:

```
/etc/auto/startauto
```

Customizing the SP Automounter Function

The SP automounter function has been implemented so that you can optionally replace some or all of the SP automounter function with your own customization scripts. By placing executable files in the **/etc/auto/cust** directory, you can provide your own automounter function that will be executed in place of the SP automounter function. This is useful if you need to do things such as change the map file entry format for entries in the **/u** map file **/etc/auto/maps/auto.u**, change the default options for invoking the **automount** daemon, or completely replace the SP use of the **automount** daemon with some other automounter that you provide.

In order to use these customization scripts, the SP automounter support must be configured for your system (the **amd_config** site environment variable is set to **true** using either the **spsitenv** command or SMIT panels). During the execution of the SP automounter function, a check will be made to see if any of the following files exist and are executable. If so, that file will be invoked in place of the corresponding automounter function:

/etc/auto/cust/cfgauto.cust	Configure the automounter directories and create default files as necessary
/etc/auto/cust/startauto.cust	Start the automounter daemon
/etc/auto/cust/refauto.cust	Refresh the automounter daemon
/etc/auto/cust/checkauto.cust	Verify the automounter configuration and daemon execution
/etc/auto/cust/mkautoent.cust	Add an entry to the automounter map file for user home directories
/etc/auto/cust/rmautoent.cust	Remove an entry from the automounter map file for user home directories
/etc/auto/cust/lautoent.cust	List an entry from the automounter map file for user home directories

Each file is independent and any or all files may exist. The SP automounter function will be replaced only for those functions which have an executable customization script. Details on how each of these files will be used by the SP is provided in the following sections.

If the automounter function is to be replaced on all boot/install servers and processor nodes, the customization scripts must be distributed. If you have file collections configured for your system, these files will automatically be distributed through the **user.admin** file collection. If you are not using file collections, you

must distribute these files to the boot/install servers and processor nodes using some other means.

cfgauto.cust Customization Script

The `/etc/auto/cust/cfgauto.cust` file is an optional customization script that you may provide to configure the automounter directories and create default files. If this file exists, and it is executable, it will be invoked by the `services_config` installation command during the system boot process to configure your automounter installation. You should provide this file when you require a different automounter configuration than that provided by the SP. IBM suggests that your `cfgauto.cust` script create all directories and default files that are required by the automounter started by your `/etc/auto/cust/startauto.cust` script.

The `services_config` process will create the SP automounter log file as `/var/adm/SPlogs/auto/auto.log`. It will then check if `cfgauto.cust` exists and is executable, and if so, will invoke it. If not, the SP automounter configuration will be invoked using the `/usr/lpp/ssp/install/bin/cfgauto` installation script which creates the `/etc/auto` and `/etc/auto/maps` directories, and creates default `/etc/auto.master` master map and `/etc/auto/maps/auto.u` map files as necessary.

There are no input parameters passed to the `cfgauto.cust` script. The return value should indicate the following:

- 0** Success
- non-0** A configuration problem occurred. In this case, the `services_config` process will not start the automounter daemon, but will continue with its other configuration functions.

Example

The following script is an example of a script you might use to configure directories and default files for a processor to use the Amd automounter instead of the AIX `automount` daemon:

```
#!/usr/bin/ksh

# Create /etc/amd directory
if [[ ! -a /etc/amd ]]; then
    mkdir /etc/amd
    if (($? != 0)); then
        echo "cfgauto.cust: Cannot create directory /etc/amd."
        exit 1
    fi
fi

# Create /etc/amd/amd-maps dir
if [[ ! -a /etc/amd/amd-maps ]]; then
    mkdir /etc/amd/amd-maps
    if (($? != 0)); then
        echo "cfgauto.cust: Cannot create directory /etc/amd/amd-maps."
        exit 1
    fi
fi

# Create default amd.u file with /defaults entry and an
# entry for the netinst userid
if [[ ! -a /etc/amd/amd-maps/amd.u ]]; then
```

```

        echo "/defaults rfs:=/homes/filesrv1;type:=nfs;sublink:=${key};
opts:=rw,hard,retry=3, timeo=40,rsize=4096,wsiz=4096" >>
/etc/amd/amd-maps/amd.u
    if (($? != 0)); then
        echo "cfgauto.cust: Cannot create default file
        /etc/amd/amd-maps/amd.u."
        exit 1
    fi
    echo "\nnetinst type:=link;fs:=/home" >> /etc/amd/amd-maps/amd.u
fi

# Set up /u so that Amd can handle mounts for amd.u map:
# No symbolic links allowed
if [[ -L /u ]]; then
    rm /u
    mkdir /u
fi
exit 0

```

startauto.cust Customization Script

The `/etc/auto/cust/startauto.cust` file is an optional customization script that you can provide to start your automounter daemon. If this file exists, and it is executable, it will be invoked by the `services_config` installation command during the system boot process to start your automounter daemon. You should provide this file when you require changes to the SP invocation of the `automount` daemon, or when you wish to replace it with the invocation of a different automounter daemon that you have made available on your system.

The `services_config` process will invoke the automounter configuration, and if that completes successfully, a check will be made to see if `startauto.cust` exists and is executable. If so, it will be invoked to start the automounter daemon. If not, the `automount` daemon will be started using `/etc/auto/startauto`.

There are no input parameters passed to the `startauto.cust` script. The return value has no affect on the `services_config` execution.

Example

The following script is an example of a user customization script that you might use to start the Amd daemon instead of the AIX `automount` daemon. This script assumes that you have Amd installed on your system and that the executables reside in the `/etc/amd` directory.

```

#!/usr/bin/ksh

# Check if Amd daemon is running
if [[ -n 'ps -ef | grep /etc/amd/amd | grep -v grep' ]]; then
    echo "startauto.cust: amd daemon is already running."
    exit 0
fi

# Build amd input list using all amd.* map files in
# /etc/amd/amd-maps directory
set -A amdmaps $(ls /etc/amd/amd-maps/amd.*)
let i=0
while (( $i < ${#amdmaps[*]} )); do
    amd_parms="$amd_parms /${amdmaps[$i]}##*/amd."
    ${amdmaps[$i]}
    let i=i+1
done

# Start the daemon
nice --4 /etc/amd/amd -t 16.120 -x all -l /var/adm/SPlogs/auto/auto.log
$amd_parms
exit $?

```

refauto.cust Customization Script

The **/etc/auto/cust/refauto.cust** file is an optional customization script that you can provide to refresh your automounter daemon. If this file exists, and it is executable, it will be invoked after a file collection update has been made to the **/etc/auto/maps/auto.u** map file for user home directories. You should provide this file when your automounter requires special refresh activity.

If you have file collections configured for your SP system, the **supper update** process on a boot/install server or processor node will update the **/etc/auto/maps/auto.u** map file when changes have been made. If this map file is updated, the automounter refresh is invoked. The refresh process will check to see if the **refauto.cust** customization script exists and is executable. If so, it will be invoked to refresh the automounter daemon. If not, no activity will occur since the **automount** daemon does not require refreshing.

If you are using file collections to distribute your map files and your automounter requires refreshing for updates to map files other than **/etc/auto/maps/auto.u**, you will need to update the list file for the collection. The **user.admin** collection is used to distribute all map files in the **/etc/auto/maps** directory. If you are also using **user.admin** to distribute your maps, you will need to edit the list file **/var/sysman/sup/user.admin/list** to contain an execute statement similar to the one for **auto.u**:

```
execute /etc/amd/refresh_amd (./etc/auto/maps/auto.u)
```

There are no input parameters passed to the **refauto.cust** script. The return value is noted by the **supper update** process.

Example

The following script is an example of a user customization script that you might use to refresh the Amd daemon. This script assumes that Amd is installed and that the executables reside in the **/etc/amd** directory.

```
#!/usr/bin/ksh

# If amd is running then refresh the maps with amq -f
if [[ -n `ps -ef | grep /etc/amd/amd | egrep -v grep` ]]; then
    /etc/amd/amq -f
fi
```

checkauto.cust Customization Script

The **/etc/auto/cust/checkauto.cust** file is an optional customization script that you can provide to verify your automounter installation. If this file exists, and it is executable, it will be invoked from the **SYSMAN_test** command to verify your automounter installation. You should provide this file when you have provided your own automounter configuration using **/etc/auto/cust/cfgauto.cust** or started your own automounter daemon with **/etc/auto/cust/startauto.cust**. IBM suggests that your **checkauto.cust** checks all files and directories required by your automounter and that your automounter daemon is running.

The **SYSMAN_test** process will check if **/etc/auto/cust/checkauto.cust** exists and is executable. If so, it will invoke it. If not, it will invoke the SP automounter verification. This code checks that all required SP automounter directories and files exist and that the SP automounter daemon is running.

The **checkauto.cust** script will be invoked with the same syntax as the **SYSMAN_test** command:

```
checkauto.cust [-q | -v] -l log_file
```

The parameters are those supplied to the **SYSMAN_test** command with the following definitions:

- q** Specifies quiet mode; error messages are only appended to the log file and not written to stdout.
- v** Specifies verbose mode; all error and informational messages are written to both the log file and stdout.
- l *logfile*** Specifies the path name of the log file to which messages are appended.

If neither **-q** nor **-v** are specified, error messages are written to both the log file and stdout.

The return value from **checkauto.cust** execution should be:

- 0** Indicates success
- non-0** Indicates the number of verification errors that were found

Example

The following script is an example of a user customization script that you might use to verify your Amd configuration and daemon execution. This script assumes that Amd is installed and that the executables reside in the **/etc/amd** directory.

```
#!/usr/bin/ksh

while getopts "qvl:" opt; do
  case $opt in
    q ) QUIET="-q" ;;
    v ) VERBOSE="-v" ;;
    l ) LOGFILE=$OPTARG ;;
  esac
done

let ERRORS=0
# Check for /etc/amd directory
if [[ ! -a /etc/amd ]]; then
  if [[ -z $QUIET ]]; then
    echo "checkauto.cust: Directory /etc/amd does not exist."
  fi
  echo "checkauto.cust: Directory /etc/amd does not exist." >> $LOGFILE
  let ERRORS=$ERRORS+1
fi

# Check for /etc/amd/amd-maps dir
if [[ ! -a /etc/amd/amd-maps ]]; then
  if [[ -z $QUIET ]]; then
    echo "checkauto.cust: Directory /etc/amd/amd-maps does not exist."
  fi
  echo "checkauto.cust: Directory /etc/amd/amd-maps does not exist." >> $LOGFILE
  let ERRORS=$ERRORS+1
fi

# Check for default amd.u file
if [[ ! -a /etc/amd/amd-maps/amd.u ]]; then
  if [[ -z $QUIET ]]; then
    echo "checkauto.cust: Map file /etc/amd/amd-maps/amd.u does not exist."
  fi
  echo "checkauto.cust: Map file /etc/amd/amd-maps/amd.u does not exist." >> $LOGFILE
  let ERRORS=$ERRORS+1
fi

# Check if /u is a symbolic link
if [[ -L /u ]]; then
  if [[ -z $QUIET ]]; then
    echo "checkauto.cust: /u is a symbolic link. Not allowed by Amd."
  fi
  echo "checkauto.cust: /u is a symbolic link. Not allowed by Amd." >> $LOGFILE
  let ERRORS=$ERRORS+1
fi

# Check if Amd daemon is running
if [[ -z `ps -ef | grep amd | grep -v grep` ]]; then
  if [[ -z $QUIET ]]; then
    echo "checkauto.cust: amd daemon is not running."
  fi
  echo "checkauto.cust: amd daemon is not running." >> $LOGFILE
  let ERRORS=$ERRORS+1
fi

exit $ERRORS
```

mkautoent.cust Customization Script

The `/etc/auto/cust/mkautoent.cust` file is an optional customization script that you can provide to add a user home directory entry to your automounter map file. If this file exists, and it is executable, it will be invoked by the SP User Management commands `spmkuser`, `spchuser`, and `mkamdent` either directly or through SMIT panels. You should provide this file when you require a different map file or map file format than that generated by the SP User Management services for the `/etc/auto/maps/auto.u` map file. If you do provide this file, you will probably also need to provide `/etc/auto/cust/rmautoent.cust` and `/etc/auto/cust/lsautoent.cust`. These files remove and list the user's home directory entry in the map file.

When SP User Management services process user home directory additions or updates, a check is made to see if `mkautoent.cust` exists and is executable. If so, it will be invoked to process the home directory information for your automounter map file. If not, the automount map file `/etc/auto/maps/auto.u` will be updated with the new home directory entry.

The Parallel Operating Environment (POE) component in the AIX Parallel Environment contains the command `mpamddir` which is dependent on the format of the SP supported automount map file. If you have this support installed on your system and are using the command, you may need to provide an additional user exit. See *IBM Parallel Environment for AIX: Operation and Use* for more information.

The `mkautoent.cust` script is invoked with the following syntax:

```
mkautoent.cust user host[,host...] pathname
```

The parameters have the following definitions:

<i>user</i>	The user name that is to be used as a key to the new map entry.
<i>host[,host...]</i>	One or more host names separated by commas indicating the target host where the user's home directory resides.
<i>pathname</i>	Directory pathname of the user's home directory as it is defined on the host. The trailing user subdirectory has been removed from the pathname.

For example, the syntax for a user `jws` with a home directory on `luna` with the path name of `/home/luna/jws` would be:

```
mkautoent.cust jws luna /home/luna
```

The return value from `mkautoent.cust` execution should indicate:

- | | |
|--------------|---|
| 0 | Success |
| non-0 | An error occurred creating the map file entry. The following actions will occur: <ul style="list-style-type: none">• The <code>mkamdent</code> command will return this value as its return code.• The <code>spmkuser</code> command creates the user before invoking this script, so the return code has no effect on the command. The user is still added.• The <code>spchuser</code> command will terminate and not change the user information. |

Example

The following script is an example of a user customization script that you might use to add an entry to your **/etc/amd/amd-maps/amd.u** map file.

```
#!/bin/ksh

# get input parms
USER=$1
HOSTS=$2
HOMEPATH=$3
AMDMAP=/etc/amd/amd-maps/amd.u
HOSTS='echo $HOSTS | sed 's/,/ /g''

# Append the entry to the amd /u map
FIRST=" "
for HOST in $HOSTS; do
    if [[ -z $FIRST ]]; then
        echo "$USER host==$HOST;type:=link;fs:=$HOMEPATH \\" >> $AMDMAP
        FIRST=$HOST
    else
        echo "host==$HOST;type:=link;fs:=$HOMEPATH \\" >> $AMDMAP
    fi
done
echo "host!=$FIRST;type:=nfs;rhost:=$FIRST;rfs:=$HOMEPATH" >> $AMDMAP
```

rmautoent.cust Customization Script

The **/etc/auto/cust/rmautoent.cust** file is an optional customization script that you can provide to remove a user home directory entry from your automounter map file. If this file exists, and it is executable, it will be invoked by the SP User Management commands **spruser** and **spchuser** either directly or through SMIT panels. You should provide this file when you have provided **/etc/auto/cust/mkautoent.cust** support for a different map file or map file format than that generated by the SP User Management services for the **/etc/auto/maps/auto.u** map file.

When SP User Management services process user home directory removals, a check is made to see if **rmautoent.cust** exists and is executable. If so, it will be invoked to remove the home directory information from your automounter map file. If not, the entry will be removed from the automount map file **/etc/auto/maps/auto.u**.

The **rmautoent.cust** script will be invoked with the following syntax:

```
rmautoent.cust user
```

The parameters have the following definitions:

user The user name that is used as the key of the map entry to be removed.

For example, the syntax for a user **jws** with a home directory on **luna** with the path name of **/home/luna/jws** would be:

```
rmautoent.cust jws
```

The return value from **rmautoent.cust** execution should indicate:

0 Success

non-0 An error occurred removing the map file entry. The following actions will occur:

- The **spruser** command removes the user before invoking this script, so the return code has no affect on the command. The user will still be removed.
- The **spchuser** command will terminate and not change the user information

Example

The following script is an example of a user customization script that you might use to remove an entry from your **/etc/amd/amd-maps/amd.u** map file.

```
#!/bin/ksh

# Get input parms
USER=$1
AMDMAP=/etc/amd/amd-maps/amd.u

# Delete the entry to the amd /u map
sed -e"/^$USER /,/[^\\]$/d" $AMDMAP > $AMDMAP.tmp
mv $AMDMAP.tmp $AMDMAP
exit 0
```

Isautoent.cust Customization Script

The **/etc/auto/cust/Isautoent.cust** file is an optional customization script that you can provide to list a user's home directory information stored in your automounter map file. If this file exists, and it is executable, it will be invoked by the SP User Management command **spluser** either directly or through SMIT panels. You should provide this file when you have provided **/etc/auto/cust/mkautoent.cust** support for a different map file or map file format than that generated by the SP User Management services for the **/etc/auto/maps/auto.u** map file.

The **spluser** command checks to see if **Isautoent.cust** exists and is executable. If so, it will be invoked to list the home directory information from your automounter map file. If not, the information stored in the automount map file **/etc/auto/maps/auto.u** will be listed.

The **Isautoent.cust** script will be invoked with the following syntax:

```
Isautoent.cust user
```

The parameter has the following definition:

user The user name that is used as the key of the map entry containing the home directory information to be returned.

The user home directory information should be echoed to stdout in the following format:

```
host [,host...]
pathname
```

where:

host [,*host*...] Lists one or more host names separated by commas indicating the target host where the user's home directory resides.

pathname Lists the directory path name of the user's home directory as it was provided to the **mkautoent.cust** script with the trailing *user* subdirectory removed from the *pathname*.

For example, the syntax for a user **jws** with a home directory on **luna** with the path name of **/home/luna/jws** would be:

```
lsautoent.cust jws
```

and the return value written to stdout would be:

```
luna /home/luna
```

The return value from **lsautoent.cust** execution should indicate:

0 Success

non-0 An error occurred listing the map file entry. The **spluser** command will list default user home directory information if the output from this script is null.

Example

The following script is an example of a user customization script that you might use to remove an entry from your **/etc/amd/amd-maps/amd.u** map file.

```
#!/bin/ksh
# Get input parms
USER=$1
AMDMAP=/etc/amd/amd-maps/amd.u

# Find the entry in the amd /u map and return the host and directory
hosts='cat $AMDMAP | sed "
/^$USER /,[^\]$/!d
/host==/!d
s/. *host==\([^; ]*\).*$/\1/g"'
hosts='echo $hosts | sed "s/\([^ ]*\)[ ] [ ]*\(\.[^ ]*\)/\1,
\2/g"'
home_dir='cat $AMDMAP | sed "
/^$USER /,[^\]$/!d
/rfs:=/!d
s/. *rfs:=\([^; ]*\)[^ ]*/\1/g"'

echo $hosts $home_dir

exit 0
```

Chapter 9. Managing Mail Service

The Mail Services provided on the SP are the same as the ones provided on a standard AIX environment. From the mail service point of view, the SP can be viewed as any other cluster of nodes. If you need to provide mail access on the nodes, the same configuration rules and techniques used for the rest of the LAN or intranet apply. Since this topic is not SP specific, but rather AIX specific, see the following documents for additional information:

- From the Internet
 - AIX Installation Guides at Web site:
http://www.rs6000.ibm.com/doc_link/en_US/a_doc_lib/aixuser/usrcomm/toc.htm

Search for the *mail overview* chapter.
 - AIX System Management Guides at Web site:
http://www.rs6000.ibm.com/doc_link/en_US/a_doc_lib/aixbman/commadm/toc.htm

Search for *mail* in the table of contents.
- From the Redbook Libraries
 - The Next Step in Messaging: Upgrade Case Studies for Lotus cc:Mail to Lotus Domino and Lotus Notes
- From the Administration Guides
 - AIX Version 4.3 System Users' Guide: Communications and Networks

Besides the IBM publications on the subject, several other books discuss all pertinent subjects like SMTP, UUCP, TCP/IP, POP, IMAP, DNS, NFS, NIS, and even configuring popular E-mail software packages.

Chapter 10. Accounting

Your hardware resources are likely to be distributed unevenly across your SP System, with some nodes equipped for greater processing capacity and I/O than others. For this reason you may want to charge different usage fees for different nodes. You may also want to consolidate all accounting records, for the entire SP system, or for groups of nodes.

One way to make efficient use of the nodes in your SP system is to logically partition them so that each pool supports a different type of workload and is considered a separate management domain. The pooling of nodes according to configuration or workload type is the concept behind the *accounting class*. SP accounting allows you to logically group SP nodes with similar operating characteristics into an accounting class for which the same charge fee applies.

You may also want to charge a different usage fee for compute-intensive jobs when they demand exclusive use of one or more nodes. SP System accounting provides a mechanism for treating these jobs separately as well.

Accounting by Node or by Class

An accounting class is a group of nodes for which accounting data is merged together, providing a single report for the nodes in that class. Individual nodes within a class may be enabled or disabled for accounting.

The **acct_class_id**, an attribute of each node object in the System Data Repository, is an arbitrary string you define and specify for all nodes for which you wish to group and merge accounting data.

Default values eliminate the need to define each attribute for each node. The default value of **acct_class_id** for each node is **default**. Therefore, all nodes initially belong to one accounting class called **default**.

The initial value of the **acct_enable** attribute for each node is also **default**. All nodes with **acct_enable=default** are enabled or disabled for accounting based on a system-wide accounting enabled attribute (**spacct_enable**) you define for your site environment. When you set **spacct_enable** to **true**, it enables accounting on all nodes for which the individual accounting enabled attribute (**acct_enable**) is set to **default**. Likewise, when you set **spacct_enable** to **false** (the initial value at installation), accounting is disabled on all nodes that have **acct_enable=default**. Thus, accounting can be enabled or disabled for the entire SP system with individual nodes treated as exceptions.

How SP Accounting Differs from Standard AIX

Standard AIX accounting uses a script called **runacct**. This script is normally run every night by **cron** to provide daily usage reports. It also keeps a running total of usage over some fiscal period (usually a month). SP accounting splits the **runacct** function into two parts. The first part is a script that is run on each SP node every night, just as **runacct** does. This script is named **nrunacct**. The second part is a script that executes on an **accounting_master** node each night, after the **nrunacct** scripts are run. This script is named **crunacct** and it consolidates the **nrunacct** information from each node. The scripts **cmonacct** and **cprdaily** are SP versions

of standard AIX programs, modified to handle accounting class and exclusive use accounting.

nrunacct

The **nrunacct** script provides the function that merges raw data from the process, login, print, fee and disk subsystems. It also provides support to account for the exclusive use of a node.

nrunacct also creates an SP version of the **loginlog** file using the **var/adm/acct/sum/loginlog** files for the specified accounting cycle. Only the most recent **loginlogyyyyymmdd** file from each node is pertinent.

nrunacct provides a user exit for any site-specific accounting procedures in the **var/adm/nsiteacct** file.

crunacct

The **crunacct** script provides the function that produces daily summary reports and accumulates data for the fiscal period, using the merged data from each node.

SP accounting merges total accounting data by user ID and login name, just as the standard AIX accounting does. This means that all SP nodes must have identical user IDs and login names. In other words, all SP nodes must be members of the same user name space, whether it is implemented through NIS, the SP User Management tools, or through other means.

cmonacct

An SP version of **monacct**, **cmonacct** creates summary files for monthly or fiscal reports.

cprdaily

An SP version of **prdaily**, **cprdaily** creates daily report files of accounting activity and cleans up files no longer needed.

Accounting for Exclusive Use of Node Jobs

While SP nodes are expected to execute exclusive parallel jobs, nonexclusive parallel jobs, and nonparallel jobs during a single accounting cycle, normal operation often results in the exclusive use of a node when executing a compute-intensive parallel job. A single job that makes such great demands on the compute resources of a node where no other jobs can share that node's resources is said to be an *exclusive use of a node* job. Such jobs can be handled separately for accounting purposes.

Standard AIX accounting produces a record for each process as the process completes. These records are later merged by user ID and login name. A user who executes one or more types of jobs on the same node has all process accounting records for that node merged together, resulting in the loss of those processes that had exclusive use of the node. It is important to track the exclusive use of a node by a user because the cost is likely to be different than the cost of shared use.

SP accounting addresses exclusive-use jobs as follows: Before starting a job that has exclusive use of a node, LoadLeveler executes a start job command under the real user ID of the job. When the job is finished LoadLeveler executes an end job command, again under the real user ID of the job. These commands consist solely

of an **exit()** system call in their **main()** procedure. Their purpose is to generate process accounting records for the user ID of the job that can then be used to *bracket* all of the normal process accounting records of the job. In other words, all of the process accounting records generated by an exclusive job can be identified and excluded from the other process accounting records generated on the node.¹

The process accounting records for the start job and end job programs are used to calculate the time the job had exclusive use of the node. The time is then charged to the user ID of the job by converting it to chargefee units. Use of the chargefee mechanism eliminates the need to generate additional accounting reports.

Accounting records for the start job and end job programs are identified by the names of the commands as found in the records. These start and stop programs are given the nonstandard names of **SJSJSJSJ** and **EJEJEJEJ** to avoid conflict with any other standard job names.

An attribute of the SP object in the SDR, **spacct_exclude_enable**, determines whether the start job program and end job program are executed for exclusive-use jobs. A node-level SDR attribute, **acct_exclude_enable**, determines whether accounting is activated to process start and end job records, and to generate chargefee records for jobs making exclusive use of that node.

Setting Up SP Accounting

IBM General Concepts and Procedures for RS/6000, describes a procedure for setting up an accounting system that involves performing the following steps for each AIX system:

- Creating directories
- Creating and updating files
- Enabling accounting
- Adding accounting procedures to **crontabs**

In a parallel environment, where there may be hundreds of nodes, it is essential that these tasks be automated.

The following procedure helps you to tailor your accounting system by defining accounting class identifiers and accounting enabled attributes on an SP system level or on an individual node basis, if necessary. Accounting class identifiers need only be specified when you want more than one accounting class. You can also specify which nodes are to have accounting enabled, however, if all nodes are to be enabled, then only SP Accounting Enabled need be set to true and the **acct_enable** attribute can be left to default on each Node object in the SDR.

In addition, this procedure shows you how to specify exclusive use accounting, and how to specify an Accounting Master other than the default control workstation. It also shows how to set an SP Accounting Active Node Threshold value other than the default value of 80.

You can define or change these values after installation by using SMIT or the **spacctnd** and **spsitenv** commands.

¹ The Resource Manager function has been integrated into the IBM LoadLeveler product.

Accounting and System Partitioning

If you do not partition your SP system or if you run with a single system partition, you can assign any node or the control workstation as the accounting master.

Step 1: Enter System Accounting Information

Use SMIT or the **spacctnd** and **spsitenv** commands to define how accounting is to be done on your SP System. This information is read by the installation scripts, which enter specific details about your site's configuration into the System Data Repository (SDR).

Note

As of PSSP 3.1, the Resource Manager function has been integrated into the IBM LoadLeveler product.

If you are setting up accounting and modifying Node objects in the SDR on an SP system that is already running LoadLeveler, you need to update the **spacct_exclude_enable** keyword on the relevant machine stanzas and reconfigure LoadLeveler.

If you are only changing the SP object in the SDR on an SP system that is already running Resource Manager (to enable exclusive use accounting, for example), then you must reconfigure the Resource Manager using the **jm_config** command.

Using SMIT

TYPE `smit enter_data`

- The Enter Database Information Window appears.

SELECT Site Environment Information

- The Site Environment Information menu appears.

TYPE The data in the fields, as needed.

Using spsitenv

The following example enables accounting as the default for all nodes and specifies 2 as the accounting master. It also enables exclusive use accounting.

```
spsitenv acct_master=2 spacct_enable=true spacct_exclude_enable=true
```

Accounting information on the `smit enter_data` panel includes:

SP Accounting Enabled

Use this field to specify whether by default, accounting is enabled or disabled on all nodes that have an accounting enabled attribute of **default**. A value of **true** indicates that accounting is enabled on all nodes with an accounting enabled attribute of **default**. A value of **false** indicates that accounting is disabled on all nodes that have an accounting enabled attribute of **default**. The initial value for this attribute is **false**.

This sets the **spacct_enable** attribute in the SP object of the SDR. You can specify this attribute by using the **spsitenv** command instead of SMIT.

SP Accounting Active Node Threshold

Use this field to specify the minimum percentage of nodes for which accounting data must be present. At the time when the new **crunacct** script runs, data must be available from at least this percentage of nodes in order for processing to continue. The default value for this attribute is **80**.

This sets the **spacct_actnode_thresh** attribute in the SP object of the SDR. You can specify this attribute when using the **spsitenv** command instead of SMIT.

SP Exclusive Use Accounting Enabled

Use this field to specify whether to activate start and end job records, and thus chargefee records for jobs requesting the exclusive use of a node. The default value for this attribute is **false**.

This field sets the **spacct_exclude_enable** attribute in the SP object of the SDR. You can specify this attribute when using the **spsitenv** command instead of SMIT.

A value of **true** specifies that exclusive use accounting is enabled on all nodes where the **acct_exclude_enable** attribute is set to true. You must check each node's **acct_exclude_enable** attribute for account processing. A value of **false** (the default) specifies that exclusive use accounting is disabled on the node.

Accounting Master

Use this field to specify which node is to act as the accounting master. This is the node that executes **crunacct**. The default value for this attribute is **0** (the control workstation) but you can specify any SP node.

This sets the **acct_master** attribute in the SP object of the SDR. You can specify this attribute when using the **spsitenv** command instead of SMIT.

Step 2: Enter Node Accounting Information

Use SMIT or the **spacctnd** command to further define SP accounting at the node level.

Using SMIT

TYPE **smit node_data**

- The Node Database Information menu appears.

SELECT Accounting Information

- The Accounting Information menu appears.

TYPE The data in the fields, as needed.

Accounting information on this panel includes:

Start Frame

Enter the frame number that corresponds to the nodes required for this operation. Specify a value between 1 and 128 inclusive.

Start Slot

Enter the slot number of first node to be used for this operation. Specify a value between 1 and 16 inclusive. Remember that a wide node counts as two slots.

Number of Nodes

Number of nodes to be used for this operation. The node information is added for successive nodes within a frame. If the count of nodes causes the nodes in a frame to be exhausted the operation continues in the next sequential frame. Specify a value between 1 and 1024 inclusive.

Accounting Class Identifier

Use this field to change the accounting class. All nodes with the same string value constitute a class for purposes of grouping and merging accounting data. All nodes will have an initial accounting class identifier of **default**.

This sets the **acct_class_id** attribute in the Node object of the SDR. You can specify this attribute when using the **spacctnd** command instead of SMIT.

You only need to specify a value for this field when you want to change the accounting class.

Accounting Enabled

Use this field to indicate whether accounting is enabled or disabled for a node. A value of **true** indicates that accounting is enabled for the node. A value of **false** indicates that accounting is disabled for the node. All nodes have an initial value of **default** assigned to this attribute. A value of **default** indicates that the node is enabled for accounting based on your entry in the SP Accounting Enabled field in the SMIT Site Environment panel in Step 1.

This sets the **acct_enable** attribute in the Node object of the SDR. You can specify this attribute when using the **spacctnd** command instead of SMIT.

Exclusive Use Accounting Enabled

Use this type field to indicate whether accounting start and end job records, and thus chargefee records, are processed for jobs having exclusive use of the node. A value of **true** indicates that exclusive use accounting is enabled and start and end job records will be generated. A value of **false** indicates that exclusive use accounting is not enabled and start and end job records will not be generated. All nodes have an initial value of **false** assigned to this attribute.

This sets the **acct_exclude_enable** attribute in the Node object of the SDR. You can specify this attribute when using the **spacctnd** command instead of SMIT.

Accounting Job Charge Value

This value is used to determine the number of charge fee units to charge a user for exclusive use of the node and its value is in units of *seconds per charge fee unit*. All nodes have an initial value of **1.0** assigned to this attribute.

This sets the **acct_job_charge** attribute in the Node object of the SDR. You can specify this attribute when using the **spacctnd** command instead of SMIT.

Using spacctnd

The following example adds accounting information for a system with one frame and 16 nodes. Accounting and exclusive use accounting is to be enabled for each node and 60 seconds of exclusive use by a user is to constitute one charge fee unit.

```
spacctnd -e true -j 60.0 -x true 1 1 16
```

Additional Accounting Setup Tasks

Depending on your environment, you may also wish to perform these tasks:

- Defining the file systems for which disk accounting is to be done.
- Defining the print queues for which printer accounting is to be done.
- Defining how the all print jobs owned by the print manager id (**prtld**) in Secure Mode are to be handled for accounting purposes.
- Disk accounting may be enabled for each file system as it is created. Alternatively, if many file systems already exist, then disk accounting may be turned on or off by executing the **chfs -t'yes' <filesystem>** command on the nodes.

Accounting Installation

The accounting installation phase is performed automatically when you select the Accounting Support option in **ssp.sysman** during the **installp** process. (Refer to the *PSSP: Installation and Migration Guide* .) This accounting installation process is also performed when you update accounting using **spsitenv**.

The accounting installation process:

1. Creates all necessary directories and files on the nodes for which accounting is enabled and on the **acct_master** node.
2. Adds the **startup** command to the **/etc/rc** file on the nodes for which accounting is enabled.
3. Creates the **/var/adm/acct/nite/jobcharge** file on each node for which accounting is enabled. This file contains the job charge value previously defined for the node.
4. The **holidays** file (**/etc/acct/holidays**) is placed in the **user.admin** file collection and is available on all nodes. Its source is the control workstation. If you do not use file collections, propagate this file, using whatever means is available in your environment, to all nodes in the SP System.
Note: You may have to update the **holidays** file to reflect the correct information for the current year.
5. Update the **crontabs** file to schedule the accounting processes. See "Accounting Template for crontabs" on page 122.
6. Initialize the login and process account data files. Enter:

```
/usr/sbin/acct/nulladm /var/adm/wtmp /var/adm/pacct
```

Accounting Template for crontabs

A default **crontab** template file, describing when each accounting command should be run, is shipped as **/usr/lpp/ssp/config/cron_template** and distributed via a file collection. If you wish to run your accounting system on a different schedule, make the appropriate changes to this template. When booting, each node for which accounting is enabled, ensures that its **crontab** file is updated using this template. If the template does not exist, it is assumed that you use file collections to distribute **crontab** and nothing is done.

The **crontab** template file, shown below, contains a partial **crontab** entry. Each line begins with a keyword specifying a command used by the accounting process.

```
# This is the crontab template file. It is read by acct_config
# to update the root crontab file with accounting-related entries.

DODISK    0 1 * * 4
CHKPACCT  5 * * * *
NRUNACCT  0 2 * * 1-6
CRUNACCT  0 4 * * 1-6
CMONACCT  15 5 1 * *
```

Figure 7. Accounting Template for crontabs

DODISK and **CHKPACCT** are standard **crontab** keywords. **NRUNACCT**, **CRUNACCT**, and **CMONACCT** are variations of standard AIX accounting commands discussed earlier in this section. Refer to Chapter 11, “Maintaining the crontabs File” on page 127 for more information on **crontabs**.

Accounting Files

All accounting data, report, and summary files reside in the **/var/adm** directory. The accounting data files belong to members of the **adm** group, and all active data files (such as **wtmp** and **pacct**) reside in the **adm** home directory, **/var/adm**.

Note

The System Administrator should monitor the capacity of the **/var** filesystem when exclusive use accounting is enabled because large **/var/adm/pacct** files may be created and accounting will save the **excl_Spacct** files in the **/var/adm/acct/nite** directory. This may require the administrator to periodically increase the **/var** filesystem size or remove unnecessary **excl_Spacct** files. The System Administrator may edit the **/usr/lpp/ssp/bin/nrunacct** script in the **CLEANUP** section to automatically remove **excl_SPacct** files.

Data Files

Data files are:

/var/adm/acct/nite/dacct

Contains disk total accounting records, created by the **dodisk** command.

/var/adm/fee

Contains output from the **chargefee** command, in ASCII **tacct** records.

`/var/adm/pacct`

Active process accounting file.

`/var/adm/wtmp`

Active login accounting file.

`/var/adm/Spaccti.yyyymmdd`

Contains process accounting files for *yyymmdd* during the execution of the **nrunacct** command.

Report and Summary Files

Report and summary files reside in the **`/var/adm/acct`** subdirectory. You must create the following subdirectories before the accounting system is enabled.

`/var/adm/acct/nite`

Contains files that **nrunacct** and **crunacct** command use in daily accounting.

`/var/adm/acct/sum`

Contains the cumulative summary files that **nrunacct** and **crunacct** update daily.

`/var/adm/cacct`

Contains files that the **crunacct** command uses in daily accounting.

nrunacct Command Files

The following report and summary files, produced by the **nrunacct** command, are of particular interest.

`/var/adm/nsiteacct`

Contains site-specific accounting procedures for node accounting.

`/var/adm/acct/nite/lineuseyyymmdd`

Contains usage statistics for each terminal line on the system. This report is especially useful for detecting bad lines. If the ratio between the number of logouts and logins exceeds 3 to 1, it is possible that a line is failing.

`/var/adm/acct/nite/daytacctyyymmdd`

Total accounting file for the previous day.

`/var/adm/acct/sum/daycmsyyymmdd`

Contains the daily command summary.

`/var/adm/acct/sum/loginlogyyymmdd`

Contains a record of the last time each user ID was used.

`/var/adm/acct/nite/rebootsyyymmdd`

Contains beginning and ending dates from the **wtmp** file and a listing of system restarts.

crunacct Command Files

`/var/adm/csiteacct`

Contains site-specific accounting procedures for cluster accounting.

These files are copied to the **acct_master** node during **crunacct** execution.

`/var/adm/cacct/node/nite/lineuseyyymmdd`

`/var/adm/cacct/node/nite/daytacctyyymmdd`

`/var/adm/cacct/node/nite/rebootsyyymmdd`

/var/adm/cacct/node/sum/daycmsyyyymmdd
/var/adm/cacct/node/sum/loginlogyyyymmdd

The *node* level files are merged to become:

/var/adm/cacct/sum/class/tacctyyyymmdd.
/var/adm/cacct/sum/class/daycmsyyyymmdd.

which in turn update cumulative summary files:

/var/adm/cacct/sum/class/tacct file..
/var/adm/cacct/sum/class/cms file..

Files in the */var/adm/acct/nite* Directory

active

Used by the **nrunacct** command to record progress and to print warning and error messages. The file **active.yyyyymmdd** is a copy of the active file made by **nrunacct** after it detects an error.

jobcharge

Contains a floating point number used to determine the number of charge fee units to charge a user for exclusive use of the node and its value is in units of *seconds per charge fee unit*.

jobrecs

Contains *start job* and *end job* records used for accounting on jobs that make exclusive use of nodes.

ptacct.yyyyymmdd

Contains summary version of **pacct** files.

ctmp

Contains connect session records during **nrunacct**

accterr

Contains standard error messages produced during the execution of the **nrunacct** command.

lastdate

Indicates the last day the **nrunacct** command executed, in **yyyymmdd** format.

lock

Used to control serial use of the **nrunacct** command.

statefileyyyymmdd

Used to record the current state during execution of the **nrunacct** command.

wtmperror

Contains **wtmpfix** error messages.

owtmp.

Contains the previous day's **wtmp** file.

Files in the `/var/adm/cacct/` Directory

active

Used by the **crunacct** command to record progress and to print warning and error messages. The file **active.yyyymmdd** is a copy of the active file made by **crunacct** after it detects an error.

lastdate

Indicates the last day the **crunacct** command executed, in **yyymmdd** format.

lock

Used to control serial use of the **crunacct** command.

statefile*yyymmdd*

Used to record the current state during execution of the **crunacct** command.

lastcycle

Indicates the date of the last successfully completed **crunacct** cycle.

fiscal_periods

Defines the start date of each fiscal period.

Files in the `/var/adm/cacct/sum` Directory

<class>/cms

Daily command summary.

loginlog

File listing current login records.

rprt*yyymmdd*

Saved output of the **cprdaily** routine executed in **crunacct**.

<class>/tacct

Process account summaries.

Files in the `/var/adm/cacct/fiscal` Directory

/var/adm/cacct/fiscal/class/cms*mm*

Total class summary file for the month, in binary format.

/var/adm/cacct/fiscal/class/tacct*mm*

Total accounting file for month, in binary format.

/var/adm/cacct/fiscal/fiscrpt*mm*

Report similar to that of the **prdaily** command for month, in binary format.

Chapter 11. Maintaining the crontabs File

The SP system uses the **crontabs** file to periodically update file collections and clean up log files. The installation process appends entries to the **crontabs** files in **/var/spool/cron/crontabs/root** on the control workstation and each of the nodes. The files contain different entries depending on their location.

- Control workstation
 - Merge accounting information
 - Clean up the logs
- Processor nodes
 - Update the **sup.admin**, **user.admin** and **node.root** file collections
 - Merge accounting information
 - Clean up the logs

The following example shows these entries on a control workstation:

```
0 * * * * /usr/lpp/ssp/bin/dev/null 2>/dev/null
0 0 * * * /usr/lpp/ssp/bin/cleanup.logs.ws
0 4 * * 1-6/usr/lpp/ssp/bin/crunacct 2>/var/adm/cacct/nite/accterr
15 5 1 * * /usr/lpp/ssp/bin/cmonacct
```

The default is to run these processes hourly, in a staggered sequence. You can customize this for your environment to make any of these processes run more or less frequently. You can also add your own entries to these files for other processes you want automated.

Updating crontabs on the SP Nodes

If you want the same **crontab** file on all SP nodes, one way to update it across the whole SP System is:

1. Get the current **crontab** from one of the nodes:

```
rsh hostname crontab -l > /tmp/mycrontab
```

2. Edit the file:

```
vi /tmp/mycrontab
```

3. Propagate the changed **crontabs** file to all SP nodes:

```
dsh -a rcp root@mynode:/tmp/mycrontab /tmp/mycrontab
dsh -a "crontab < /tmp/mycrontab"
```

If you change the default configuration to modify the hierarchy of file collection servers, you need to modify the **crontabs** files to reflect the new relationship. See "Modifying the File Collection Hierarchy" on page 63 for more information. If you create your own file collections, you will also need to add them to the **supper update** commands in the **crontabs** file to keep them on schedule with the other upgrades.

Security on the SP System

Chapter 12. Security Features of the SP System

This chapter provides an introduction to general security concepts and terminology, as well as an overview of the way authentication services works. In addition, this chapter discusses how to install and configure your chosen distributed authentication service. First, consult the book *IBM RS/6000 SP: Planning Volume 2, Control Workstation and Software Environment* to plan which authentication service is right for your environment. Other topics include how to administer security, how to use the AIX remote commands, and how to use AFS as an alternative to PSSP authentication servers.

For more information about installing and configuring authentication services, see the book *PSSP: Installation and Migration Guide*. See the book *PSSP: Diagnosis Guide* for more information about diagnosing authentication problems.

Basic Security Concepts

There are primarily two facets of security that an administrator needs to address; how to keep non-authorized individuals from accessing the system and how to ensure that authorized individuals access only those areas for which they have approved access. The second facet involves protecting the resources of the system from unauthorized access. In order to address these two facets of security, it is important to first understand the different levels of security.

The first level of security is identifying the individual or client attempting to access your system. The second level involves authenticating the individual, usually by using an ID and password. The third level of security involves authorizing an individual or client to access certain parts of the system for which they have authorization.

Understanding Authentication

Administering an SP system from a single point-of-control requires the use of administrative applications that are based upon the client-server programming model. Typically, a client-side command-line or GUI interface is used to perform an administrative task on one or more network-attached nodes or workstations, in parallel or serially. Where privileged applications such as these communicate across networks, security can be compromised unless a reliable means is used to restrict access to authorized users. Accomplishing this authorization first requires authentication, which is the validation of the identities of client and server. Authentication is also the foundation for providing other security features such as accountability, discretionary access controls, and least privilege.

An SP system implementation, for example, can consist of SP nodes with internet connections, a control workstation, and connections to other clustered workstations. Any of these system elements can be vulnerable to a security compromise. All networked services, as well as clients who may need to provide sensitive data to services, must be able to restrict access to those services according to a meaningful security policy. Such access restrictions can be successful only by the reliable authentication of the identities of clients and servers.

The PSSP software includes two types of administrative programs that benefit from these security features:

- Applications that manage the hardware resources of the SP system
- Applications that provide secure remote execution of SP administration commands. These include support for delegation of root user tasks to other users without distributing root's password.

Overview of Distributed Client/Server Environments

Distributed client/server environments represent a way to quickly and transparently deliver information to users from various different sources and locations. This technology can help companies reduce cost and time as well as improve quality and customer satisfaction. The client/server model implies cooperative and distributed processing and relies on a message-based communication between a requester (or client) that asks for a specific service and a responder (or a server) that provides the information. The message exchange can be synchronous or asynchronous.

The most basic form of client/server computing has only two pieces: a client process and a server process connected through a network. The server process is the provider of services and the client is the consumer of services. Clients usually manage the user-interface portion of the application, while server programs generally receive requests from client programs, execute the specified action, and dispatch the response to the clients.

Suggested Reading

- *IBM Distributed Computing Environment for AIX: Introduction to Distributed Computing Environment* provides an overview of DCE and serves as an introduction to the rest of the DCE documentation. It also contains a glossary of terms used in the DCE documentation.
- *IBM Distributed Computing Environment for AIX: Up and Running* provides information on the packaging, installation, and configuration of DCE clients and servers.
- *IBM Distributed Computing Environment for AIX: Release Notes* contains last minute information that could not be included in other documents
- *IBM Distributed Computing Environment for AIX: Administration Guide* provides conceptual and task-oriented information for the DCE administrator. The first part is an overview, which describes administering DCE as a whole, including planning and configuring information. The second part, describes how to manage RPC, CDS, DTS, and Security Service.
- *Distributed Computing Environment for AIX: Administration Command Reference* provides reference material for DCE commands, including administrative commands.
- *IBM Distributed Computing Environment for AIX: DFS Administration Guide and Reference* provides concepts and procedures for managing the Distributed File Service (DFS) in a DCE cell. It provides detailed reference information about the complete syntax and use of each DFS command and configuration file.
- *IBM Distributed Computing Environment for AIX: Network File System to Distributed File System Authenticating Gateway Guide* provides information on how to associate an NFS client request with the DFS filesystem.

- *IBM Distributed Computing Environment for AIX: Application Development Guide – Introduction and Style Guide* provides information for the distributed application programmer. It provides conceptual and task-oriented information for developing an application using DCE and describes programming with DCE in general, using various components and facilities.
- *IBM Distributed Computing Environment for AIX: Application Development Guide – Core Components* describes the APIs for the various DCE components.
- *IBM Distributed Computing Environment for AIX: Application Development Guide – Directory Services* describes the DCE naming and access to CDS through XDS, the use of the X/Open Directory Service Interface, and the Object Classification Tables and other information on XDS, X500 Directory, and GDS objects.
- *IBM Distributed Computing Environment for AIX: Application Development Reference* provides reference material for the DCE programming interfaces and has command references for commands needed by the DCE programmer.

Terminology

Before proceeding, you should be familiar with terms in the following list. Note that these terms mostly apply to Kerberos V4, although some terms also apply to Kerberos V5. Some terms, however, may be defined differently in DCE documentation. For a complete understanding of DCE terminology, refer to the appropriate DCE documentation.

ACL	Access Control List. A list that defines who has permission to access certain services; that is, for whom a server may perform certain tasks.
AFS	A distributed file system that uses Kerberos authentication and includes authentication services that can be used by an SP system.
Authentication	The process of validating the identity of either a user of a service or the service itself.
Authentication Database	A set of files containing the names and authentication information of all principals within a realm. An authentication realm has one primary database and may have multiple secondary databases. Secondary databases are backup copies of the primary database and may be provided to improve performance or availability.
Authenticator	An authentication protocol string created each time authentication occurs and sent with the ticket to the server. It contains a time-stamp encrypted in the session key that can reliably show that the authentication request actually came from the client identified in the ticket.
Authorization	The process of obtaining permission to access resources or perform tasks. In SP security services, authorization is based on the principal identifier.
Cell	An independently administered collection of file server and client machines running AFS. An AFS cell is equivalent to a Kerberos realm.

Client	The user of the shared services of a server. This term may refer to a program or to the person who invoked it. For authentication purposes it is a principal identifier registered in the authentication database.
Credentials	A protocol message, or part thereof, containing a ticket and an authenticator supplied by a client and used by a server to verify the client's identity. The message can contain additional information used by the server to verify its identity to the client.
Data Encryption Standard (DES)	The secret-key (also known as "private-key") encryption algorithm that is used by Kerberos.
Domain	A collection of systems over which an administrator exercises control.
Instance	A qualifier for a principal name. For services, an instance represents a particular occurrence of the server. For users, an instance allows a single user to assume additional (or alternate) roles with different authority.
Kerberos	A service for authenticating users in a network environment.
Key	A value used to encrypt protocol strings used for authentication. Principal's private keys are stored in the authentication database. Session keys are contained (encrypted) in tickets and other protocol strings.
Kerberos Master Key	The key derived from the Kerberos Master Password supplied initially by the administrator when the primary SP authentication server is created. This key is saved in the <i>/k</i> file, from whence the Kerberos daemons read it (in lieu of prompting for a password). It can also be read by certain database utility commands for the same purpose.
Kerberos Master Password	The password the administrator supplies when initializing the primary authentication server.
Principal	An entity whose identifier and key are maintained in the authentication database. A principal can represent a user or a service.
Realm	A domain which shares an authentication database and servers. There is a single name-space for principal name/instance pairs within a realm. A realm is also a logical collection of clients and servers registered in the database.
Server	A functional unit (usually a daemon program and child processes) running on a particular host that provides shared services to users over a network.
Service	The name of a principal whose identity is assumed by a server for purposes of authentication. Multiple servers can use the same service name.

Server Key File	(Also referred to as a srvtab) A file containing the names and private keys of the local instances of services. It is accessible only to processes that run under the UID of the user owning the server daemons. On an SP system all services run as root.
Session Key	A temporary key supplied by an authentication server to clients and servers, that is used to encrypt parts of authentication protocol messages. Its lifetime is the same as the ticket with which it is created.
Ticket	An encrypted protocol message used to pass the identity of a user from a client to a server. Tickets are created by the Kerberos authentication server and cached in disk files on the client's system.
Ticket-granting-ticket	The initial ticket obtained by a user. It is used by client programs to obtain additional tickets for authentication with application services.

Conceptual Information Regarding Kerberos V4

PSSP 3.1 offers you Kerberos V4 as an authentication service that you can install and configure and provides support for the use of Kerberos V5 administered through DCE. This section provides detailed conceptual information regarding Kerberos V4 and some of the information may also apply, in a general sense, to Kerberos V5 (DCE). For specific conceptual information regarding Kerberos V5 (DCE), refer to the appropriate DCE documentation.

Kerberos uses a Kerberos server to achieve mutual authentication between a client and a server in a client/server environment.

Kerberos Components

In general, Kerberos consists of systems, objects, tickets, and keys.

Systems

The systems that make up a Kerberos environment include a:

- Client which makes a request to the server
- Server which provides service to the client's request
- Server which provides authentication information for both the client and the server.

Objects

The objects that make up the Kerberos environment include principals, instances, realms, tickets, and keys:

Principals

Kerberos principals are either users who use authentication services to run the Kerberos-authenticated applications supplied with the SP system or the individual instances of the servers that run on SP nodes, the control workstation, and on other IBM RS/6000 workstations that have network connections to the SP system.

User Principals for SP System Management – An implementation of the SP system must have at least one user principal defined. This user is the authentication database administrator, who must be defined first so that other principals can be added later. When AFS authentication servers are being used, the AFS administrator ID already exists when the SP authentication services are initialized. When PSSP authentication servers are being used, one of the steps included in setting up the authentication services is the creation of a principal whose identifier includes the **admin** instance. It is suggested, but not essential, that the first authentication administrator also be the root user. Various installation tasks performed by root, or other users with UID 0, require the Kerberos authority to add service principals to the authentication database. At sometime prior to invoking the **setup_server** command to complete setup of the control workstation and boot server nodes, a user with UID 0 must be defined as a Kerberos database administrator with authority to add other principals. This is because the service principals used by Kerberos-authenticated applications are all defined automatically by **setup_server** during installation and customization. Apart from this installation requirement, you may use other user principal identities to perform system administration (and authentication administration) tasks.

Service Principals Used by PSSP Components – Two service names are used by the Kerberos-authenticated applications in an SP system:

1. **hardmon**, used by the System Monitor daemon on the control workstation and by logging daemons
2. **rcmd**, used by **sysctl**

By convention, a Kerberos V4 server runs as a Kerberos principal whose identifier is formed from the service name and the short host name of the node on which the server runs, converted to lowercase. Different principal identifiers thus indicate the target network address of the different providers of each service. The authentication support in the SP system expands the Kerberos implementation to support workstations with multiple network interfaces as servers. This provides instances for all network interfaces rather than just the standard host name. Kerberos works with the resolved network names, rather than any aliases you may use locally on your hosts.

The **hardmon** daemon runs only on the control workstation. The SP logging daemon, **splogd**, can run on other IBM RS/6000 workstations. Therefore, for each (short) network interface name on these workstations, a service principal is created with the name **hardmon** and the network name as the instance.

The remote commands can be run from, or to, any IBM RS/6000 host on which the SP system authenticated client services (**ssp.clients**) are installed. Therefore, for each (short) network interface name on all SP nodes, the control workstation, and other client systems, a service principal is created with the name **rcmd** and the network name as the instance.

Instances

Instances can be on a client machine or on a server machine. An instance on a client machine is defined as the level of authorization the principal user has. An instance on a server machine is defined as the hostname of the server machine. Kerberos uses the following notation to refer to principals and instances:

`principal.instance`

Realms

The authentication realm is the set of systems that is served by a single logical authentication database. A realm can include any combination of one or more SP systems, IBM RS/6000 workstations, and other systems/workstations. The authentication database for a realm is administered as a single entity: principals are defined across the systems within a realm and principal identifiers must be unique within a realm.

There are options for defining authentication realms within a network of systems:

- Each system may be its own realm
- A single realm may include all of the systems in a network
- Systems may be grouped into realms with multiple realms occurring within a network

Generally, systems that are administered together, or by the same set of administrators, should be grouped into a single authentication realm. For example, one or more SP systems and their associated workstations could form a single authentication realm. Within an SP system, all SP nodes are automatically added to the control workstation's local realm during installation.

If multiple realms are configured within a network, the realm names should be unique. During setup of the SP authentication services, the name of the realm defaults to the local system's network domain name converted to uppercase. If a different realm name is going to be used, the local system's realm name must be supplied by the administrator in the `/etc/krb.conf` file. For information on the format of this file, refer to the *PSSP: Command and Technical Reference*.

If you are using AFS authentication servers, the entire SP system must be in the same cell. The authentication realm known to the SP authentication services is the local AFS cell. The realm name is set to the AFS cell name converted to uppercase.

Although a realm is served by a single logical authentication database, multiple copies of the database may exist on different workstations within a realm. One must be the primary database in a realm; others are secondary servers.

Tickets

Kerberos tickets are encrypted byte strings returned to clients by an authentication server. Tickets contain the client principal's identifier, a time-stamp and ticket lifetime used to determine the expiration time, the session key, and other information, all encrypted in the private key of the service instance with whom the client wishes to authenticate. The client cannot understand the key; it simply forwards it to the service with an authenticator in an internet protocol packet, using either UDP or TCP socket communications. In the case of a ticket-granting-ticket, that service is the Ticket Granting Service provided by the **kerberos** daemon. Other

tickets are service-tickets, encrypted in the private key of one particular instance of an application service.

- Ticket-granting-ticket

Obtained by the **k4init** or **rcmdtgt** command, which saves it by creating the current ticket cache file, over-writing one that already exists. As long as the current ticket cache file contains an unexpired ticket-granting-ticket, the owner of the file can invoke application client programs that require Kerberos V4 credentials.

- Service-ticket

Obtained by an application client, such as the **sysctl** command, which adds it to the current ticket cache file. As long as the current ticket cache file contains an unexpired ticket for a particular service instance, it can be used by any process that runs under the UID of the file owner to submit requests to that service instance.

Ticket Cache Files: The caching of tickets takes place automatically in disk files on the client system that is the workstation or SP node on which the client user is logged in. These files are created with UNIX permissions set so they can be accessed only by the owner, for they contain sensitive information. Any process using the tickets can authenticate as the owner of the file.

A user can have tickets cached in any number of files simultaneously, and may have reason to do so. If you have multiple logins, run background processes, or use a shared UID, you must do so to avoid destruction of tickets still in use by other processes. This is accomplished by setting the **KRBTKFILE** environment variable to the path name of a unique ticket cache file prior to using authentication services for each login session or background process. If you do not set **KRBTKFILE**, then your tickets will be cached in your default file: **/tmp/tktuid**.

The first ticket stored in the file is always the ticket-granting-ticket. The file is created by the **k4init**, **ksrvtgt**, or **rcmdtgt** command; and when the TGT is stored, all previously issued tickets in the file are deleted. The file contains, in addition to the TGT, any number of other tickets for application services. There will be one for each service instance with which you have communicated using that particular current ticket cache file. They are obtained by the client commands for the various Kerberos-authenticated services, such as **rcp** and **rsh** (and indirectly by SP system administration tools that invoke them).

Kerberos does not automatically delete tickets after they expire. Users must either explicitly destroy their ticket cache files or replace them by re-authenticating (by obtaining a fresh TGT). Tickets are never removed individually from the ticket cache file. The **k4destroy** command eliminates the entire current ticket cache file. Ticket cache files other than the current file are untouched, however.

Ticket-lifetimes: The tickets that you obtain when you issue the **k4init** command are valid only for a limited time. The same limit applies also to service tickets obtained by client programs that you execute. In most circumstances, you probably will not be concerned about how this limit is determined and applied. However, when you run **k4init**, you have the option of requesting a ticket with a specific lifetime. The maximum lifetime you may specify, which is also the default value, was assigned when your principal was registered in the authentication database, or it was changed subsequently by the database administrator. The minimum ticket

lifetime is five minutes; the maximum lifetime can be set to one of several discrete values from five minutes to thirty days. The ticket lifetimes of service tickets are always the same as the lifetime of the ticket-granting-ticket used to obtain them. It is the TGT held in the current ticket cache file owned by the user who invokes the application client command.

The external representation of the ticket lifetime is rather inconsistent among the various user and administrator commands. Some commands represent the one-byte ticket lifetime value as a decimal integer between 0 and 255. This is the representation used by the administrator when creating or modifying a principal using the **kdb_edit** command. It is also the format of the ticket life field displayed by the **get** subcommand of **kadmin**. The output of the **k4list** command shows the actual ticket creation and expiration as standard date and time stamps. If you specify **-l** to request a specific ticket lifetime when you run **k4init**, you are prompted to enter the value as a number of minutes. The value you enter is then rounded up to the next higher discrete lifetime interval or the maximum ticket lifetime allowed for your principal, whichever is less, and then mapped to a one-byte value between 1 and 191.

The following table shows the ticket lifetimes that correspond to the encoding used internally by Kerberos. The values on the left of each column are the one-byte encodings; on the right, the lifetime interval in days, hours, and minutes.

Ticket lifetimes from 0 to 128 represent multiples of 5 minutes, for example:

0-1	00:05	12	01:00	24	02:00	36	03:00	48	04:00
60	05:00	72	06:00	84	07:00	96	08:00	108	09:00
120	10:00	125	10:25	126	10:30	127	10:35	128	10:40

Lifetimes from 129 to 255 represent intervals of from 11:24 to 30 days:

129	11:24	130	12:11	131	13:02	132	13:56	133	14:54
134	15:55	135	17:01	136	18:12	137	19:28	138	20:48
139	22:15	140	23:47	141	1d+01:26	142	1d+03:11	143	1d+05:04
144	1d+07:05	145	1d+09:14	146	1d+11:32	147	1d+13:59	148	1d+16:37
149	1d+19:25	150	1d+22:26	151	2d+01:38	152	2d+05:04	153	2d+08:44
154	2d+12:40	155	2d+16:51	156	2d+21:21	157	3d+02:08	158	3d+07:16
159	3d+12:45	160	3d+18:36	161	4d+00:52	162	4d+07:34	163	4d+14:44
164	4d+22:23	165	5d+06:35	166	5d+15:20	167	6d+00:41	168	6d+10:42
169	6d+21:23	170	7d+08:50	171	7d+21:03	172	8d+10:08	173	9d+00:06
174	9d+15:03	175	10d+07:01	176	11d+00:06	177	11d+18:22	178	12d+13:53
179	13d+10:46	180	14d+09:05	181	15d+08:56	182	16d+10:27	183	17d+13:44
184	18d+18:53	185	20d+02:04	186	21d+11:24	187	22d+23:02	188	24d+13:08
189	26d+05:52	190	28d+01:26	191+	30d				

Keys

Keys are another Kerberos component. They are similar to passwords in that they are used for encrypting the tickets being sent back and forth between the client and server machines. “Kerberos Authentication” on page 140 explains how keys are used in the authentication process.

Kerberos Authentication

SP authentication for Kerberos V4 is based on 4 of the Kerberos authentication service developed at MIT.

SP authentication services use the authentication database and the server to:

- Initially authenticate the identity of the user
- Provide the information through which a server can authenticate the identity of a client (user) in a distributed environment

Kerberos servers function as a trusted third party to authenticate the identities of users and distribute encryption keys to clients and servers, known as *principals*. Every user and every service instance on a particular host is a principal in the Kerberos database. Each Kerberos principal has an identifier, a password, an expiration date, and a maximum ticket lifetime registered in the authentication database. Kerberos does not store the clear-text password, but a 64-bit binary value derived from it using a one-way hash. Database information is also encrypted in the Kerberos master key, known only to the database administrator. Passwords are never sent across the network in clear text-form, either for user authentication or during database administration.

Overview of Kerberos Interaction with Client and Server

On the authentication server system there are two daemons that access the database. The **kadmin** daemon provides network access for database administration via the **kpasswd** and **kadmin** commands. The **kerberos** daemon is the actual authentication server. It is sometimes referred to as the “Key Distribution Center (KDC)” because of its role in distributing session keys. The KDC actually consists of two separate functions: the Authentication Service, which generates ticket-granting-tickets based on authentication using the client user's private key; and the Ticket Granting Service (TGS), which generates service tickets, based on authentication using a ticket-granting-ticket.

The underlying mechanism for accomplishing the authentication is a ticket scheme, consisting of the following steps, illustrated in Figure 8 on page 141.

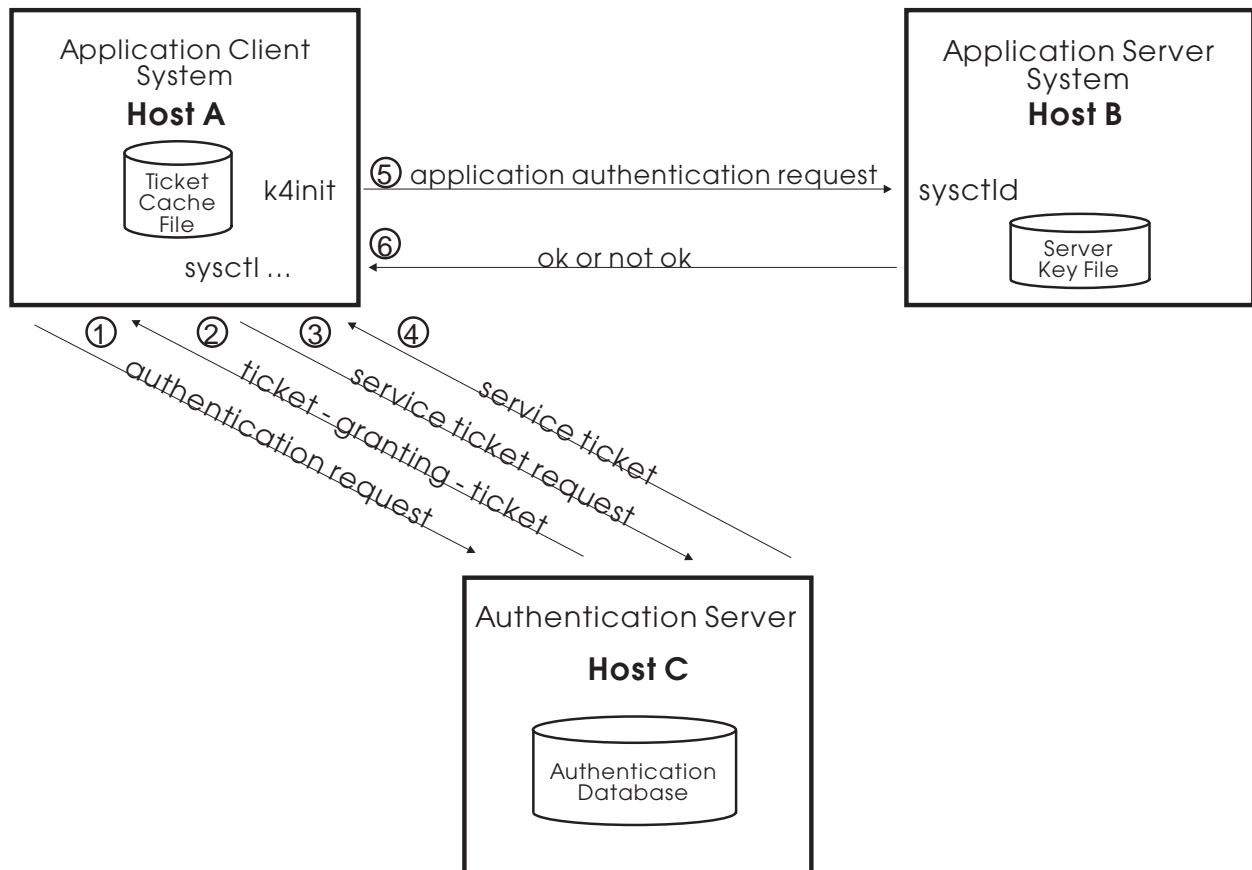


Figure 8. An Overview of Kerberos Authentication

1. The client process, usually as the result of a user invoking the **k4init** command, sends a message to the Authentication Service containing its principal identifier and the identifier of the Ticket Granting Service (a predefined Kerberos principal). This message is not encrypted. The Authentication Service generates a random session key to be shared by the client and the TGS and creates a ticket for the client to use with the Ticket Granting Service. It contains the client identifier and the session key, as well as fields defining the ticket's valid lifetime. The ticket is encrypted in the private key of the Ticket Granting Service. A return message is built, containing the key, the client's identifier, and the ticket-granting-ticket. The Authentication Service encrypts the message in the private key of the user, a one-way hash of the password.
2. The client program reads the user's password and applies the one-way hash necessary to determine the private key. If the password is correct, the key can be used to successfully decrypt the returned message block containing the session key, the client identifier, and the ticket. When successful, the ticket and the session key are stored in the current ticket cache file.
3. The user invokes an application's client interface, which needs to perform authentication on its first network communication with its server. It creates an authenticator, containing the client identifier and a current time-stamp, and encrypts the authenticator using the session key returned with the ticket-granting-ticket. It sends to the Ticket Granting Service a message consisting of the authenticator, the ticket-granting-ticket, and the principal identifier of the service instance which it wants to use. The TGS reads its private key from the authentication database and uses it to decrypt the ticket,

obtaining a copy of the session key and the identifier of the client to whom the ticket-granting-ticket was issued. Using the session key, it decrypts the authenticator. The TGS verifies that the client's identity is the same, and that the authenticator's time stamp indicates that it was very recently created.

4. Like the Authentication Service, the TGS generates a session key, this one to be shared by the client and the application server. It also generates another ticket, called a service ticket, encrypted in the private key of the application service principal. The ticket is returned to the client in a message encrypted with the old session key (shared by the client and the TGS).
5. The client program uses the original session key to decrypt the message. It extracts the new session key and ticket, and saves them in the ticket cache file. When the client program sends its first application protocol message to the application server, it includes the service ticket and a new authenticator, encrypted in the new session key. This session key was also placed in the ticket by the TGS, so the server can retrieve it when it decrypts the service ticket. In order to do so, the application server obtains its private key from the server key file, `/etc/krb-srvtab` on the server system.
6. The authentication by the application server is performed in the same way as by the TGS. The message returned to the client simply indicates success or failure, and an error message number if failure.

The scenario outlined previously assumes no unexpired service ticket was found in the ticket cache file. The ticket-granting-ticket can be used to get as many different tickets as are needed, without requiring the user to reenter the password.

A client's service ticket can be used to re-authenticate as often as the client and server require. Typically, clients using TCP/IP authenticate once on connection to the server. Users of connectionless protocol should authenticate each message exchange, though less rigorous schemes can be used following an initial authentication, by using the shared session key to encrypt a small piece of known data.

Kerberos and DES

The authentication services use the Data Encryption Standard (DES) algorithm. The DES algorithm is used only as an internal mechanism for authenticating the identities of clients and servers. In order to comply with U.S. export regulations on the DES algorithm, SP authentication services provide a subset of the full MIT Kerberos V4 services. No interface is provided for using the encryption routines directly. No interface is provided to allow the encryption of any portion of service messages that are not directly connected with the authentication of the client and server identities.

For additional information, refer to the **Kerberos** man page in the *PSSP: Command and Technical Reference*.

Conceptual Information Regarding DCE

This section provides some general conceptual information regarding DCE. It does not provide any in-depth detail. For complete conceptual information, refer to the documentation listed in "Suggested Reading" on page 132.

Kerberos V5 is also an authentication method available which is provided through DCE. DCE can be used in the SP environment to issue AIX remote commands. AIX 4.3.1 provides AIX authenticated remote commands which do support Kerberos V5 authentication and Kerberos V5 authentication is part of the DCE product and can be used with the SP software.

Overview of DCE

DCE has established a defacto standard in the client/server arena that is available on all IBM operating systems or platforms, such as AIX, OS2, MVS, VM, and OS/400. It is also available on competitor platforms. DCE takes complete advantage of the client/server paradigm. It offers a set of services and APIs that you can use to build distributed applications and a set of management tools to manage the distributed environment. It can also interoperate with other environments. Some of the benefits provided by DCE are:

- Support of Distributed Applications

DCE provides a high-level, coherent environment for developing and running applications on a distributed system. DCE components fall into two categories; tools for developing distributed applications and services for running distributed applications.

The tools, such as Remote Procedure Call and Threads, assist in the development of an application.

The services, such as the Directory Service, Security Service, and Time Service, provide the services required in a distributed system. These are somewhat similiar to some of the services that an operating system provides in a centralized system.

The components and services provided for the application programmer and the system administrator perform many of the tasks that used to require their own code. With DCE, application programmers do not need to write explicit code that handles the network communications between machines or between distributed applications.

- DCE Services are Integrated and Comprehensive

In addition to providing all the tools and services needed for developing and running distributed applications, DCE components themselves are well integrated. Because many of the DCE components are themselves distributed applications, they use each other's services whenever possible. DCE not only supports the development of distibuted applications, but also includes services that address some of the new problems inherent in the distributed system itself, such as data consistency and clock synchronization. DCE includes management tools for administering all of the services and many aspects of the distributed environment itself.

- DCE Provides Interoperability and Portability Across Heterogeneous Platforms

By design, DCE is oriented toward heterogeneous rather than homogeneous systems. The DCE architecture allows for different operating systems and hardware platforms. Using DCE, a procss running on one workstation can interoperate with a process on a second workstation, even when the two have different hardware or oeprating systems. Therefore, DCE accomodates a wider range of networks, especially networks needing distributed computing for the historical reasons previously listed. Moreover, applications built using DCE are portable to other hardware or operating system platforms that run DCE.

- DCE Support Data Sharing

Another benefit is support of data sharing through the directory service and distributed file service. A user anywhere in the distributed system can share data by placing it in the namespace or in a file, whichever is appropriate for the application. The data is then accessible by authorized users throughout the system.

- DCE Participates in a Global Computing Environment

DCE interacts with the outside world. In addition to supporting cooperation within and between systems, DCE systems can also interoperate with computing environments outside of DCE. In particular, the DCE Directory Service can interoperate with two standard, global directory services (X.500 and Domain Name Service), which allow users from within DCE to access information about the outside world. In this way, DCE participates in a global directory service. One benefit of such participation can be seen in the DCE distributed file system which looks like one global file system and users anywhere in the world can address the same file using the same global name.

DCE Security Service

Distributed computing encourages a free flow of data between nodes, thereby expanding the capabilities of interconnectivity and interoperability. The DCE Security Service is a building block of the DCE core services based on Kerberos technology that provides secure authentication, authorization, and auditing mechanisms for users and distributed client/server applications. For data encryption, DCE uses either Data Encryption Standard (DES) or the Common Data Masking Facility (CDMF). CDMF works with a 40-bit encryption key in contrast to the 52-bit DES key. CDMF is allowed to be exported from the USA, whereas DES underlies certain export restrictions.

Developers can use DCE Security Service to make their distributed client/server applications or products secure. Most of the DCE security is related to the concept of a principal. A principal is an entity that can be authenticated and can engage in trusted communication. A principal usually represents a user, a network service, a particular host, or a cell. The services that make up the security component are:

- Registry Service (RS) — a replicated service that maintains the cell's security database. This database contains entries for accounts, principals, groups, organizations, and administrative policies.
- Authentication Service (AS) — used to verify the identify of principals. It contains a ticket-granting service that grants tickets to these principals and services so they can engage in secure communication.
- Privilege Service (PS) — certifies a principal's credentials that are going to be forwarded in a secure way to DCE servers. The credentials allow the target server to check the principal's access rights to resources.

Installing and Configuring Distributed Authentication Services

This section discusses how to install and configure your chosen distributed authentication service. You can choose to install and configure any or all of the following authentication services:

- Kerberos V4

- Standard AIX
- Distributed Computing Environment (Kerberos V5) — this method cannot be installed automatically but you can install it manually.

You can enable authentication methods for a system partition but you cannot enable authentication methods for individual nodes or the control workstation. All nodes within a single system partition will have the same set of authentication methods enabled. The authentication methods enabled on the control workstation will be the union of all authentication methods enabled for all system partitions plus any other methods already set. There are restrictions on the set of authentication methods that can be enabled for a partition containing releases of PSSP prior to PSSP 3.1. For more information on these restrictions, see “Enabling Authentication Methods” on page 152. There are no restrictions on the setting of authentication methods between different partitions within a single SP system; different system partitions can have different sets of authentication methods enabled.

In order to use the AIX authenticated remote commands within the SP system and to allow the system use of rsh and rcp, select the following:

1. The type of authorization to use for root user access using the authenticated remote commands within each system partition. The options are:
 - Kerberos V4
 - Standard AIX
2. The authentication methods to enable for each system partition. The options are:
 - Kerberos V4
 - Kerberos V5 (provided through DCE)
 - Standard AIX

The use of Kerberos V4 authentication and Kerberos V4 authorization for root user rsh is required in PSSP 3.1.

When determining which authentication method to use for remote commands, AIX examines the order of precedence set by the AIX **chauthent** command. This order determines which authentication method is used when the remote commands are issued on the workstation. This means that if the first method fails authorization, the second method is tried, and so on. The order of precedence, defined as being from the highest level of security to the lowest level of security, is Kerberos V5, Kerberos V4, and Standard AIX.

Overview of Steps

1. Create the authorization files (.rhosts is the file for standard AIX and .klogin is the file for Kerberos V4) using the SMIT fastpath command, **smitty spauth_config** or the pssp command **spsetauth** for the .rhosts file or the **setup_authent** command for the .klogin files. The .rhosts file is also generated or updated, if necessary, by the **spauthconfig** command. The .klogin files are maintained separately from this process. These files contain hostnames of nodes which are allowed root access using the remote commands. This step sets the authorization method for each partition for root access using remote commands. This command is run from **spauthconfig** on the nodes so that with each new network boot, the correct configuration is maintained on the system.

2. Enable selected authentication methods using the SMIT fastpath command, **smitty spauth_config** or the pssp command **chauthpar**. This step sets the sets of authentication methods to be enabled for use. The **chauthpar** command is called to set the authentication methods on each node. Once set, the selected authentication methods will be in effect.

Installing and Configuring Kerberos V4

For PSSP 3.1, you must install and configure Kerberos V4 for all partitions within the SP system. In addition to Kerberos V4, you can also select to install and configure DCE for each system partition.

Setting Up PSSP Authentication

To provide authentication services on your SP System, choose from the following three authentication implementations.

1. The authentication services provided with the SP system, based on MIT Kerberos V4. SP authentication services are designed to provide an easily-installable and configurable Kerberos for the SP system.
2. An existing implementation of MIT Kerberos V4, provided it is compatible with the SP Kerberos-authenticated services.
3. AFS version 3.3 or later.

Once you have installed and initialized authentication on your SP system, you can add other IBM RS/6000 workstations to your authentication realm. Two reasons for adding additional workstations might be:

1. If you are not using AFS authentication servers, you may want the added reliability of one or more secondary SP authentication servers. Providing secondary servers is particularly important if your SP system has to provide high availability with no single points of failure.
2. If you also want to install the SP authenticated services on office workstations. This will help you to manage the SP system, without having to login to the control workstation or the SP nodes directly.

When you use the SP authentication server support in the **ssp.authent** option, you may choose to have more than one server system. If you configure your local authentication realm with more than one server, you must designate one as the primary server. All others are secondary servers. Only the primary server has the administrative daemon, **kadmind**, that manages the content of the authentication database. The databases used by the secondary authentication servers are copies of the primary database and are updated periodically from the primary authentication server.

Additional servers can provide for greater reliability, since the authentication service will try all servers listed in the configuration file before failing a service request. If there are network problems, or if your primary server system fails, authentication requests can be handled as long as one of the configured servers is active and accessible. There may also be performance reasons to configure a secondary server. Only when there is an authentication server running on the control workstation can the switch be used for authentication protocol traffic to and from the SP nodes. In configurations where you are integrating the SP system into an existing authentication realm, where the primary authentication server is already on another workstation, you can make this possible by setting up a secondary server

on the control workstation. In this way, you could have multiple SP systems in the same authentication realm, and give all SP nodes access to an authentication server across the switch.

Adding principals and changing passwords takes place in the primary database and is propagated to secondary databases through periodic updates only. Secondary databases are maintained by the **kpropd** daemon that runs only on secondary server workstations and receives the database content in encrypted form from the **kprop** program that runs on the primary server workstation. The SP authentication services include a script that you can schedule for execution in the root **crontab** file to keep your secondary authentication databases up-to-date.

Configuration Files: Use the following configuration files located in the **/etc** directory to set up your system's authentication realm:

- **/etc/inetd.conf**

This file contains information used by the internet routing daemon **inetd** to route incoming requests for service to one of a large number of dynamically started daemons, named in the file. When the SP authentication services are installed on your systems, the file is updated to route "kshell" service requests to the AIX Kerberos-authenticated remote command daemon (krshd). You should not modify this information, but could possibly have to resolve conflicts with locally installed services with conflicting names.

- **/etc/services**

This file maps the names of network services to the well-known ports that they use to receive requests from their clients. This file may already contain entries relating to authentication services, since the file shipped with AIX 4.1 contains entries used by DCE. In addition to the DCE entries, many other reserved port names were added in this version of AIX, including an entry for the Kerberos V5 port, 88. The service name for this port is given as "kerberos", which is also the name used by the standard MIT Kerberos V4 service. The port number usually assigned to the Kerberos V4 service is 750. In order to be consistent with and interoperate with AIX 3.2.5 systems running PSSP 1.2 with authentication services based on Kerberos V4, it was necessary to use the name "kerberos4". You do not have to create an entry in the file for "kerberos4", because the default port of 750/udp will be used if no "kerberos4" entry is found. If you are not using AFS Version 3.4 authentication servers, you should only have to modify **/etc/services** if you are using some other service that uses one or more of the traditionally used (but not formally reserved) Kerberos V4 ports. They are:

Service name: kerberos4 Port: 750/udp
Service name: kerberos_admin Port: 751/tcp
Service name: krb_prop Port: 754/tcp

You will also have to modify this file if you are using AFS Version 3.4 authentication servers. The **kaserver** in AFS Version 3.4 for AIX 4.1 accepts Kerberos V4 protocol requests using the well-defined udp port assigned to the "kerberos" service assigned to port 88 in the file distributed with the base operating system. MIT Kerberos V4, on which PSSP authentication services are based, uses a default port number of 750. PSSP authentication commands use the service name "kerberos4" to avoid this conflict with the Kerberos V5 service name, which is also used by DCE. For PSSP authentication commands to communicate with an AFS 3.4 **kaserver**, you must do either of the following:

1. Stop the **kaserver**, redefine the udp port number for the “kerberos” service to 750 on the server system, then restart the **kaserver**.
2. Add a statement to **/etc/services** that defines the udp port for the “kerberos4” service as 88 on the SP control workstation and on any other independent workstation that will be a client system for PSSP authenticated services.

- **/etc/krb.conf**

This file contains the name of the local realm for your SP system and identifies the host names of all of the authentication servers for all Kerberos realms known to the local realm. For more information on the content of the file, refer to **krb.conf** in the *PSSP: Command and Technical Reference*.

If you configure your realm to use AFS authentication servers, this file is built for you automatically from the corresponding AFS configuration file **CellServDB**. When you use PSSP authentication servers, you create this file or one is generated by default by the **setup_authent** script that must run on each workstation after you install the PSSP files. You must provide the file if the workstation is a client workstation or a secondary authentication server. If it is the primary authentication server, you need to provide the file only if the system's host name contains no domain portion, following conventional network naming rules.

By convention, Kerberos uses the domain portion of a network interface name, converted to upper case, as the default realm to which the host belongs. If you want to define your own local realm name that does not follow convention, or if the host name of your primary server host has no domain portion, you must supply a file with your choice of local realm name in the first line. When using AFS authentication, the local AFS cell name, contained in the **ThisCell** file, converted to uppercase, becomes the local Kerberos realm name.

- **/etc/krb.realms**

This file maps network interface (host) names to realms. When Kerberos needs to determine to which realm a network interface is assigned, it looks in this file. If there is no entry for the host name or for the domain part of the host name, the default is a realm name equal to the domain portion of the host name, converted to upper case. Any network interface names on systems using SP authenticated services which have a domain portion which is not the same as the local realm name, except for case, must have an entry in this file.

Entries for all network interfaces on the SP nodes and on the control workstation that require this mapping in **/etc/krb.realms** are added automatically by either **setup_authent** initially or by the **setup_server** script. The file is kept identical on the control workstation and the nodes by the node installation customization process. There may be cases involving other workstations, however, where you will have to add entries to the file on one or more systems.

When you set up a secondary authentication server or set up a system as an authentication client system, you are instructed to copy the configuration files from the primary server. If the new system requires mapping in **krb.realms**, the **setup_authent** script will create the entry in the local copy of the file. You will have to manually add the entry for the workstation you just configured into the control workstation's file and into the files on any other client and server workstations already installed.

Setting Up a User Account to Obtain Kerberos Credentials at AIX Login: The system administrator can make using the authenticated services easier for nonroot users by setting up their user accounts so that Kerberos credentials are obtained whenever they login to the system. To do this, you must:

1. Add a stanza to the `/etc/security/login.cfg` file that names a program that will perform the required **Kinit** processing on the user's behalf. The name of the method (shown in the following example as **KERBEROS**) can be anything you want. The rest of the stanza should be entered exactly as shown:

```
KERBEROS:
```

```
    program = "/usr/bin/ksh /usr/lpp/ssp/kerberos/etc/Kinit"
```

2. Use SMIT to update either the **auth1** or **auth2** attribute in the user account for each user you want to use this capability. Add this method after the default **SYSTEM** method in the **auth1** attribute, if you want to require the user to obtain Kerberos credentials in order to successfully login to the system. Specify this method separately in the **auth2** attribute if you want to allow (but not require) the user to obtain Kerberos credentials at AIX login. The *Kerberos-principal* must be the user's AIX login name, optionally followed by a period and an instance (for example, **foo** or **foo.admin**). Note the placement of the comma (,) and the semicolon (;) in this rather unusual syntax.

The following is an example of an **auth1** attribute for AIX user **foo**. When **foo** logs in, the **k4init foo** command will be executed after the AIX password for principal **foo**. If **k4init** fails, **foo** is not logged into the system.

```
auth1 = "SYSTEM,KERBEROS;foo"
```

The following is an example of an **auth2** attribute for AIX user **bar**. When **bar** logs in, the **k4init bar.admin** command will be executed after the AIX password is validated. The user is prompted for the Kerberos password for **bar.admin**. In this case, if **k4init** fails, **bar** remains logged into the AIX system, but has no Kerberos credentials.

```
auth2 = "KERBEROS;bar.admin"
```

Setting Up the Primary Authentication Server: The first system that you set up, when you install your PSSP software is your primary authentication server. The primary authentication server may be your control workstation or it could be some other IBM RS/6000 workstation. You may want to have your servers entirely off of the SP system itself to provide greater physical security for them. You may want to allow logins to the control workstation that are not appropriate for the authentication servers.

The **setup_authent** script is used to configure your authentication services. If you have installed the **ssp.authent** fileset on the workstation prior to running the script, it assumes that the system will be a PSSP authentication server. If you provide your own **krb.conf** file, the script will look for an entry for the local host name. If it cannot find one, it allows you to configure the system without any server (as an authentication client system).

The **setup_authent** script creates the service principals for local service instances and creates the server key file for them. It updates the **krb.realms** file as needed, and creates the root user's **.klogin** file to authorize the initial administrator principal to use remote commands.

The daemons which provide the authentication server (the KDC), and the database administration server, are started by adding entries to **/etc/inittab**.

Refer to the following procedure to set up the primary authentication server:

1. Install **ssp.clients** and **ssp.authent**.
2. Create a **/etc/krb.conf** file if you want or if your host names do not follow Kerberos convention.
3. Execute the **/usr/lpp/ssp/bin/setup_authent** program and follow the instructions in the prompts.

Setting Up Secondary Authentication Servers: Refer to the following procedure to set up a secondary authentication server:

1. Install **ssp.clients** and **ssp.authent**.
2. Copy the **/etc/krb.conf** file from the primary authentication server to this secondary server system.
3. Add a line to the **/etc/krb.conf** file listing this system as a secondary server for the local realm.
4. Copy the **/etc/krb.realms** file from the primary server to the secondary server system.
5. Execute the **/usr/lpp/ssp/bin/setup_authent** program and follow the instructions when prompted.
6. If **setup_authent** created an entry for the local **/etc/krb.realms** file, copy the file to the other systems.
7. Add an entry for the new secondary server to the **/etc/krb.conf** file on the systems on which you had previously initialized authentication.
8. On the primary server, if this is the first secondary server, you should create a root **cron** entry that invokes the script **/usr/kerberos/etc/push-kprop** to periodically propagate database changes.

These steps are the same as those you would have followed to set up your SP control workstation as a secondary authentication server. For a more detailed description of how to do this, see *PSSP: Installation and Migration Guide*.

Setting Up Authentication Client Systems: Refer to the following procedure to set up an authentication client system:

1. Install **ssp.clients**.
2. Copy the **/etc/krb.conf** file from the primary authentication server to this system.
3. Copy the **/etc/krb.realms** file from the primary server to this system.
Note: If the new workstation is outside the realm of the primary server, the new workstation needs to be added to the primary server's **/etc/krb.realms** file first, before copying the **/etc/krb.realms** over to the new workstation. Otherwise, the next step, which runs the **setup_authent** program, will fail on the new client workstation.
4. Execute the **/usr/lpp/ssp/bin/setup_authent** program and follow the instructions when prompted.

5. If **setup_authent** created an entry for the local **/etc/krb.realms** file, copy the file to the other systems.

These steps are the same as those you would have followed to set up your SP control workstation as an authentication client system. For a more detailed description of how to do this, see *PSSP: Installation and Migration Guide*.

Directory Pathnames for SP Authentication Services: The following are some of the directory paths that are used with authentication services.

/usr/lpp/ssp/bin

Contains **setup_authent**

/usr/lpp/ssp/kerberos/bin

Contains commands for users of authentication services and authentication database administrators.

/usr/kerberos/bin

Contains a symbolic link for **/usr/lpp/ssp/kerberos/bin**

/usr/lpp/ssp/kerberos/etc

Contains authentication daemons and commands used by root to maintain a local SP authentication database.

/usr/kerberos/etc

Contains a symbolic link for **/usr/lpp/ssp/kerberos/etc**

/var/kerberos/database

Contains the SP authentication database on the server

/var/adm/SPlogs/kerberos

Contains error logs for the authentication servers

/etc

Contains the files **krb.conf**, **krb.realms**, and **krb-srvtab**

/usr/lpp/ssp/rcmd/bin

Contains an rsh and rcp link to the AIX versions of the remote commands. On the SP, these versions support Kerberos V4 through an SP-supplied remote command library.

/usr/vice/etc

Contains AFS cell information and utilities

/usr/afsws/etc

Contains AFS executables

Installing and Configuring DCE (Kerberos V5)

If you choose to install and configure DCE, you must set up the DCE cell prior to configuring the SP for using DCE, install DCE on the control workstation, and install and configure DCE on the desired nodes. IBM recommends that DCE servers not be on any workstation accessible by users. The SMIT interfaces and commands are documented in the DCE documentation and are not duplicated here. For more information, see "Suggested Reading" on page 132.

When configuring authentication for the SP, you need to:

1. Create .k5login files for remote command use
2. Enable Kerberos V5 authentication

Kerberos V5 is enabled as an authentication method (as opposed to enabling DCE) because the actual method used for authentication is the standard Kerberos V5 protocol. If DCE is installed and configured on the system, then Kerberos V5

authentication will be provided through the DCE Kerberos V5 compatibility mechanism.

Tips for Installing DCE on the SP

DCE will use all configured network interfaces available for any DCE runtime traffic. There may be circumstances where certain network interfaces or addresses should not be used. DCE provides a mechanism to exclude these interfaces or adapters. Excluding these interfaces does not preclude the use of these for remote command traffic.

DCE accomplishes this through the use of environment variables. These are: **RPC_UNSUPPORTED_NETADDRS** and **RPC_UNSUPPORTED_NETIFS**. The two variables accomplish the same task, so only use one of these variables. The recommended value to use is **RPC_UNSUPPORTED_NETIFS**.

Within the SP, there are specific adapters or interfaces, like the switch (css<#>) adapters, which do not communicate between the control workstation and the nodes. These adapters are prime candidates for exclusion from DCE traffic.

For example, to exclude the switch adapters, css0 and css1, do one of the following:

- Edit /etc/environment file on all nodes and add **RPC_UNSUPPORTED_NETIFS=css0:css1**.
- On the command line, enter **export RPC_UNSUPPORTED_NETIFS=css0:css1**

Start DCE within the same session the previous command was entered. If there are adapters on the control workstation, through which no DCE communication is expected, exclude these adapters as well using the same method described previously.

Installing and Configuring Standard AIX

Standard AIX authentication does not require any installation and configuration support beyond the support already provided for installation and configuration of the base AIX operating system.

When configuring authentication for the SP, you need to:

1. Select Standard AIX for rsh authorization for root
2. Enable Standard AIX authentication

Enabling Authentication Methods

During the installation and configuration of your RS/6000 SP system, and also during certain re-configuration tasks, it is necessary to enable the set of authentication methods that the AIX authenticated remote commands will use. On the control workstation and on other independent workstations that already have AIX installed at release level 4.3.1 or later, you may already have performed this task using the AIX **chauthent** command.

When you configure the nodes on the RS/6000 SP, you must use the **chauthpar** command to select the active authentication methods for each system partition. If

you have not created multiple system partitions, then all nodes are in a single, default partition. It is necessary for all the nodes in a particular system partition to use the same authentication methods. Like the AIX **chauthent** command, **chauthpar** activates the methods specified and deactivates all others. The methods you select must include Kerberos V4 (k4), because some PSSP services require Kerberos V4, as in prior releases of PSSP.

As you set the methods for each partition, **chauthpar** will invoke the AIX **chauthent** command as necessary to ensure that all methods used by all system partitions are also activated on the control workstation. The **chauthpar** command also executes **chauthent** remotely on any nodes in an existing system partition that are already installed and running.

When you deactivate an authentication method in a system partition, **chauthpar** will not remove it from the active set on the control workstation, even if the method is not used in any other partition. This is because it cannot determine whether the deactivated method is still required for AIX authenticated remote command use between the control workstation and other independent workstations. It must assume that is the case. If, in fact, the deactivated authentication method is not in use in any other partition and will not be required for AIX remote command execution between the control workstation and other workstations, you can use the AIX **chauthent** command to deactivate it. You must be careful not to also deactivate Kerberos V4 or any other authentication method still in use in any system partition. Use the AIX **lsauthent** command first to display the full set of active methods; then use **chauthent**, specifying the same set, minus the one method you want to deactivate.

Use the **lsauthpar** command to query the set of active authentication methods for a system partition.

Administering Security for Kerberos V4

This section discusses how to administer security on your SP system. Some of the tasks discussed may be common to multiple environments but the specific details apply only to Kerberos V4. For more specific information regarding administering security on a DCE system, refer to the appropriate DCE documentation. See “Suggested Reading” on page 132 for additional information.

Protecting Your Authentication Database

An important consideration when you install and configure your SP system is the security of the workstations on which you will be running authentication servers. Because it is the repository of all principals' secret keys, you need to consider steps to protect it from break-in or compromise of the master password. The following are steps you should consider:

- Authentication server systems should be located in physically secure areas, with access limited to administrators authorized to perform tasks related to their maintenance.
- Ordinary users, including SP system administrators who are not authorized as security administrators, should not have user IDs on these systems. Remote access to them via **telnet**, **rlogin**, and **ftp** should be disabled.
- AIX system auditing should be enabled on these systems to record security events.

- An appropriate password selection and aging policy should be implemented for the master password, and it should be changed regularly.
- Because compromise of the master password would expose all private keys stored in the database, a recovery plan should be available that would provide for changing all passwords, replacement of all server key files, and destruction of all outstanding tickets.

The root User and Authentication Administration Roles

When you set up your authentication realm and install and configure your SP system, the roles of system installer and administrator and security administrator become intertwined. This is necessitated by the fact that the PSSP installation and customization scripts are designed to automatically recognize the service principals that have to be created for the **hardmon** and **rcmd** services. The **setup_server** script creates service principals whenever it discovers a new network interface defined on the control workstation for which it has no Kerberos principal defined. At these times, the script must be running in an environment where the invoking user has authentication credentials as a database administrator. If the script needs to add a principal but finds that you have not got a ticket as an **admin** principal, it will fail. If this happens, simply get the required ticket and issue the command or invoke it from SMIT again.

Other installation and customization tasks invoke the **setup_server** script internally. Any of these tasks which involve adding or renaming of adapters or host names affect the Kerberos principals, and will also require a ticket as a database administrator. Examples of installation scripts used to perform these tasks are **spadaptrs**, **sphostnam**, **spbootins**, and **spethernt**.

Other administration tasks that simply use the remote command facilities of **dsh**, **rsh**, or **rcp** require normal user principal's credentials, since they do not involve changes to the authentication database.

Managing the Authentication Database

Backing up the Authentication Database

Keeping a current back-up of the database contents is an important task that you should perform. The database utility commands that are available to the root user on the primary authentication database system include functions you can use for this purpose. If you have secondary authentication servers and follow the steps outlined for setting them up, you will automatically create a backup of the database every time the **cron** file entry runs that propagates the database to the secondary servers.

The **push-kprop** script, invoked daily or at some other interval from the root user's **cron** file, creates a backup file in ASCII format called **slavesave**. This file is created in the database directory, where it is accessible only to the root user. If you want to create a backup file without using **push-kprop** (for example, if you do not have secondary servers and want to avoid the mail messages **push-kprop** sends to root), just add a **cron** entry to invoke the following command instead:

```
kdb_util dump /var/kerberos/database/slavesave
```


The result will be a backup file with the same pathname that can be used with the **kdb_util load** command to recover from a lost or corrupted database, or viewed or printed to examine the database contents.

Changing the Kerberos V4 Master Password

Changing the master password can enhance database security somewhat, but should not be substituted for adequate physical security. Changing the master password re-encrypts the database in the new key. It does not, however, change any of the Kerberos principals' passwords (keys) themselves. Therefore, it offers limited protection unless all (possibly compromised) passwords are changed also.

Changing the master key is one of the capabilities of the **kdb_util** command. See the command description in the reference manual. Note that only the root user can issue the various **kdb_*** commands on a system configured as a PSSP authentication server. An idiosyncrasy of these commands is that they prompt only once for the new password (unlike normal password-changing protocol).

After you change the password, be sure to issue the **kstash** command to store the new key in the **/.k** file. You will also have to kill and respawn the Kerberos daemons on the server (**kerberos** and **kadmind**). If you have secondary servers, you will have to force re-propagation of the database and respawning of the backup servers also.

Perform the following steps on the primary server:

1. Login to Kerberos as the admin principal used to set up authentication. For example:

```
k4init root.admin
```

2. Change the password (you will be prompted for old and new).

```
/usr/kerberos/etc/kdb_util new_master_key /var/kerberos/database/newdb.$$  
/usr/kerberos/etc/kdb_util load /var/kerberos/database/newdb.$$
```

3. Replace the **/.k** file (another prompt for the new password).

```
/usr/kerberos/etc/kstash
```

4. Kill and respawn the primary authentication server daemons.

```
kill 'ps -e | egrep 'kerberos|kadmind' | cut -d'-' -f1'
```

5. If secondary servers exist, propagate the new database, copy the new master key cache to the secondary servers, and kill and respawn the secondary server daemons.

```
if [[-s /var/kerberos/database/slavelist]]  
then  
  /usr/kerberos/etc/push-kprop  
  cat /var/kerberos/database/slavelist | while read slave  
  do  
    /usr/lpp/ssp/rcmd/bin/rcp /.k $slave:/  
    /usr/lpp/ssp/rcmd/bin/rsh $slave \  
      kill `ps -e \| grep kerberos \| cut -d'-' -f1`  
  done  
fi
```

Managing Authentication Principals

Listing Principals in the Authentication Database

With PSSP Authentication: Use the **lskp** command to list all or some principals in the local authentication database. Information displayed for each principal includes the name and instance, the maximum ticket lifetime, the key version number, and the expiration date (local time on the server).

The command allows you to select principals to list in several different ways:

- The entire authentication database
- Individual entries by name
- Entries with a specific name
- Entries with a specific instance
- Client (user) principals
- Service principals
- Principals pre-defined by Kerberos

The following example lists all principals who are Kerberos administrators plus all instances of the **rcmd** service principal, used by the SP remote commands and the Sysctl facility:

```
#lskp .admin rcmd.  
  
mary.admin tkt-life: 30d key-vers: 3 expires: 2000-12-31 23:59  
root.admin tkt-life: 30d key-vers: 5 expires: 2004-05-31 23:59  
rcmd.cwsta tkt-life: Unlimited key-vers: 1 expires: 2037-12-31 23:59  
rcmd.node1 tkt-life: Unlimited key-vers: 1 expires: 2037-12-31 23:59  
rcmd.node2 tkt-life: Unlimited key-vers: 1 expires: 2037-12-31 23:59  
rcmd.node3 tkt-life: Unlimited key-vers: 1 expires: 2037-12-31 23:59  
  
#
```

When logged into a system that is a Kerberos authentication server, the root user can run the **lskp** command directly. Additionally, any users who are Kerberos database administrators listed in the **/var/kerberos/database/admin_acl.get** file can invoke this command remotely through a Sysctl procedure of the same name. The administrator need not be logged in on the server host to run **lskp** through Sysctl, but must have a Kerberos ticket for that **admin** principal (*name.admin*).

In the following example, **control2** is the Kerberos server host name. The user has previously obtained a ticket as *name.admin*, for whom an entry exists in the **admin_acl.get** file:

```
$sysctl -h control2 lskp default  
  
default tkt-life: 30d key-vers: 1 expires: 2037-12-31 23:59  
  
$
```

For more information, refer to **lskp** in *PSSP: Command and Technical Reference*.

With AFS Authentication: To list the authentication database information, an AFS administrator uses the **kas list** command. It lists all entries in the database. For example (other entries have similar format):

```
$kas list -l -admin_user afsadm
```

```
Administrator's (afsadm) Password:
```

```
User data for afsadm (ADMIN)
```

```
key (0) cksum is 1479418564, last cpw: no date
entry never expires. Max ticket lifetime 720.00 hours.
last mod on Fri Jun 28 09:12:56 1996 by afsadm
```

```
User data for joseph
```

```
key (0) cksum is 1852746310, last cpw: Mon Mar 25 14:45:00 1996
entry never expires. Max ticket lifetime 720.00 hours.
last mod on Fri Jun 28 09:15:09 1996 by afsadm
```

```
...
```

```
$
```

Displaying Information about Principals

With PSSP Authentication: There are two ways for to examine information about principals. The **kadmin** command offers any administrator listed in the **admin_acl.get** file access from any network-connected system running SP authentication services. The root user on the primary authentication database system can also directly examine the database content by viewing the database backup file in **/var/kerberos/database/slavesave**.

Using **kadmin**, for example:

```
kadmin
Welcome to the Kerberos Administration Program, version 2
Type "help" if you need it.
admin: get ben
Admin password:
Info in Database for ben.:
    Max Life: 255    Exp Date: Fri Dec 31 23:59:00 1999

    Attribs: 00    key: 0 0
admin: quit
Cleaning up and exiting
```

Using the database backup file:

```
grep "^rcmd node3sw" /var/kerberos/database/slavesave
rcmd node3sw 255 1 2 0 a49bf286 d45c6560 200001010459 199503301502 root admin
```

The fields in each entry are:

- The principal's name
- The principal's instance
- The maximum ticket lifetime
- The key version of the master key used to encrypt the entry.
- The key version of the principal's secret key.
- The two halves of the encrypted private key.
- The expiration date (in Universal Coordinated Time) as YYYYMMDDHHMM

- The date the entry was created or modified
- The name and instance of the administrator who made the last update

With AFS Authentication: To display the database information for a user of AFS, an AFS administrator uses the **kas examine** command. For example:

```
kas ex -na ben -admin_user afsadm
Administrator's (afsadm) Password:
```

```
User data for ben
  key (0) cksum is 1004957312, last cpw: Mon May 22 10:12:50 1995
  password will never expire.
  An unlimited number of unsuccessful authentications is permitted.
  entry never expires. Max ticket lifetime 25.00 hours.
  last mod on Mon May 22 10:12:50 1995 by afsadm
```

Adding Principals and Assigning Initial Passwords

With PSSP Authentication: To add a principal and set the initial password in the authentication database, a database administrator with an **admin** principal that is listed in the **admin_acl.add** file, can use **kadmin** or **add_principal**.

To use **kadmin**:

1. Enter the **kadmin** command:

```
kadmin
```

A welcome message and explanation of how to ask for help are displayed.

2. Enter the **add_new_key** or **ank** subcommand:

```
ank name[.instance]
```

The only required argument for the subcommand is the principal's name.

3. At the prompt, enter your **admin** password
4. At the prompt, enter the principal's new password
5. At the prompt, reenter the principal's new password.

add_principal operates like **kadmin**, but allows an arbitrarily large number of principals to be added at one time. Whereas **kadmin** always prompts for your administrative password, **add_principal** will use tickets you have already obtained as *login-name.admin*. This makes **add_principal** suitable for use in scripts or background processes where a password cannot be provided interactively.

1. Create an ASCII file containing a list of principals and their initial passwords. Make sure it is readable only by you.
2. Enter the **add_principal** command.

```
add_principal filename
```

3. Immediately remove the file containing the initial passwords.

The following script fragment shows one way to add a user to the database without interactively editing a file, and keeping the initial password on disk for a minimal amount of time:

```

#( OMASK=$(umask); umask 077; INITIALPW=changeme$RANDOM; \
  print "johndoe $INITIALPW">/tmp/Addp$$; \
  /usr/kerberos/bin/add_principal /tmp/Addp$$; \
  /bin/rm /tmp/Addp$$; umask $OMASK; \
  print "johndoe's initial password is $INITIALPW"; )
johndoe's initial password is changeme28745
#

```

On the system which is the primary authentication server, the root user can use the **kdb_edit** command to add principals. The steps are:

1. Enter the command, which operates interactively, prompting for requests.

```
kdb_edit -n
```

2. Reply to the prompt for a principal name by entering the user name of the new principal and then the instance.

```
Principal name: joeuser
Instance:
```

3. Reply **y** to confirm that you want to add the principal.
4. Enter the initial password, when prompted.
5. Reenter the password, when prompted.
6. Accept the default expiration date by pressing Enter or type an alternate date in the format shown by the prompt.
7. Accept the default maximum ticket lifetime value or enter another.
8. Press enter when prompted for attributes.
9. The command will respond Edit OK and begin prompting for another principal to add. Repeat the same sequence or press enter to terminate the command.

For a complete example and more information, refer to **kdb_edit** in the *PSSP: Command and Technical Reference*.

You can use the **mkkp** command to add Kerberos principals to the authentication database instead of **kdb_edit**.

When logged into a system that is a Kerberos authentication server, the root user can run the **mkkp** command directly. Additionally, any users who are Kerberos database administrators listed in the **/var/kerberos/database/admin_acl.add** file can invoke this command remotely through a Sysctl procedure of the same name. The administrator need not be logged in on the server host to run **mkkp** through Sysctl, but must have a Kerberos ticket for that **admin** principal (*name.admin*).

Like **kdb_edit** (and unlike the **kadmin** command), **mkkp** can set attributes other than the password. Because it is not interactive, however, **mkkp** cannot set an initial password. To complete that step, so users can **k4init** as the new principals, an administrator should use the **cpw** sub-command of **kadmin**.

In the following example, these steps are used to add two new principals and then assign passwords to enable Kerberos logins. **control2** is the Kerberos server host name. The user has previously obtained a ticket as *name.admin*, for whom an entry exists in the **admin_acl.add** file:

```
$sysctl -h control2 mkkp -e 1998-06-30 jack jill
```

```
$kadmin -m
```

```
Welcome to the Kerberos Administration Program, version 2
```

```
Type "help" if you need it.
```

```
admin: cpw jack
```

```
Admin password:
```

```
New password for jack:
```

```
Verifying, please re-enter New password for jack:
```

```
Password changed for jack.
```

```
admin: cpw jill
```

```
New password for jill:
```

```
Verifying, please re-enter New password for jill:
```

```
Password changed for jill.
```

```
admin: quit
```

```
Cleaning up and exiting.
```

```
$
```

With AFS Authentication: To add a principal and set a password in an AFS authentication database:

1. Enter the **kas** command:

```
kas create -name user [-cell cellname]
```
2. At the prompt, enter your AFS password
3. At the prompt, enter the principal's new password
4. At the prompt, reenter the principal's new password to verify

Adding an Authentication Administrator

With PSSP Authentication: To perform the tasks involved in administering SP authentication services, the system administrator must be authorized to do so. The administrator must be a user with an **admin** instance whose name appears on one of the authentication administration access control lists (ACLs).

For example, to perform authentication administration, the administrator **gail** must have a principal **gail.admin** in the authentication database. In addition, **gail.admin** must be in the access control list file for the task she wants to perform. There are three ACL files:

/var/kerberos/database/admin_acl.add

The list of principals authorized to add entries to the authentication database

/var/kerberos/database/admin_acl.get

The list of principals authorized to retrieve entries from the authentication database

/var/kerberos/database/admin_acl.mod

The list of principals authorized to modify entries in the authentication database

Adding an authentication database administrator requires two steps:

1. Adding the principal with the **admin** instance, as described previously.
2. Adding the principal identifier to one or more of the ACL files, which must be done by the root user on the primary database system.

With AFS Authentication: AFS administration does not use an **admin** instance or ACL files. You, an AFS administrator, can authorize another AFS user to perform administrative tasks by assigning it the **admin** attribute:

1. Follow the steps described above to add the user, if not already defined.
2. Enter the **kas** command:


```
kas setfields -name user -flags admin [-cell cell]
```
3. At the prompt, enter your AFS password

Changing a Principal's Password and other Attributes

With PSSP Authentication: To change a password for a principal in the authentication database, a PSSP authentication database administrator can use either the **kpasswd** command or the **kadmin** program's **change_password** subcommand. You can issue these commands from any system running SP authentication services and do not require a prior **k4init**.

To use the **kpasswd** command:

1. Enter the **kpasswd** command with the name of the principal whose password is being changed:


```
kpasswd -n name
```
2. At the prompt, enter the old password
3. At the prompt, enter the new password
4. At the prompt, reenter the new password

To use the **kadmin** program:

1. Enter the **kadmin** command:


```
kadmin
```

A welcome message and explanation of how to ask for help are displayed.
2. Enter the **change_password** or **cpw** subcommand with the name of the principal whose password is being changed:


```
cpw name
```

The only required argument for the subcommand is the principal's name.
3. At the prompt, enter your **admin** password
4. At the prompt, enter the principal's new password
5. At the prompt, reenter the principal's new password

To change your own **admin** instance password, you can use either the **kpasswd** command or the **kadmin** program's **change_admin_password** subcommand.

To use the **kpasswd** command:

1. Enter the **kpasswd** command with your **admin** instance name:


```
kpasswd -n name -i admin
```

where *name* is usually replaced with **root**.

2. At the prompt, enter your old **admin** password
3. At the prompt, enter your new **admin** password
4. At the prompt, reenter your new **admin** password

To use the **kadmin** program:

1. Enter the **kadmin** command:

```
kadmin
```

A welcome message and explanation of how to ask for help are displayed.

2. Enter the **change_admin_password** or **cap** subcommand:

```
cap
```

3. At the prompt, enter your old **admin** password
4. At the prompt, enter your new **admin** password
5. At the prompt, reenter your new **admin** password

In addition to changing the password, you may want to change either the expiration date of the principal or its maximum ticket lifetime, though these are not so likely to be necessary. To do so, the root user on the primary authentication database system must use the **kdb_edit** command, just as when adding new principals locally. Instead of not finding the specified principal, the command finds it already exists, and prompts for changes to all its attributes, starting with the password, followed by the expiration date and maximum ticket lifetime.

The following example shows the use of the **kdb_edit** command to change a user principal's maximum ticket lifetime from the default value (30 days) to approximately 7 days:

```
#!/usr/kerberos/etc/kdb_edit -n
Opening database...
Previous or default values are in [brackets] ,
enter return to leave the same, or new value.

Principal name: johndoe
Instance:

Principal: johndoe, Instance: , kdc_key_ver: 1
Change password [n] ?
Expiration date (enter yyyy-mm-dd) [ 1996-09-01 ] ?
Max ticket lifetime [ 255 ] ? 171
Attributes [ 0 ] ?
Edit O.K.
Principal name:
#
```

Use the **chkp** command to change the maximum ticket lifetime and expiration date for Kerberos principals in the authentication database.

When logged into a system that is a Kerberos authentication server, the root user can run the **chkp** command directly. Additionally, any users who are Kerberos database administrators listed in the **/var/kerberos/database/admin_acl.mod** file

can invoke this command remotely through a Sysctl procedure of the same name. The administrator need not be logged in on the server host to run **chkp** through Sysctl, but must have a Kerberos ticket for that **admin** principal (*name.admin*).

In the following example, **chkp** is used by root to set the expiration date to the end of the year 2000 and the lifetime to approximately two weeks for several existing principals:

```
#chkp -e 2000-12-31 -l 181 joeuser joe.admin harry.admin user12
#
```

In the following example, the maximum ticket lifetime for new users (those added subsequently) is set to approximately a week. **control2** is the Kerberos server host name. The user has previously obtained a ticket as *name.admin*, for whom an entry exists in the **admin_acl.mod** file:

```
$sysctl -h control2

Sysctl (Version 1.1) on control2.abc.com
sysctl> chkp -l 170 default
sysctl> lskp default
default tkt-life: 7d+08:50 key-vers: 1 expires: 2037-12-31 23:59
sysctl> quit

$
```

For more information, refer to **chkp** in *PSSP: Command and Technical Reference*.

With AFS Authentication: To change the attributes of AFS user principals, use the **kas setfields** command. For example, to change the maximum ticket lifetime to two days:

```
kas setfields -name ben -lifetime 48:00 -admin_user afsadm
administrator's (afsadm) Password:
```

Deleting Principals

With PSSP Authentication: Use the **rmkp** command to remove Kerberos principals from the authentication database.

When logged into a system that is a Kerberos authentication server, the root user can run the **rmkp** command directly. Additionally, any users who are Kerberos database administrators listed in the **/var/kerberos/database/admin_acl.add** file can invoke this command remotely through a Sysctl procedure of the same name. The administrator need not be logged in on the server host to run **rmkp** through Sysctl, but must have a Kerberos ticket for that **admin** principal (*name.admin*).

Normally, this command requires you to confirm your intention to remove each principal as it is processed. An option is provided (**-n**) to allow the command to suppress prompting and run non-interactively. Running **rmkp** through Sysctl requires non-interactive execution; and doing so forces that option to be applied. When running **rmkp** interactively, the root user can select multiple principals by name or instance using the operand notation *name.* or *.instance* as for the **lskp** command.

Removed entries are preserved in an ASCII text file, in the format produced by the **kdb_util dump** command. If a principal must be recovered after having been

removed, the line in that file containing its entry can be appended to a current database backup file (see “Backing up the Authentication Database” on page 154) to be restored using **kdb_util load**. The following example shows how **rmkp** could be used by root to remove obsolete entries after the host name on the control workstation has been changed from **cwksta** to **cwksta2**:

```
#rmkp -v .cwksta

Confirm removal of principal hardmon.cwksta? (y or n): y
hardmon.cwksta was removed
Confirm removal of principal rcmd.cwksta? (y or n): y
rcmd.cwksta was removed
Removed entries were saved in /var/kerberos/database/rmkp.save.3036

#
```

In the following example **rmkp** is used to remove two principals. **control2** is the Kerberos server host name. The user has previously obtained a ticket as *name.admin*, for whom an entry exists in the **admin_acl.add** file:

```
$sysctl -h control2 rmkp -v pauline frank

pauline was removed
frank was removed
Removed entries were saved in /var/kerberos/database/rmkp.save.18365

$
```

With AFS Authentication: An AFS administrator can delete principals directly by using the **kas delete** command. For example:

```
kas delete -name ben -admin_user afsadm
Administrator's (afsadm) Password:
```

Managing Service Keys

Management of principals centers primarily on changing passwords in accordance with local security policy. For user principals this task is straightforward and can be done using simple tools. For service principals, password maintenance requires more than periodic changes to the authentication database. It also requires creation and distribution of files containing encrypted password information to the application server hosts.

Creation of Service Key Files

During setup for a node or control workstation, the keys for service principals are stored in the authentication database (for use by the authentication server) and on the node corresponding to the instance in the file **/etc/krb-srvtab** (for use by the service itself). Therefore, every node in the SP system contains a file, **/etc/krb-srvtab**, that contains the keys for the services that are provided on that node. On the control workstation, the **hardmon** and **rcmd** service principals are in the file. On the nodes, the **rcmd** service principals are in the file.

The local server key files on the control workstation and on other workstations are created by the **setup_authent** command when authentication is first set up on the system.

With PSSP Authentication: The `setup_server` script creates server key files for the SP nodes. When the control workstation is a PSSP authentication server, this occurs for each node when the script is run on the node's boot server.

`setup_server` creates the files in the `ftftpboot` directory, readable only by their owner `nobody`. When network boot occurs during install or customization, the file is transferred to the node using `ftftp`. The node uses the file to obtain tickets and invokes authenticated `rsh` to execute a script on the boot server that deletes the file.

With AFS Authentication or No Server on the Control Workstation: The `setup_server` script creates server key files for the SP node, when the script is run on the control workstation for the first time. `setup_server` creates the files in the `ftftpboot` directory, readable only by their owner `root`. The file is copied to the node's boot server and ownership is changed to `nobody`, when the script runs on the boot server. When network boot occurs during install or customization, the file is transferred to the node using `ftftp`. The node uses the file to obtain tickets and invokes authenticated `rsh` to execute a script on the boot server that deletes the file (gives it back to root, if the boot server is the control workstation). In this case the copy on the control workstation is never deleted.

Examining Service Key Information

There are two programs that you can use to examine the content of a server key file on the local system, `ksrvutil` and `k4list`: To view the local system's server key file in `/etc/krb-srvtab`, enter `ksrvutil list`. The output will look like:

```
Version    Principal
  1      rcmd.cwktr@XYZ.ABC.COM
  1      hardmon.cwktr@XYZ.ABC.COM
  1      rcmd.cwkfddi@XYZ.ABC.COM
  1      hardmon.cwkfddi@XYZ.ABC.COM
  1      rcmd.cwken0@XYZ.ABC.COM
  1      hardmon.cwken0@XYZ.ABC.COM
```

The same file, viewed by entering `k4list -srvtab` appears as:

```
Server key file: /etc/krb-srvtab
Service          Instance          Realm            Key Version
-----
rcmd             cwktr            XYZ.ABC.COM     1
hardmon         cwktr            XYZ.ABC.COM     1
rcmd            cwkfddi          XYZ.ABC.COM     1
hardmon         cwkfddi          XYZ.ABC.COM     1
rcmd            cwken0           XYZ.ABC.COM     1
hardmon         cwken0           XYZ.ABC.COM     1
```

To examine the server key file in `/ftftpboot` directory for a node whose host name is `sp2node8`, you would enter `ksrvutil list -f /ftftpboot/sp2node8-new-srvtab` and the output might look like:

```
  1      rcmd.n08tr0@XYZ.ABC.COM
  1      rcmd.n08tr1@XYZ.ABC.COM
  1      rcmd.n08fddi@XYZ.ABC.COM
  1      rcmd.n08css0@XYZ.ABC.COM
  1      rcmd.n08en0@XYZ.ABC.COM
```

Changing Service Keys

Changing the service keys involves two responsibilities:

- Changing the service keys in accordance with local system policy
- Ensuring that the service keys contained in the authentication database and in the server key files on all workstations and SP nodes always match.

You decide how frequently service keys need to be changed on your system. You may decide to set the service keys at setup time and never change them or you may decide to change them every six months. You should change the service keys if you have reason to believe system security has been compromised.

Use the **ksrvutil** command to change service keys. This command changes all of the service keys in the local service key file. The keys for all of the services provided on a single node, or the control workstation, are changed in a single call to **ksrvutil**. The keys are set to randomly generated values. The method of running **ksrvutil** to change the service keys for a node varies according to the authentication configuration of the system:

Primary Authentication Server on the Control Workstation: The **ksrvutil** command should be run on each node where the keys need to be changed. To change all of the service keys for a node to randomly generated values, enter:

```
ksrvutil change
```

The keys are updated in the node's **/etc/krb-srvtab** file and in the authentication database on the control workstation. To issue the command, you must be root. Use the **dsh** command to issue the **ksrvutil** command on multiple nodes.

You can also use the **ksrvutil** command to change service keys on the control workstation, regardless of the location of the primary authentication database.

All Other Authentication Configurations: If the primary authentication database does not reside on the control workstation, do not use the **ksrvutil** command on the nodes of the SP system. If the primary authentication database for the local realm resides in some location other than the control workstation (even if a secondary authentication database is on the control workstation) or if the system is using AFS-based authentication services, the system maintains a copy of each node's **/etc/krb-srvtab** file on the control workstation. These files are in the directory **/tftpboot** with a filename of **nodename-new-srvtab**. Changes to the **srvtab** file should be made only to the copy on the control workstation in **/tftpboot**. The **ksrvutil** command is used to change the service keys.

To change all of the service keys for a node:

1. Ensure that no users are using authenticated services on the target node
2. On the control workstation, issue:

```
ksrvutil -f nodename-new-srvtab change
```

3. Copy the **nodename-new-srvtab** file from the control workstation to the **/etc/krb-srvtab** file on the target node. To do this, you can use **ftp** or, as an alternative, you can customize the node using the **spbootins** command, followed by a **netboot** on the node.

To change the service keys for a node, you must have read/write permission to the **/ftpboot/nodename-new-srvtab** file and have write access to the **/etc/krb-srvtab** file on the node.

When you change the service keys for a node, or for the control workstation, any service tickets that have been previously issued to users for those service instances are no longer valid. An attempt to use such a ticket to authenticate the user's identity to the server will fail. A message will be issued that the server could not decode the ticket authenticator. In such a case, the user must issue a **k4init** command to obtain a new ticket-granting ticket and then reissue the service request.

Refer to **ksrvutil** in the *PSSP: Command and Technical Reference* for more information. Also, refer to the *PSSP: Diagnosis Guide* for information on recreating server key files.

Distributing Service Key Files to Client Systems

Normal distribution of server key files to nodes occurs as described using **ftp**. This entails some risk, but the risk is reduced by not making the files world-readable, as is usually the case for files transferred in that way. The most secure method is to log into the destination system, use **k4init** to get a ticket, and then invoke AIX **rcp** to pull the file from the boot server. That program cannot be used to push the file in the other direction because the remote command daemon on the target system needs a valid server key file to perform authentication.

Another means of copying the file that offers some security is the use of the standard AIX authentication for the **rcp** command, requiring the creation, temporarily, of a root **.rhosts** file. If the rcp cannot authenticate using Kerberos V4, then it will try Standard AIX, if enabled.

Less secure yet, and probably unacceptably so, are **ftp**. Unless it is done over a secure (internal) network interface.

Stopping and Restarting PSSP Kerberos V4 Daemons

If a Kerberos V4 daemon fails, it records an error condition in its log file and stops responding to client requests. Under these circumstances, you would need to restart it, after correcting the problem which caused the failure. There can be other circumstances, such as application of certain maintenance, where you might also have to stop and then restart one or more of the daemons that provide Kerberos V4 services. Each host that is an authentication server has two Kerberos V4 daemons that are started when the system is booted. The primary authentication server runs the **kerberos** daemon to provide ticket-granting services and the **kadmind** daemon to provide authentication database administration. Each secondary authentication server host also runs **kerberos**, and also the **kproppd** daemon, which is the "catcher" for database updates propagated from the primary server, usually via a cron job.

Authentication Daemons (Primary or Secondary Servers)

To stop the Kerberos V4 authentication server, enter the command:

```
stopsrc -s kerberos
```

To restart the Kerberos V4 authentication server, enter the command:

```
startsrc -s kerberos
```

Administration Daemons (Primary Servers Only)

To stop the Kerberos V4 database administration server, enter the command:

```
stopsrc -s kadmind
```

To restart the Kerberos V4 database administration server, enter the command:

```
startsrc -s kadmind
```

Database Propagation Daemons (Secondary Servers Only)

To stop the Kerberos V4 database propagation server, enter the command:

```
stopsrc -s kpropd
```

To restart the Kerberos V4 database propagation server, enter the command:

```
startsrc -s kpropd
```

Managing Your Authentication Credentials When Using Kerberos

This section discusses how to obtain, display, and delete authentication credentials and how to change authentication passwords. To manage authentication credentials when using DCE, refer to the appropriate DCE documentation. See “Suggested Reading” on page 132 for additional information.

When your realm is set up to use PSSP authentication servers, the PATH variable for these users should include **/usr/kerberos/bin**. When AFS authentication servers are used, the PATH variable should include **/usr/afsws/bin:/usr/kerberos/bin**.

Naming conflict if using both Kerberos V4 and DCE

Note that both DCE and Kerberos V4 can be installed, configured, and used on the same SP system. However, there is a naming conflict with some DCE and Kerberos V4 commands; both subsystems use **kinit**, **klist**, and **kdestroy** command names. The Kerberos V4 commands reside in **/usr/lpp/ssp/kerberos/bin**; the DCE commands reside in a DCE library.

The conflict occurs in the use of the pathnames in **/usr/bin**. When DCE is installed, it will automatically create symbolic links in **/usr/bin** for the DCE **kinit**, **klist**, and **kdestroy** commands.

PSSP 3.1 creates the following symbolic links:

- **/usr/bin/k4init** points to Kerberos V4 **/usr/lpp/ssp/kerberos/bin/kinit**
- **/usr/bin/k4list** points to Kerberos V4 **/usr/lpp/ssp/kerberos/bin/klist**
- **/usr/bin/k4destroy** points to Kerberos V4 **/usr/lpp/ssp/kerberos/bin/kdestroy**

If you use the command names **kinit**, **klist**, or **kdestroy**, be careful when setting the command search paths to ensure that they are executing the correct commands (either the DCE or Kerberos V4 commands). As an alternative, you can use the **k4init**, **k4list**, and **k4destroy** names to invoke the Kerberos V4 specific commands.

Obtaining Authentication Credentials

The first step in using the SP authentication services is to login to the authentication system. Logging in to the SP authentication services is how you obtain a ticket-granting ticket. The ticket-granting-ticket permits the client programs you invoke, such as **sysctl**, to obtain service tickets they can use to certify your identity to the servers with which they communicate.

Logging in to SP authentication services is a separate step from logging in to your AIX system; the Kerberos principal name space is separate from the AIX user name space. The administrator may assign principal names and passwords that are the same as AIX login names and passwords.

With PSSP Authentication

Use the **k4init** command to login to the SP authentication services. You can either enter the command interactively as needed, or you could add the command to your login script (**.login** or **.profile** file). If you use Kerberos-authenticated services infrequently, you might prefer the first option. Another alternative for frequent users of Kerberos authenticated management tools is to have the System Administrator set up Kerberos authentication as an authentication method to be invoked automatically at AIX login. For more information on this option, see “Setting Up a User Account to Obtain Kerberos Credentials at AIX Login” on page 149.

Use the **k4init** command to specify your Kerberos principal identifier as the only argument with no options.

```
k4init principal-identifier
```

For the principal identifier, specify your Kerberos user name, followed by a period and the instance, if you have more than one Kerberos identity and the instance you want to use is not null. Append an “at” sign (@) and a realm name if the realm you are logging into is not the local realm. Since the entire SP system is in a single realm, you will probably never need to specify a realm. Examples of this form of **k4init** are:

```
k4init joeuser  
k4init root.admin  
k4init ben@XYZ.ABC.COM
```

You might also choose to invoke the command without arguments. It responds by prompting for your Kerberos name, but only for the name portion of the fully qualified identifier.

```
k4init  
Kerberos name: ben
```

A common error is to omit the identifier on the command line and try to enter both name and instance in response to the prompt for the name. The result is:

```
k4init  
Kerberos name: root.admin  
2502-003 Bad Kerberos name format
```

Instead, you can enter the instance separately by specifying the **-i** flag on the command line and responding to the prompt:

```
Kerberos instance:
```

Similarly, the realm qualifier can only be entered as part of the command-line identifier unless you use the optional **-r** flag.

You can also use the **-l** option to request a ticket with a lifetime shorter than the maximum allowed for your principal by the database administrator. If you specify this flag, **k4init** prompts you:

```
Kerberos ticket lifetime (minutes):
```

After you enter the **k4init** command and respond to these optional prompts, the command contacts the authentication server to obtain a ticket. If no authentication server can be contacted, you will be notified of a time-out and the command will fail. Reasons for failure could be a missing or invalid configuration file or a networking problem.

When a ticket is obtained, **k4init** prompts you for your Kerberos password. If you enter it correctly, **k4init** saves the ticket-granting-ticket in the ticket file specified by your **KRBTKFILE** environment variable. If this variable is undefined, your ticket is stored in the file **/tmp/tktuid**, where *uid* specifies your AIX user identification number.

When the ticket-granting-ticket expires, subsequent attempts to be authenticated to use a Kerberos-supported service will fail. If the ticket-granting-ticket has expired, use **k4init** again to obtain a new ticket. There is no automatic renewal of tickets upon their expiration.

When you use **k4init**, it is valid for all processes running with your AIX user ID. It is not necessary to run this command for each window you open, as long as you have no background processes that require authentication. Note, however, that if you run authenticated applications as root in background processes or login under a shared user ID, you can potentially destroy the tickets of someone who logged in prior to you using the shared ID. In those cases you should use the **KRBTKFILE** environment variable to specify multiple ticket cache files. If you are uncertain of the current ticket cache file, use the **k4list** command to display the name and content of the file.

With AFS Authentication

In most situations, users of your SP system will be able to use the same **k4init** command described previously without regard to which type of authentication servers are being used. **k4init** can always be used whenever a user is logged into an SP node or the control workstation. When the user is logged into another IBM RS/6000 workstation which has been set up as an authenticated services client system, **k4init** can be used if the user has the **SP_NAME** environment variable set to the network name of the control workstation in the local realm.

Users who do not have the **SP_NAME** variable set, or who wish to use more familiar AFS commands, should use the **klog.krb** command. The following example shows the format of this command:

```
klog.krb -pr root.admin -li 48:00
```

When the **admin** instance is included in the principal identifier, **klog.krb** responds with a warning message:

```
Non-null instance (admin) may cause strange behavior.
```


You can ignore this message, since these principals are not used with the AFS file system.

As with **k4init**, you can specify a lifetime but in a more reasonable *hh:mm* format.

Displaying Your Credentials

With PSSP Authentication

Use the **k4list** command to display a list of currently held authentication tickets and their expiration time. The output would appear like the following, for example, if the user **ben** enters the **k4list** command after using the **k4init** command followed by any of the remote command services on four other hosts:

```
Ticket file: /tmp/tkt12763
Principal:  ben@XYZ.ABC.COM
```

Issued	Expires	Principal
Aug 12 16:26:11	Sep 11 16:26:11	krbtgt.XYZ.ABC.COM@XYZ.ABC.COM
Aug 12 16:45:15	Sep 11 16:45:15	rcmd.spcwkst@XYZ.ABC.COM
Aug 15 09:02:42	Sep 14 09:02:42	rcmd.snode3@XYZ.ABC.COM
Aug 15 09:02:43	Sep 14 09:02:43	rcmd.snode5@XYZ.ABC.COM
Aug 15 09:02:43	Sep 14 09:02:43	rcmd.snode6@XYZ.ABC.COM

With AFS Authentication

You may also use the **k4list** command to display tickets obtained from an AFS authentication server. The format of the output is the same as shown above, except that a service ticket for the “afs” service will appear, if you used the AFS **klog.krb** command to identify yourself. This additional service ticket is not used by SP authentication services. As an AFS user, you may prefer to use the **tokens.krb** command to display your tickets. Its output will appear like:

Tokens held by the Cache Manager:

```
User's (AFS ID 12763) tokens for afs@afs.abc.com [Expires May 31 03:24]
User ben's tokens for krbtgt.AFS.ABC.COM@afs.abc.com [Expires Jun 25 19:49]
User ben's tokens for rcmd.wksta2@afs.abc.com [Expires Jun 25 19:50]
--End of list--
```

Notice that **tokens.klog** shows the AFS cell name in place of the Kerberos realm name.

Deleting Your Credentials

With PSSP Authentication

Use the **k4destroy** command to destroy Kerberos tickets in the current ticket cache file. Depending on your local security policy, you may want to destroy your tickets whenever you end your AIX login session. **k4destroy** overwrites the ticket cache file with nulls before removing the file, so storage containing credentials cannot be reused. When you enter this command, the system will respond:

```
Tickets destroyed.
```

If the ticket cache file does not exist, you will get the response:

```
No tickets destroyed.
```

With AFS Authentication

You may also use the **k4destroy** command to delete tickets obtained from an AFS authentication server and stored in a ticket cache file. The responses are the same as shown previously. **k4destroy** will not remove the token held by the AFS Cache Manager, however, that was also obtained if you used **klog.krb** instead of **k4init** to obtain your initial ticket.

The AFS **unlog** command will remove your tokens held by the AFS Cache Manager, but will not remove tickets from the Kerberos ticket cache file.

Changing Your Authentication Password

With PSSP Authentication

To change your password, use the **kpasswd** command:

1. Enter the **kpasswd** command:
2. At the prompt, enter your old password
3. At the prompt, enter your new password
4. At the prompt, reenter your new password

For example:

```
kpasswd
Old password for ben:
New Password for ben:
Verifying, please re-enter New Password for ben:
Password changed.
```

With AFS Authentication

To change a password for a principal in AFS:

1. Enter the AFS **kpasswd** command. Be sure you have the directory for the AFS commands, **/usr/afsws/bin** ahead of the directory path for the Kerberos command, **/usr/kerberos/bin** in your search order, or specify the full pathname:
2. At the prompt, enter your old AFS password
3. At the prompt, enter your new password
4. At the prompt, reenter your new password to verify

For example:

```
kpasswd
Changing password for 'ben' in cell 'abc.com'
Old password:
New password (RETURN to abort):
Retype new password:
Password changed.
```

Using the AIX Remote Commands

PSSP 3.1 allows you to configure and enable multiple authentication methods to use within the AIX authenticated remote commands. The commands are:

- rsh** This command executes the specified command at the remote host or logs into the remote host.
- rcp** This command transfers files between a local and a remote host or between two remote hosts.
- telnet** This command connects the local host with a remote host, using the Telnet interface.
- ftp** This command transfers files between a local and a remote host.
- rlogin** This command logs into a specified remotes host and connects your local terminal to the remote host

All of these commands support both DCE and standard AIX as authentication methods. In addition, rsh and rcp also support authentication methods through Kerberos V4 on the SP only. All uses of rsh and rcp commands within PSSP use the AIX authenticated versions of rsh and rcp.

The rcmdtgt command is provided for applications requiring Kerberos V4 root credentials.

Authorization Files

The methods of authorizing users for access on a target system vary according to the authentication methods used. There is one type of authorization file for each authentication method in the SP system. The control workstation and the nodes which are configured for an authorization method maintain their own copy of that method's authorization file. For all methods, access is based on the contents of a file in the target user's home directory:

- Kerberos V4 – the .klogin file contains a list of authorized Kerberos V4 principal names. For PSSP 3.1, Kerberos V4 is required for all system partitions.
- Standard AIX – the .rhosts file contains a list of authorized source host names and user names
- DCE (Kerberos V5) – the .k5login file contains a list of authorized Kerberos V5 principal names

These authorization files for the root users are constructed such that:

- The root user on the control workstation can issue rsh as root to every node in the system
- The root users on a node can issue rsh as root to all nodes within all partitions which are configured with any authentication methods in common with that node.

If the authorization files already exist in the root user's home directory when you select the authorization method for the partition, the SP generated entries will be added to the existing file and the existing authorization files will remain intact.

The root user's authorization files are initially created when one or more authorization methods are chosen for a partition. Whenever new nodes are installed in a partition, nodes are moved from one partition to another partition, configured

for a different set of authorization methods, or nodes are removed from the system, the authorization files are automatically updated on:

- The node involved in the move
- The control workstation and all nodes in all partitions which share common authorization methods with the node's new partition (to add entries for the new node)
- All nodes in all partitions which share common authorization methods with the node's old partition (to remove the node's entries)

Whenever a node boots, the **/etc/rc.sp** command executes on the node and ensures that the node's authorization files are current and the files are updated, if necessary.

Using AFS to Provide Kerberos V4 Authentication

AFS is a distributed file system developed by Transarc Corporation. AFS includes an authentication implementation that is based on Kerberos V4. Therefore, AFS authentication servers can be used in place of SP authentication servers to provide credentials for SP principals.

Although AFS and SP authentication services can work together, there are differences in the way the authentication database is administered (for example, adding principals and changing passwords).

AFS uses a different set of servers, protocols, interfaces, and commands for Kerberos database administration.

Administering AFS Authentication Services

The primary tasks the system administrator performs in administering AFS authentication services are:

- Adding principals (users and services) and setting passwords in the authentication database
- Changing passwords for principals
- Managing the authentication keys for the set of services that use authentication (refer to "Managing Service Keys" on page 164 for more information)

The AFS system has its own administration commands and utilities for the administrator user. These commands and utilities provide for authentication, protection of files, file manipulation, and backup services. The following commands and utilities are relevant for administering authentication services. Refer to the AFS documentation for more information.

afsd	Daemon used to connect AFS clients to AFS server
cacheinfo	Specifies the directory to place the AFS directory on an AFS client
CellServDB	Specifies the name of AFS cell and location of AFS server
kas	Administrator interface to work with AFS authentication
klog.krb	User interface to obtain a Kerberos ticket in addition to AFS tokens
pts	Administrator interface to work with AFS protection services
token.krb	Allows the user to list the current tokens and Kerberos tickets

The following SP authentication services commands work in an AFS authentication system:

- **k4init** (The SP_NAME environment variable must be set in order to use this command on a client workstation, when using AFS authentication)
- **k4list**
- **k4destroy**
- **ksrvutil -afs change**
- **ksrvtgt**
- **rcmdtgt**
- **setup_authent**

The SP authentication services provide only the interfaces from the SP System to work in an AFS authentication system. The SP authentication services do not supply any of the AFS executables.

The AFS system provides multiple daemons that are activated on AFS server machines and support authentication tasks:

bosserv	Basic OverSeer server monitors the AFS server processes on the AFS server machine
kaserv	The AFS authentication server
ptserv	Used to create system entries in the authentication database
buserver	Maintains a backup authentication database on the authentication server

The authentication services work primarily with the **kaserv** and the **ptserv**. The **kaserv** allows the administrator to execute various commands to administer the authentication database. The **admin** user can enter **kas help** to view the commands available. The **admin** user needs to provide a password when using the **kas** command.

The **ptserv** protects and maintains system entries that are stored in the authentication database. The **admin** user can enter **pts help** to view the commands available. The **admin** user needs to provide a password when using the **pts** command. The admin user needs to create an AFS token (**klog.krb**) when working with the **pts** command.

Refer to AFS documentation for information on installing and administering the AFS system.

Chapter 13. Sysctl

Sysctl is an authenticated client/server system for running commands remotely and in parallel. It provides:

- **Least privilege capability**

Root authority can be dynamically provided to non-root users based on their authenticated identity, the task they are trying to perform, access control lists, and any other relevant criteria. The root password need not be given out to as many people, and thus is kept secure.

- **Distributed execution**

Sysctl applications can be executed on remote hosts with full authentication and authorization. It is essentially a secure, easy-to-program remote command execution mechanism for AIX commands, scripts and programs.

- **Parallel execution**

Sysctl applications can be efficiently executed in parallel on many hosts.

- **Programmability**

Sysctl applications can be coded as scripts for ease of implementation.

For example, consider a systems management script that contains commands that are only executable by root. If implemented as a Sysctl application, its security, usability, and performance are enhanced. Least privilege capability lets one of a select group of administrators run the script on each node in the SP without having access to the root password on any node in the system. With distributed execution, the administrator does not have to log in to each node to run the script. With parallel execution, the script can be run on all the nodes simultaneously with one command invocation. With programmability, the task can be automated or simplified with scripting.

Sysctl Components

Sysctl contains the following components:

- A server daemon (**sysctld**) that runs on all nodes and the control workstation.

This server daemon has root privileges. It executes commands as root on behalf of authorized clients.

- Commands built into the server.

These do the actual system administration work. Each command is paired with an authorization-checking script (callback) that has access to authenticated information about the client attempting to run the command. These commands include IBM-provided commands and typically also include installation-written ones.

- Configuration files that control aspects of the server operation as well as extend the command set available on a given node.
- Access control lists that list authorized users.
- A client program (**sysctl**) that offers command-line and interactive interfaces for communicating with **sysctld** servers.

Figure 9 on page 178 illustrates the way in which the Sysctl elements work together.

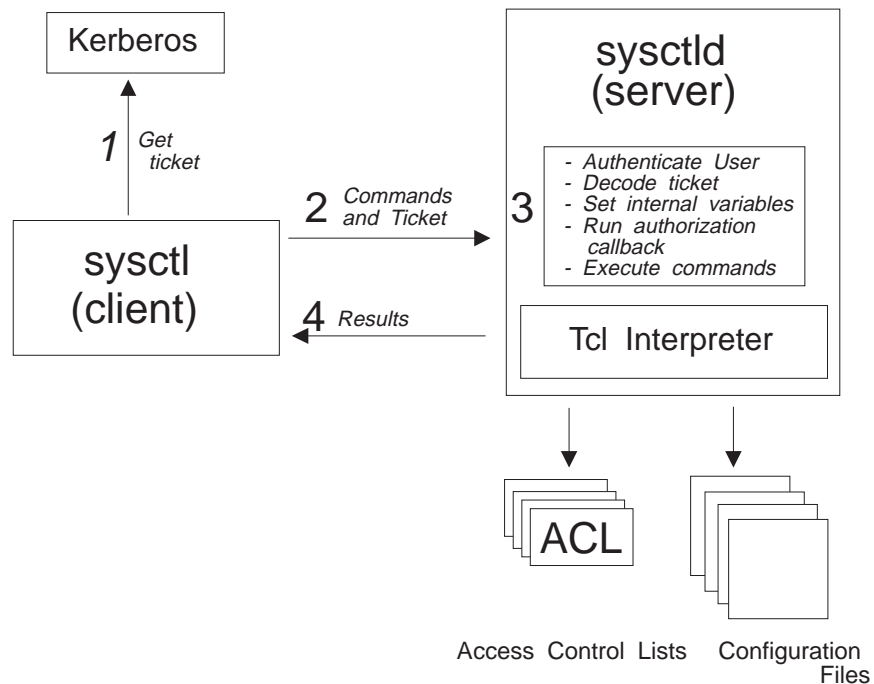


Figure 9. Sysctl Elements

The interpretation of a Sysctl command starts when the user issues a Sysctl client command. This can be embedded in a script or issued directly from the command line. It is of the following form:

```
sysctl target_nodes_specification
command args
```

Note that more than one node can be targeted. The command is the name of a Sysctl server command to be run as root on each of the targeted servers, if the client user is authorized.

1. The Sysctl client code obtains authentication information about the command issuer from SP Authentication Services.
2. The Sysctl client code sends the authentication information, along with the command string, to each of the specified Sysctl servers in parallel.
3. Each server then performs the following steps:
 - a. The authentication information is decoded and the authenticated information about the client is obtained.
 - b. Internal variables, such as the username and host of the client are set. These variables are available to the server routines that will check the authorization of the client (the authorization callbacks) as well as the Sysctl commands to be run.
 - c. The authorization is checked via an authorization callback. This can be a supplied callback, or an installation-written one. This callback may check an ACL. If the client passes the authentication check, the next step is done.
 - d. The server commands requested are executed as root, on behalf of the client.

4. **stdout** and **stderr** are sent back to the client. Output from each server is displayed with labeling indicating its source.

Relationship to the SP Authentication Services

Sysctl uses the SP Authentication Services. Client identity is authenticated across the network to the server using these services. Users of Sysctl typically need to identify themselves to the SP Authentication Services via the **kinit** command.

See Chapter 12, “Security Features of the SP System” on page 131 for more information on using Authentication Services.

Overview

Architecture

A Sysctl server provides remote access to a set of commands for authenticated and authorized clients. Since the Sysctl server is running with UID 0, these commands are executed with root privilege. These commands are embedded in an interpreter within the server. Sysctl uses the embeddable Tool Command Language (Tcl) as the foundation for its built-in command interpreter. The commands are of the following types:

- Base Tcl commands.

These are the commands making up the Tcl language. They include control commands (such as **if**), input/output commands (such as **puts**), arithmetic commands (such as **expr**) and commands allowing the execution of other commands, including external AIX commands and scripts (**exec**).

- Built-in Sysctl commands.

These are provided for you. They include commands to configure and manage Sysctl, as well as some systems management applications. See “Built-in Sysctl Commands” on page 197.

- Your applications.

The Tcl interpreter that Sysctl uses is built from base Tcl and Extended Tcl. Refer to the **ssp.public.README** file for more information on Tcl as supplied with the SP.

Using an embedded language, particularly Tcl, provides the following advantages:

- The command language interface is ideal for administrative tasks.
- The Sysctl command set is extended by writing Tcl scripts (instead of C code). These new Tcl scripts are embedded in the server's command set.
- Users can write simple Tcl wrappers for Perl and shell scripts and then add the scripts to the server command set.
- The Sysctl authentication mechanism gives the administrator complete control over the set of commands available to a given set of users. (see Chapter 12, “Security Features of the SP System” on page 131).

Security

The management tools used to administer a large site need to be secure to prevent outside intrusion and to prevent accidental disasters, such as inadvertently deleting critical data.

Sysctl security has the following components:

- Client authentication
- Authorization callbacks
- Access Control Lists

Client Authentication

The SP Authentication Service provides trusted third-party authentication of users. When a client issues a Sysctl command, a ticket identifying the client is sent along with the command. The server uses the ticket to verify the identity of the client.

Authorization Callbacks

The Sysctl authorization mechanism is based on callbacks. A callback is a Tcl routine paired with a Sysctl command that determines whether the client is authorized to run the command. The server invokes the authorization callback for a command just before executing the command. If the authorization callback returns a Tcl error, the command is not executed. When the callback returns a normal Tcl result, the command is executed.

With the callback mechanism, you can customize the server's authorization model to fit your needs. You can use the built-in authorization callbacks or write new ones, as needed.

Built-In Authorization Callbacks: These authorization callbacks are provided:

- | | |
|---------------|--|
| NONE | No special authorization is required to run a command. Even clients with unauthenticated identity (clients not identified to the SP Authentication Services) can run commands with this callback. |
| AUTH | Any client who has been successfully authenticated by the SP Authentication Service can run commands with this callback. |
| ACL | This callback checks an ACL file for an authenticated client. If the client is listed in the ACL, the callback succeeds. The pathname of an ACL file can be specified as an argument. If no pathname is specified, the client's principal must be listed in the global Sysctl ACL file, /etc/sysctl.acl . |
| SYSTEM | This level is used for commands that will only be run when authorization callbacks are bypassed (see below). This authorization callback always returns a Tcl error. A client cannot issue commands with this callback from the command line or the Sysctl prompt. |

Bypassing Authorization Callbacks: In some situations, the server bypasses the authorization callbacks. Authorization callbacks are bypassed while the server:

- Reads configuration files (that is, when starting up)
- Executes an authorization callback
- Executes the body of a procedure that a user is authorized to run

Access Control Lists (ACLs)

An ACL is a text file that can be used to grant authority to issue particular commands to particular users. Sysctl provides ACL-based authorization. See “Sysctl Files” for more information.

Sysctl Executables

Sysctl Daemon (sysctld)

The Sysctl server daemon, **sysctld**, processes all Sysctl client requests for the node on which it runs. There is a **sysctld** daemon running on each node of the SP system, as well as on the control workstation. **sysctld** runs as **root** and executes Tcl procedures for authenticated and authorized clients who may be local or remote users. See the **sysctld** man page for more information.

Client (sysctl)

The Sysctl client shell program, **sysctl**, offers a command line interface for communicating with **sysctld**. Together, **sysctl** and **sysctld** provide monitoring and execution abilities needed to remotely manage SP nodes. **sysctl** connects to a remote node's **sysctld** using TCP/IP, passes keywords and commands to the server, and writes any output returned to **stdout** and **stderr**. See the **sysctl** man page for more information.

Sysctl Files

Sysctl Configuration File

A configuration file is read in and interpreted each time the Sysctl server is started or restarted. When Sysctl is installed, each SP node will have a provided configuration file. The default pathname for this file is **/etc/sysctl.conf**. Alternate configuration files can be used. See the **sysctld** man page for more information.

The configuration file modifies the state of the server by executing one or more configuration commands. The configuration commands create read-only variables, register procedures, include other configuration files, and create classes of commands.

See “Installation and Configuration” on page 182 or the **sysctl.conf** man page for more information on configuring your system for Sysctl.

Access Control Lists (ACLs)

An ACL is a text file that lists authorized clients. (These files should be inaccessible to change by other than trusted administrators.)

There are two type of ACLs:

- The global Sysctl ACL file **/etc/sysctl.acl**

This identifies a server's **trusted** users. Trusted users are authorized to run any Sysctl or Tcl command, except for those with the SYSTEM authorization level. Trusted users have privileges equal to those of a root user. This ACL identifies those clients with the ACL authorization level.

The supplied **/etc/sysctl.acl** file contains commented lines that are samples. Copy the sample lines and insert your own principal identifiers.

- Installation-created ACLs

You can create ACLs for specific reasons. For example, you may designate a set of users that can issue a particular Sysctl command with an ACL for that command. The authorization callback for the command could check if the client was in the ACL.

Each line in an ACL contains either a Kerberos name (principal identifier) or the name of another ACL file. The principals in the included ACL file are added to those listed explicitly in the ACL file in determining the set of authorized clients. For example, an ACL file stored in one system can identify a core set of system administrators, followed by the name of another ACL file that is local to a node. Authorization for each SP node can be customized, while allowing the core administrators access to every machine.

The following is a sample ACL (note that the first line in an ACL file always begins with the magic cookie **#acl#**):

```
#acl#
_PRINCIPAL sarah.admin@TESTCELL.IBM.COM
_PRINCIPAL andrew@TESTCELL.IBM.COM
_ACL_FILE /usr/admin/acIs/global_sysctl.acl
```

This ACL identifies the Kerberos names of two trusted users, **sarah.admin** and **andrew**. Both of these users are trusted only if their tickets come from realm **TESTCELL.IBM.COM**. The ACL also names another file, **/usr/admin/acIs/global_sysctl.acl**, that contains additional ACL entries.

Note: You may want to use file collections to maintain configuration and ACL files. See Chapter 4, “Managing File Collections” on page 41.

Logfile

/var/adm/SPlogs/sysctl/sysctld.log is a logging file to which the Sysctl daemon (**sysctld**) writes. It logs the invocation of each Sysctl command. This is helpful for debugging. See the **sysctld** man page for information on specifying logfile pathnames.

Installation and Configuration

Installation

Sysctl is contained in the installation option file **ssp.sysctl**. This is installed when you install the SP system management software. See the *PSSP: Installation and Migration Guide* for more information.

Configuring the System

The basic procedure for configuring the system for Sysctl follows. Most of the steps are performed for you, either by the installation process or when you run **setup_authent** (both processes are described in *PSSP: Installation and Migration Guide*).

Note: Some of these steps require administrative access to the authentication database. See Chapter 12, “Security Features of the SP System” on page 131 for details.

Configuration Steps Done for You

When you install **ssp.sysctl** the following steps happen automatically:

- The **/etc/sysctl.conf** file, which contains configuration information for the **sysctld** daemon, is installed on each node.
- The following, which describes the IP port and protocol used for that port, is added to **/etc/services**:

```
sysctl 6680/tcp sysctld
```

- **/etc/inittab** is updated to automatically start the **sysctld** daemon. **init**, as root, starts **sysctld** as follows:

```
sysctld:2:once:/usr/bin/startsrc -s sysctld
```

Configuration Steps You Need to Perform

Here are the configuration steps that you need to perform:

- Add any trusted users to the global Sysctl ACL (**/etc/sysctl.acl**).

When the server is operational, you can manipulate Sysctl ACLs through the Sysctl built-in commands described in Table 9 on page 198.

- Define in the **/etc/sysctl.conf** file any new commands you have created and distribute it to each node either manually or via file collections.
- Optionally, create any new ACLs and distribute them either manually or via file collections to each node.
- If you have modified the configuration file, the **sysctld** daemons must be restarted. See “Developing a Sysctl Application” on page 190 for more information.

Debugging Configuration Problems

At this point, your system should be configured for Sysctl. After running **kinit**, you should be able to do Sysctl tasks, such as:

```
sysctl whoami
```

where, if your principal name is **andrew@kingston.ibm.com**, the system response looks like this:

```
andrew@kingston.ibm.com
```

If the authentication principals and server key files are set up properly, the server echoes your authenticated identity. If it returns with **unknown** or another error message, some portion of your configuration is incorrect. See the *PSSP: Diagnosis Guide*.

If a command does not work, run **sysctld** with debugging and security auditing turned on. First, you need to kill the local server and prevent its automatic respawning (such as with **/usr/bin/stopsrc -s sysctld**). Then you can start the daemon with:

```
sysctld -sd -l /var/adm/yourdebuglogfile
```

The **-s** option writes a security audit trace to the log file as the server executes each user request. The **-d** option writes debugging information to the server log file.

It is useful to run **tail -f** (to view the debugging information as it's written to the log file) on the log file.

Once you have issued **kinit** and issued any Sysctl commands, you can issue **klist** to display your current tickets. The **klist** command displays the principal name used with **kinit** and displays the list of service tickets currently active (with an expiration date) for the user.

After issuing **klist**, do the following:

1. Check the principal name. If this is incorrect, you need to issue **kinit** and enter the correct principal name.
2. Check the list of service tickets. The list of service tickets should contain one ticket for each **rcmd** service for which a Sysctl command was issued.
 - If the **rcmd** tickets do not appear, there is a problem with the principal definition set up in the authentication database.
 - If the **rcmd** service tickets appear, but issuing **sysctl whoami** returns unknown, it is likely that the SP Authentication Service is improperly configured. See the *PSSP: Diagnosis Guide* for more information.

Sysctl Configuration Commands

Configuration commands create read-only variables, register procedures, include other configuration files, and create classes of procedures. Configuration commands are interpreted when a server is started or restarted, and are contained in configuration files. The configuration commands are:

- **include**
- **create var**
- **create proc**
- **create class**
- **set**
- **load**
- Any other valid Sysctl command

include

The **include** command specifies additional configuration files for the server to read. This makes it easier to manage many external command definitions by letting them be split among a hierarchical set of files. The following example causes the Sysctl server to read the contents of the specified file at initialization time:

```
include $buildTop/samples/sysctl/pdfpfck.cmds
```

The syntax of **include** is:

```
include filename
```

create var

The **create var** command defines a read-only variable in the Sysctl server. Variables defined in the configuration file using the standard Tcl **set** command exist only while the configuration files are being read.

The syntax of **create var** is:

```
create var variable value [authcallback]
```

where:

variable A read-only variable to which you assign a value.

value The value of the variable

authcallback

An authorization callback for the command. This can be **NONE**, **AUTH**, **ACL** or **SYSTEM** (to assign it one of the built-in authorization levels), or the name of another previously-registered procedure.

The following example defines a variable (**STARTTIME**) that contains the time you started up the server:

```
create var STARTTIME [exec /bin/date] NONE
```

create proc

The **create proc** command defines external Sysctl procedures.

The syntax of **create proc** is:

```
create proc name args authcallback body
```

where:

name The name of the Sysctl procedure you are creating.

args A list of argument names to the procedure.

authcallback

An authorization specifier for the command. This can be **NONE**, **AUTH**, **ACL** or **SYSTEM** (to assign it one of the built-in authorization levels), or the name of a previously-registered procedure.

body A Tcl expression that forms the body of the new procedure.

See “Developing a Sysctl Application” on page 190 for an example of **create proc**.

create class

The **create class** command defines classes of commands within the server. Command classes provide a way to organize commands into logical groups for clarity.

The syntax of **create class** is:

```
create class name file [authcallback]
```

where

name The name that you assign to the command class.

This name constitutes a **tag** that is prefixed to all commands defined in the command file.

For example, if a class named **test** is created and the class file defines a procedure named **help**, the procedure is created in the interpreters as **test:help**.

file The pathname of the file containing the definition of the class objects.

authcallback

An authorization specifier for the command. This can be **NONE**, **AUTH**, **ACL** or **SYSTEM** (to assign it one of the built-in authorization levels), or the name of a previously-registered procedure.

set

The **set** statement sets the value of the server variables ACL, LOG or KEY, or sets values in the Tcl env() array. The default values for the Sysctl server's ACL file, log file and Kerberos key file can be overridden by assigning values to the ACL, LOG and KEY variables in the configuration file. For example, the following line overrides the default value for the log file name:

```
set LOG /var/sysctld.logfile
```

The values assigned to the ACL, LOG and KEY variables are overridden by the optional command line arguments **-a**, **-l** and **-k** on the **sysctld** command line. Environment variables (such as the default PATH) can also be set within the configuration file by assigning values to the env() array. For example:

```
set env(PATH) /usr/bin:/bin:/usr/etc:/etc
```

sets the PATH environment variable for the Sysctl server. The env() array is assigned an authorization callback of SYSTEM which prevents its modification from outside the Sysctl server by a request sent by a Sysctl client.

load

The **load** command dynamically loads the shared library at **lib_path** into memory. If the **init_proc** parameter is given, it is used as the library's initialization procedure. Otherwise the name of the initialization function is derived from the library name as follows:

library name	init function
=====	=====
libxxx.sl	xxx_Init()
libxxx.a	xxx_Init()

The Sysctl server exports an API that the library uses to define commands, variables, authorization callbacks and interpreter deletion callbacks. See the **load** help page for details.

Other Sysctl Commands

You can include other commands in the Sysctl configuration file. These commands are executed by the server when it is started (or restarted).

Authorization Tasks

Authorization Callbacks

Sysctl provides an extremely flexible authentication mechanism to allow you to restrict the use of registered Sysctl procedures to any easily defined subset of users. Authorization callbacks are the means by which this authentication is implemented. As previously discussed, Sysctl includes four supplied authorization callbacks which provide the following authorization levels:

- NONE
- AUTH
- ACL

- SYSTEM

Adding New Authorization Callbacks

As an administrator you can develop a procedure with an authorization level that does not fit one of the supplied authorization callbacks. To do this, write your own authorization callback. Here is an example of a custom authorization callback:

```
create proc sdate {} SYSTEM {exec /bin/date}

create proc RCRAUTH {cmdName} SYSTEM {
    global SCUSER
    global SCPRINCIPAL
    if {$SCUSER != "rcr"} {
        svclog "Denying \"${cmdName}\" access for user $SCPRINCIPAL"
        error "Authorization denied!"
    } else {
        svclog "Authorizing \"${cmdName}\" access for user $SCPRINCIPAL"
        return "Authorization OK!"
    }
}

create proc rcrdate {} {RCRAUTH rcrdate} {sdate}
```

This example contains the definition of three procedures:

- **sdate** is a procedure that the SYSTEM authorization callback is attached to. The SYSTEM callback prevents **sdate** from being directly executed from outside of the server (that is, a user cannot successfully issue **sdate** from the Sysctl client). The **sdate** procedure can, however, be successfully executed from within another procedure which has been previously registered with the Server (that is, included in the Sysctl configuration file), since authorization callbacks are bypassed during this execution.
- **RCRAUTH** is a procedure that functions as an authorization callback. It accepts a command name as its single argument (used for logging purposes only). Its function is to return a normal result if the basename of the user is **rcr** and to return an error result in all other cases.
- **rcrdate** is a procedure that invokes the **sdate** procedure. Before control is given to the **sdate** procedure, the RCRAUTH authorization callback is invoked. If the base name of the user is **rcr**, then the callback returns a normal result and the **sdate** command is allowed to be invoked. If the callback returns an error result, then the **rcrdate** procedure is terminated.

ACL Files in Authorization Callbacks

The supplied **ACL** authorization callback is used to check for the remote user's principal name in an ACL file. If this callback is used without any parameters then the ACL file named by the ACL variable (normally **/etc/sysctl.acl**) is checked. If a file name is supplied as a variable to the ACL callback, then that file is checked for the user's principal name.

This allows an authorization callback to reference an alternate ACL file in performing the authorization check:

```

create proc myproc {} {ACL /etc/myproc.acl} {
    ....
    ....
    ....
}

```

The previous example defines the procedure **myproc**, which takes no parameters, and has an authorization callback of **ACL /etc/myproc.acl**. This callback returns a normal result if the remote user's principal name appears in the file **/etc/myproc.acl** and an error return if it does not.

ACL files can be created and deleted and their contents can be manipulated and checked by using the Sysctl commands **aclcreate**, **acldestroy**, **acladd**, **acldelete**, **acllist**, **aclcheck** and **aclrecreate** (see the Sysctl help information for these commands).

Using Sysctl

Prerequisite Checklist

Before using Sysctl, the following is required:

- You have a principal in the local Kerberos realm and have run the **kinit** command to authenticate your identity to the system.
- The configuration is complete (see “Configuring the System” on page 182).
- The server daemon (**sysctld**) is running on all nodes.

Using Sysctl in Interactive Mode

Sysctl has an interactive mode. Note that only one node can be communicated with at a time in interactive mode.

To enter interactive mode, enter the **sysctl** client command without a command argument. For example, to use Sysctl in interactive mode on the local node, enter:

```
sysctl
```

If you issued the command from **diane.kgn.ibm.com**, you get the following prompt:

```
Sysctl (Version 1.1) on diane.kgn.ibm.com
sysctl>
```

At this point, you can enter Sysctl commands.

You can also specify some of the Sysctl flags when going into interactive mode. For example, to use Sysctl in interactive mode for node **redhook.kgn.ibm.com**, enter:

```
sysctl -h redhook.kgn.ibm.com
```

Sysctl issues this response:

```
Sysctl (Version 1.1) on redhook.kgn.ibm.com
sysctl>
```

In this example, if you enter **svcversion** and **pdf**, the interaction is as follows, where your actions are in **bold**:

```
sysctl> svcversion
```

```
1.1 sysctl> pdf
```

```
redhook.kgn.ibm.com: =====  
Filesystem      Size-KB  Used-KB  Free-KB  %Free   iUsed   iFree  %iFree  
/                8192     5164     3028     37%     815    1233   61%  
/var            16384     4176    12208     75%     343    3753   92%  
/usr           552960   467888   85072     16%    23203 116061  84%  
/tmp            49152    23860   25292     52%     235   12053  99%  
/home           65536    63556    1980      4%     5375  11009  68%  
sysctl> quit
```

To exit the interactive mode, enter one of the following:

```
exit  
quit  
<Ctrl-d>
```

Using Sysctl in Noninteractive Mode

Sending Commands to a Local Server

If no target nodes are specified on the Sysctl command, the command is sent to the server on the local node. For example, to send the **whoami** command to the local server, enter:

```
sysctl whoami
```

Directing Commands to Remote Hosts

To direct a command to remote host **sivle**, use **-h** and enter:

```
sysctl -h sivle sys:info
```

The system response is similar to:

```
sivle.kgn.ibm.com power AIX 4.3.2
```

To direct a command to remote hosts **sivle**, and **yelserp**, enter:

```
sysctl -h sivle -h yelserp sys:info
```

The system response is similar to:

```
>> sivle  
sivle.kgn.ibm.com power AIX 4.3.2  
<<  
>> yelserp  
yelserp.kgn.ibm.com power AIX 4.3.2  
<<
```

Sending Commands to a Collection of Nodes

Multiple nodes can be targeted with a single Sysctl command. The collection of nodes can be specified in a file. Alternatively, the Sysctl command will read node names from **stdin**. In the following example, the file **/tmp/node-file** contains node names **r05n13.hpssl.kgn.ibm.com**, **r05n15.hpssl.kgn.ibm.com**, and **r05n09.hpssl.kgn.ibm.com**.

To direct the **pdf** command to the **/tmp/node-file** collection, enter:

```
sysctl -c /tmp/node-file pdf
```

The response in this example is the following:

```
>> r05n13
r05n13.hpssl.kgn.ibm-----
Filesystem          Size-KB Used-KB Free-KB %Free  iUsed  iFree %iFree
/                   4096   3372   724   18%   792   232   23%
/var                8192   6320  1872   23%   297  1751   86%
/usr               335872 331632  4240    2% 17820 66148   79%
/tmp                8192    520  7672   94%    67  1981   97%
/home              4096    168  3928   96%    18  1006   99%
<<
>> r05n15
r05n15.hpssl.kgn.ibm-----
Filesystem          Size-KB Used-KB Free-KB %Free  iUsed  iFree %iFree
/                   4096   3364   732   18%   796   228   23%
/var                8192   3028  5164   64%   281  1767   87%
/usr               335872 331412  4460    2% 17825 66143   79%
/tmp                8192    464  7728   95%    57  1991   98%
/home              4096    168  3928   96%    18  1006   99%
<<
>> r05n09
r05n09.hpssl.kgn.ibm-----
Filesystem          Size-KB Used-KB Free-KB %Free  iUsed  iFree %iFree
/                   4096   3340   756   19%   790   234   23%
/var                8192   3336  4856   60%   285  1763   87%
/usr               331776 331012   764    1% 17742 66226   79%
/tmp                8192   1212  6980   86%    58  1990   98%
/home              4096    164  3932   96%    17  1007   99%
<<
```

Developing a Sysctl Application

Creating a Sysctl application consists of the following basic steps:

1. Examine what you are trying to do and divide it among client/server lines. The server side runs as **root**. The client side typically does not require root authority, but requires that the user has run **kinit**.
2. Define the client/server interface. This consists of Sysctl command names and their arguments, return values, and authorizations.
3. Code the server commands. This consists of Tcl code that performs your task, as well as callback authorizations if you do not use one of the supplied ones. Note that the Tcl code may be simple (such as a wrapper around an AIX command or script) or elaborate (a detailed Tcl program).

4. Code the client side. This typically gathers any necessary information and then issues the **sysctl** command to cause the servers to perform the desired action. This can be a GUI, Perl or shell script. Note that client side code may not always be necessary. In some cases, issuing the **sysctl** command directly from the command line is sufficient.
5. If the authorization scheme requires changes to existing ACL files or new ACL files, these changes must be made, and the changed files distributed.
6. Register the new server command with each node's **sysctld** daemon by restarting the daemons with updated configuration files.

Here is an example of how to write a Sysctl application that allows a user to execute a script including some commands requiring root execution privileges on all the nodes in an SP system in parallel. The user is not required to know any of the root passwords, but must be listed in an ACL specifically set up to authorize use of the script. A copy of this ACL resides on each node. The script containing the commands executable only by root is called **/etc/root_script**. Assume that **/etc/root_script** accepts a single argument, and that this script resides on each node.

The server side of the application does the root part of the job, which is to issue the **/etc/root_script** command with the correct arguments on behalf of authorized users. It first needs to determine if the client is authorized based on an authorization policy. As mentioned above, authorization is based on an access control list. There are no return values, but an error message is displayed when the user is not authorized to run the command. **stdout** and **stderr** are returned to the client.

The server command resides in **/etc/sysctl/apps/adminscript** and is coded as follows:

```
create proc adminscript {argument} AUTH {
    global SCPRINCIPAL
    if [ aclcheck -f /etc/sysctl.adminscript.acl $SCPRINCIPAL ] {
        exec /etc/root_script $argument
        return
    }
    puts "Not authorized to run adminscript"
}
```

The first line creates a new Sysctl proc called **adminscript**. One argument is accepted, which will be passed unchanged to **/etc/root_script**. This proc uses the provided AUTH authorization callback, which means that only authenticated clients can execute the command.

The second line issues the Tcl **global** command to provide the content of the SCPRINCIPAL variable to other Sysctl commands. (This is needed because SCPRINCIPAL is passed as an argument to **aclcheck** later.) The SCPRINCIPAL variable is the authenticated identity of the command issuer and is one of the variables set by the server and available to procedures and callbacks. (See "Authorization Variables" on page 193 for a list of all these variables.)

The third line is Tcl code that uses the built-in **aclcheck** command to see if the user is in the ACL (**/etc/sysctl.adminscript.acl**) set up for this command. If the user is authorized, the program issues the **/etc/root_script** command with the

supplied argument and then returns. If the user is not authorized, the program issues an error message. The program returns any output (information from **/etc/root_script** command or the error message) to the client user's terminal.

Note that in this example, further authorization checking is done in the body of the procedure after the authorization callback has completed. This is to demonstrate the use of the SCUSER variable provided by the server to procedures. Alternatively, this check could have been made part of the authorization callback itself.

The client side of the application needs to find out the necessary information to provide to the servers. It does not require **root** authority to be issued. The information in this case includes the single argument to be passed through to **/etc/root_script**. The client also needs to determine the hostnames of all the nodes in the SP System, so that the command can be sent to and run in parallel on all the nodes.

The client side is coded as follows in **/usr/bin/doadminscript**:

```
#!/usr/lpp/ssp/perl/bin/perl

# Run /etc/root_script on all the SP nodes in parallel by invoking
# the adminscript sysctl command on all the nodes. Invoke via:
# doadminscript arg_to_root_script

$#ARGV > 1 && die "too many arguments";

exec "/usr/lpp/ssp/bin/hostlist -av |
      /usr/bin/sysctl -c - adminscript $ARGV[0]";
```

The first line checks for extraneous arguments. The **exec** statement uses the SP **hostlist** command to write all the currently responding hostnames in the SP System to **stdout**. The single argument to **adminscript** is passed along to the Sysctl **adminscript** server command.

A new ACL file needs to be put on each node in the system. The file **/etc/sysctl.adminscript.acl** looks like this:

```
#acl#
# These are the users that can issue the adminscript command on this
# node:
_PRINCIPAL root@HPSSL.KGN.IBM.COM
_PRINCIPAL newbie@HPSSL.KGN.IBM.COM
_PRINCIPAL guru@HPSSL.KGN.IBM.COM
```

Note that each node can authorize different users to run the new command. In this case, assume that the ACL will be the same on all nodes. To distribute the ACL file to all the responding nodes, an authorized user enters the following (see the **pcp** man page for details):

```
pcp "-av" /etc/sysctl.adminscript.acl
```

In addition, the **/etc/sysctl.conf** file needs to be updated to include the new command. If the server code is to reside in **/etc/sysctl/apps** directory, the following lines can be added to **/etc/sysctl.conf**:

```
# Add the adminscript application
include /etc/sysctl/apps/adminscript
```

The updated **sysctl.conf** file and the actual server code need to be distributed:

```
pcp "-av" /etc/sysctl.conf
pcp "-av" /etc/sysctl/apps/adminscript
```

Now the servers need to be restarted to include the updated **sysctl.conf** file. Note that the following command requires ACL authorization:

```
hostlist -av | sysctl -c - svcrestart
```

Now, any of the users in **/etc/sysctl.adminscript.acl** (**root**, **newbie**, or **guru**) can run **/etc/root_script** with the argument **mikef** on all the nodes simultaneously by entering the following commands:

```
kinit (to get authentication information)
```

```
doadminscript mikef
```

Authorization Variables

These variables are set by the server based on information on the client provided by SP Authentication Services. Authorization callbacks and Tcl scripts have access to these variables. Table 6 lists the authorization variables.

<i>Table 6. Authorization Variables</i>	
Variable	Description
SCACL	Indicates if the client has been authenticated and is a trusted user. Value is 1 if the client has been authenticated and is listed on the server's ACL of trusted users, 0 otherwise.
SCHOST	The name of the host SP node from which the Sysctl request was issued.
SCINSTANCE	The instance of the user or NULL if unauthenticated.
SCLHOST	The host name of the local server.
SCLREALM	The local realm of the server.
SCMODE	The communication mode between the client and server. May be WAIT, NOWAIT, or SOCKET.
SCPRINCIPAL	The authenticated identity of the command issuer, in the form user.instance@realm. If the client has not been authenticated, the value is NULL.
SCREALM	The realm of the user or NULL if unauthenticated.
SCSCRIPT	The commands sent from the client to the server. If a script was passed, the contents of the script is shown.
SCUSER	The base name of the user. NULL if unauthenticated.

Sysctl Scripts

Replaying a Sysctl Script

A Sysctl script consists of a set of Sysctl commands that can be executed with a single Sysctl client invocation. Sending a Sysctl script to a server or set of servers is called *replaying* the script on those servers.

When a script is replayed, each Sysctl command is run sequentially by the servers. Authorization is done on a command by command basis as the script runs. The script stops execution when any command fails to be executed due to an authorization failure.

To run a Sysctl script, send it to the Sysctl server by specifying **-r** (replay) on the **sysctl** command:

```
sysctl -h node -r script-filename
```

The following is a simple script, **/examples/sysctl/script1**:

```
echo I am $SCLHOST
echo You are $SCPRINCIPAL
```

Here is the command that user **sarah** issues to run the script on SP node **r05n11**:

```
sysctl -h r05n11 -r /examples/sysctl/script1
```

sarah receives the following output:

```
I am r05n11.testcell.ibm.com
You are sarah@testcell.ibm.com
```

You can also have Sysctl read a script from **stdin** by typing a dash (-) after the **-r** flag, instead of a file name. For example, to run three commands for node **tappan.kgn.ibm.com**, enter the following sequence of commands:

```
sysctl -h tappan.kgn.ibm.com -r -
echo $SCLHOST
pdf
svcversion
<Ctrl-D>
```

You will not get any responses until you press **<Ctrl-D>**. In this example, the response is the following:

```
tappan.kgn.ibm.com
tappan.kgn.ibm.com: =====
Filesystem          Size-KB Used-KB Free-KB %Free  iUsed  iFree %iFree
/                   8192   5164   3028   37%    815   1233   61%
/var                16384   4184  12200   75%    343   3753   92%
/usr               552960 467888  85072   16%  23203 116061   84%
/tmp               49152   23868  25284   52%    239  12049   99%
/home              65536   63556   1980    4%   5375  11009   68%
1.1
```

In addition, Sysctl scripts can be written as self-contained programs. Do this by beginning the script with (flags other than **-r** can be specified this way as well):

```
#!/usr/lpp/ssp/bin/sysctl -r
```

Enter the name of the script to run it.

Consider the following example, **/etc/fscheck**. This script checks filesystem usage and reports on filesystems that have usage higher than that specified in an argument. The **-L** flag causes output to be formatted as a set of lines, each preceded by the hostname from which it came:

```
$ cat /etc/fscheck
#!/usr/lpp/ssp/bin/sysctl -Lr
#
# Check filesystem usage - high-water pct mark is passed as a parameter
#
foreach fs [listfs] {
    statfs $fs sbuf
    set pct [expr $sbuf(used).0/$sbuf(size).0*100]
    if {$pct > $argv(1)} {
        set pct [format "%.2f" $pct]
        echo "$fs ==> ${pct}% Used, $sbuf(free) KB Free"
    }
}
}
$ cat /tmp/hostlist
harlem
badger
$ fscheck -c /tmp/hostlist 80
harlem: / ==> 84.29% Used, 1538 KB Free
badger: /usr ==> 98.35% Used, 3656 KB Free
$
```

You can override any parameters specified in the script by invoking it explicitly. For example, when you issue:

```
sysctl -r /etc/fscheck
```

Sysctl ignores the **#!** line in the script. Since the **-L** option is not interpreted in this case, the output would be:

```
$ sysctl -c /tmp/hostlist /etc/fscheck 80
<<
>> harlem
/ ==> 84.29% Used, 1538 KB Free
<<
>> badger
/usr ==> 98.35% Used, 3656 KB Free
<<
```

Specifying a Host Target in a Script

Sysctl allows you to target a particular host within a script. The target host is specified in the body of the script via use of the **#host#** keyword. If the script contains a line beginning with **#!**, the **#host#** line must immediately follow it. Otherwise, **#host#** must be the first line in the script.

If you use the **-h** option to specify an SP node name when you invoke **sysctl**, any target host specified in the script is ignored. Here is a script, **/examples/sysctl/script3**, targeted to host **issc3**:

```
#!/usr/bin/sysctl -r
#host#issc3
echo I am $SCLHOST
```

Using the **-h** option to run the script overrides the host specified within the script. For example, if you specify:

```
sysctl -h r05n11 -r /examples/sysctl/script3
```

the script runs on **r05n11** instead of **issc3**.

Passing Parameters to a Script

When you issue the **sysctl** command with the **-r** parameter, any command line arguments following the script name are treated as parameters to be passed to the script. Here is a sample script, **/examples/sysctl/script6**:

```
#!/usr/bin/sysctl -r
#
echo Number of parameters is $argc
for {set i 0} {$i < $argc} {incr i 1} {
    echo Parameter #$i is \"$argv($i)\"
}
}
```

When you invoke the script with parameters **arg1** and **arg2**:

```
/examples/sysctl/script6 arg1 arg2
```

The output is the following:

```
Number of parameters is 3
Parameter #0 is script6
Parameter #1 is "arg1"
Parameter #2 is "arg2"
```

In this example, two parameters were passed to the script. The parameters are stored in the **argv** array, and the number of parameters is stored in variable **argc**, just as in the standard Tcl **exec** function. Because **argv[0]** always contains the name of the script being run, the value of **argc** is one greater than the number of arguments passed to the script. That is why the first line of output says that there are three parameters, when only two were passed to the script.

Sysctl Reference Information

Online Help Facility

To access online help on built-in Sysctl commands, enter:

```
sysctl help command-name
```

For example, to access help on the **setauth** command, enter:

```
sysctl help setauth
```

Tcl Commands

Table 7 shows the commands available in the Tcl language interpreter and the Sysctl base authorizations required to use them.

Table 7. Built-in Tcl Commands and Their Default Authorization Callbacks

append (AUTH)	array (AUTH)	break (AUTH)	case (AUTH)
catch (AUTH)	cd (ACL)	close (ACL)	concat (AUTH)
continue (AUTH)	eof (ACL)	error (AUTH)	eval (ACL)
exec (ACL)	exit (AUTH)	expr (AUTH)	file (ACL)
flush (ACL)	for (ACL)	foreach (ACL)	format (AUTH)
gets (ACL)	glob (ACL)	global (AUTH)	history (AUTH)
if (AUTH)	incr (AUTH)	info (ACL)	join (AUTH)
lappend (AUTH)	lindex (AUTH)	linsert (AUTH)	list (AUTH)
llength (AUTH)	lrange (AUTH)	lreplace (AUTH)	lsearch (AUTH)
lsort (AUTH)	open (ACL)	pid (ACL)	proc (SYSTEM)
puts (ACL)	pwd (ACL)	read (ACL)	regexp (AUTH)
regsub (AUTH)	rename (SYSTEM)	return (NONE)	scan (AUTH)
seek (ACL)	set (AUTH)	source (SYSTEM)	split (AUTH)
string (AUTH)	switch (AUTH)	tell (ACL)	time (ACL)
trace (ACL)	unset (AUTH)	uplevel (SYSTEM)	upvar (ACL)
while (ACL)			

Extended Tcl Commands

Table 8 shows the commands available in the TclX language interpreter and the Sysctl base authorizations required to use them.

Table 8. Built-in Extended Tcl Commands and Their Default Authorization Callbacks

alarm (ACL)	bsearch (ACL)	catclose (ACL)	catgets (ACL)
catopen (ACL)	cequal (AUTH)	cexpand (AUTH)	chgrp (ACL)
chmod (ACL)	chown (ACL)	chroot (SYSTEM)	cindex (AUTH)
clength (AUTH)	cmdtrace (ACL)	commandloop (ACL)	convertclock (AUTH)
copyfile (ACL)	crange (AUTH)	csubstr (AUTH)	ctoken (AUTH)
ctype (AUTH)	dup (ACL)	echo (AUTH)	execl (ACL)
fcntl (ACL)	flock (ACL)	fmtclock (AUTH)	fork (ACL)
frename (ACL)	fstat (ACL)	funlock (ACL)	getclock (AUTH)
id (ACL)	infox (ACL)	keyldel (AUTH)	keylget (AUTH)
keylkeys (AUTH)	keylset (AUTH)	kill (ACL)	lassign (AUTH)
lempy (ACL)	lgets (ACL)	link (ACL)	lmatch (AUTH)
loop (ACL)	lvarcat (AUTH)	lvarpop (AUTH)	lvarpush (AUTH)
max (ACL)	min (ACL)	mkdir (ACL)	nice (ACL)
pipe (ACL)	profile (ACL)	random (AUTH)	readdir (ACL)
replicate (AUTH)	rmdir (ACL)	scancontext (ACL)	scanfile (ACL)
scanmatch (ACL)	select (ACL)	server_open (ACL)	signal (ACL)
sleep (ACL)	sync (ACL)	system (ACL)	times (ACL)
translit (AUTH)	umask (ACL)	unlink (ACL)	wait (ACL)

Built-in Sysctl Commands

The following Sysctl commands are provided:

<i>Table 9. ACL Processing Commands</i>		
ACL Processing Commands	Default Auth	Description
acladd	ACL	Adds entries to an ACL file. If a filename is not specified, the entry is added to the server's ACL file.
aclcheck	ACL	Checks to see if a principal is in an ACL file. Returns 1 if in ACL, 0 if not in ACL. If no filename is specified, Sysctl looks in the server's ACL file.
aclcreate	ACL	Creates a new ACL file, inserting the principals specified. The -f parameter must be supplied.
acldelete	ACL	Deletes an ACL from a file.
acldestroy	ACL	Erases an ACL file.
aclist	NONE	Lists the ACLs in a file. If no file is specified, the server's ACL file is listed.
aclrecreate	ACL	Similar to aclcreate , but removes the target ACL file if it already exists before creating it again.

<i>Table 10. Service Commands</i>		
Service Commands	Default Auth	Description
svconnect	AUTH	Determines the connection authorization policy for the sysctld server.
svcdetach	ACL	Detaches the current process from the calling client; that is, breaks the connection between server and client. Use to start a background daemon.
svclog <i>string</i>	ACL	Writes the <i>string</i> to the server log file.
svclogevent	SYSTEM	Writes a line to the server log file indicating a user has connected to the server.
svcpid	NONE	Prints the process ID of the sysctld server daemon.
svcredirect <i>handle</i>	ACL	Redirects the stdout and stderr for the current session to a new file.
svcrestart	ACL	Restarts the server, causing configuration files to be reread and interpreters to be reinitialized. Use to activate changes to Sysctl servers.
svcversion	NONE	Returns the Sysctl version number.

<i>Table 11. Other Commands</i>		
Other Commands	Default Auth	Description
ACL	NONE	A supplied authorization callback.
AUTH	NONE	A supplied authorization callback.
NONE	NONE	A supplied authorization callback.
SYSTEM	NONE	A supplied authorization callback.
checkauth	NONE	Determines if you are authorized to access an object.
confadd	ACL	Adds items to a server's configuration file.
confdelete	ACL	Removes items from a server's configuration file.
create	SYSTEM	Creates new objects in the server.
getauth	NONE	Displays the authorization callbacks for an object.
include	SYSTEM	Includes additional configuration files.
listfs	ACL	Lists file system names.
load	SYSTEM	Dynamically load a shared library into the server and call an initialization routine within the library.
quit	NONE	Ends the current session.
safeargs <i>arg-list</i>	ACL	Checks arguments for illegal shell characters. Does not execute the command.
safeexec <i>commands ...</i>	ACL	Runs the Tcl exec or system command, checking arguments for illegal shell characters and preventing execution if any are found.
safesystem	ACL	Runs the Tcl exec or system command, checking arguments for illegal shell characters and preventing execution if any are found.
setauth	SYSTEM	Sets the authorization callback for an object.
statfs	ACL	Queries the status of all local file systems.
unload	SYSTEM	Unloads a previously loaded shared library.
whoami	NONE	Echoes the authenticated identity of the issuer or unknown if the user is not authenticated.

Managing Sets of Nodes

Chapter 14. Using a Switch

Using a switch can be a complex task. You should also consult the following documents:

- *PSSP: Installation and Migration Guide*. See the chapter on reconfiguring the system and refer to the section on installing a switch.
- *PSSP: Diagnosis Guide*. See the chapter on diagnosing switch problems.
- *IBM RS/6000 SP: Planning Volume 2, Control Workstation and Software Environment*.

This chapter describes the purposes of using a switch, the different types of supported switches, as well as the following:

- Switch communication modes
- Primary node takeover
- System partitioning and the switch
- Extension nodes and the switch
- Selecting switch clocks
- Selecting the primary and primary backup node
- Managing the switch topology file
- Starting the switch
- Automatic unfence of nodes
- Preparing for system repartitioning
- Monitoring nodes on a switch
- Global shutdowns and reboots of nodes with a switch
- NFS mounts over a switch

Understanding the Switch and the Supported Switch Types

A switch provides a means for nodes to communicate with each other faster and more efficiently. For example, a switch can dramatically speed up TCP/IP, file transfers, remote procedure calls, and relational database functions. A switch consists of a switch assembly and the internal cables to support connections to the processor nodes. A switch can provide some or all (depending upon the type of switch) of the following capabilities:

- Interframe connectivity and communication
- Scalability for increased number of nodes including intermediate switch frames
- Constant bandwidth and latency between node pairs
- Support for Internet Protocol (IP) communication between nodes
- IP Address Resolution Protocol (ARP) support
- Support for dedicated or multi-user environments
- Error detection and retry
- Fault isolation
- Concurrent maintenance for nodes

The switch supported on the SP system is the SP Switch. The SP Switch connects all the processor nodes, providing enhanced scalable high performance communication between processor nodes for parallel job execution. This switch can be ordered with 16 ports for use in a tall frame or with 8 ports for use in a short frame.

Usage Notes

1. Throughout this chapter, use of the term SP Switch applies to both the 8–port and 16–port unless specifically stated otherwise. Differences are noted in discussions of system partitioning, switch node numbering, and cabling.
2. Note that the setting of the AIX 4.2.1 base environment variable EXTSHM to ON in the /etc/environment for shared memory segment should not be used if PSSP is installed since PSSP functions need to be using the segment boundary instead of the page boundary. The default setting of EXTSHM is OFF and it should not be turned on. For more information on the EXTSHM variable, refer to the appropriate AIX documentation.

Switch Communication Modes

Two communication protocols are offered for the adapter:

1. AIX sockets using standard IP protocol
2. Dedicated user space access (via message passing libraries) and multiuser environments (via IP)

Both modes can be used concurrently, as if the adapter was actually two separate adapters.

Switch Communication Using IP (TCP/IP and UDP/IP)

The switch adapters are configured for IP by default during the installation process. You can assign specific IP network addresses statically during installation or dynamically using ARP (Address Resolution Protocol). When you assign the IP address statically, there are restrictions on mapping of nodes to low order addresses (see the information on switch node numbering in the *PSSP: Installation and Migration Guide*). When you assign the IP address range dynamically using ARP, there are no restrictions.

Any application can communicate over the switch by opening a standard socket and specifying appropriate IP addresses. Additional system facilities based on IP, such as NFS, and licensed programs such as AFS, can also be configured to use the switch. With the possible exception of applications that depend on network-specific functions (LAN broadcasts, for example), most applications work over the switch without modification.

Notes:

1. The **iptrace** command supports both switch adapters and provides interface-level packet tracing for incoming and outgoing packets.
2. The standard AIX commands **ifconfig** and **arp** have been modified to support the switch adapters. Refer to the *PSSP: Command and Technical Reference* for more information.
3. Static routes associated with the switch will not be automatically reconfigured during node reboot. The routes for the switch are associated with the default interface (**en0**) upon reboot because the switch is not available during the phase of system initialization when the static routes are reconfigured. If you

want the routes added automatically, IBM suggests that you put them in a script and have the script called as the last entry in `/etc/inittab`.

Switch Communication Using User Space Message Passing

The user space message passing mode supports parallel applications requiring maximum bandwidth and minimum latency. This mode is used by Parallel Environment for AIX to provide their exported message passing interfaces.

This mode uses *adapter windows* which are composed of instances of parallel tasks and protocols (either the Low Level Application Programming Interface (LAPI) or the Message Passing Interface (MPI)). Each adapter window is associated with a protocol and a task. A task running two protocols (such as MPI and LAPI) has two windows associated with it. The number of adapter windows sharing the SP Switch adapter cannot exceed 4.

For example, four adapter windows, each containing one parallel task and one protocol can share the following switch adapter. Note that the number of adapter windows and the number of protocols does not exceed four.

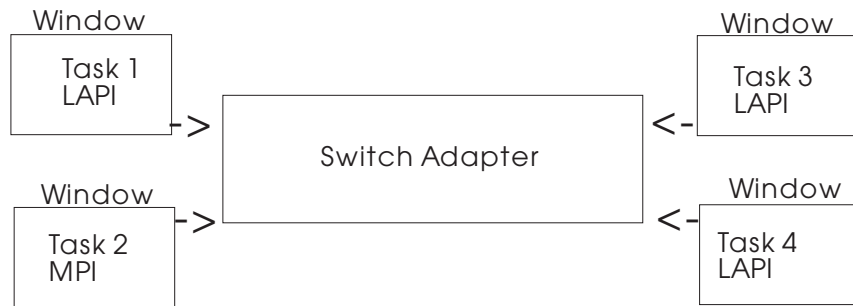


Figure 10. User Space Message Passing

Alternatively, for example, four adapter windows in the following graphic encompass three parallel tasks and four protocols. The third task uses both LAPI and MPI. Again, note that the number of adapter windows does not exceed four.

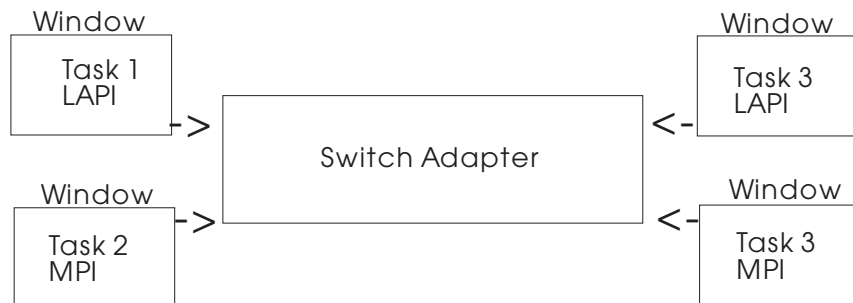


Figure 11. User Space Message Passing

The tasks can be part of the same job or of separate jobs. All tasks in the same job must use the same combination of protocols.

Primary Node Takeover

Error recovery on the SP Switch is done locally. On the SP Switch, detection and fault isolation are done at the link level while normal traffic flows across the rest of the fabric. Detected faults are forwarded to the active primary node for analysis and handling. When the primary node completes assessing the fault, the remaining nodes on the fabric are nondisruptively informed of status changes.

The primary node initializes the switch fabric and processes switch operations and errors reported from the switch fabric. The primary node is no longer a single point of failure on an SP system containing an SP Switch.

The primary backup node passively listens for activity from the primary node. When the primary backup node detects that it has not been contacted by the primary node for approximately 7 minutes, it assumes the role of the primary node. This takeover involves nondisruptively reinitializing the switch fabric, selecting another primary backup, and updating the System Data Repository (SDR).

The primary node also watches over the primary backup node. If the primary node detects that the primary backup node can no longer be contacted on the switch fabric, it selects a new primary backup node. The criteria to select a new primary backup node follows:

1. First, select a node on a switch assembly other than the switch assembly to which the primary node is attached.
2. Second, if no other switch assembly exists, select a node attached to a switch chip other than the one to which the primary node is attached.
3. If no other switch chip exists, select any available node on the switch chip to which the primary node is attached.

During the period of time between the failure or loss of the primary node and the takeover by the primary backup node, the switch fabric continues to function. Secondary failures are detected during the reinitialization of the switch fabric.

System Partitioning and the Switch

As discussed in Chapter 15, “Managing System Partitions” on page 221, the SP system can be organized into groups of nodes for various purposes. System partitioning has the following implications for the switch:

- Commands related to the switch:
 - **Eannotator** - Operates only within the current system partition
 - **Eclock** - Always operates globally (across system partitions)
 - **Efence** - Can operate globally (**-G**) or only within the current system partition
 - **Eprimary** - Operates only within the current system partition
 - **Equiesce** - Operates only within the current system partition
 - **Estart** - Operates only within the current system partition
 - **Etopology** - Operates only within the current system partition

- **Eunfence** - Can operate globally (-G) or only within the current system partition
- **Eunpartition** - Operates only within the current system partition
- Switch faults and message traffic are contained *within a system partition*.
- The subset of the switch defined by the topology file for the system partition is viewed as a logical switch.
- There is no connectivity over the switch *between system partitions*. (A gateway node with routing set up to the switch network may require routing changes if the gateway is to remain a gateway for more than one system partition. Use the explicit host routes on the gateway node or enable ARP on all system partitions and redefine the IP addresses within a system partition as a different subnetwork.)
- There remains one switch clock source, regardless of system partitioning.
- There is one primary node per system partition.

Extension Nodes and the SP Switch

Extension nodes are non-standard nodes that extend the SP system's capabilities or scope, but cannot be used in all the same ways as standard SP nodes.

A specific type of extension node is a dependent node. Dependent nodes are attached to the SP Switch via a special dependent node adapter.

A dependent node implements much of the switch-related protocol that standard nodes use on the SP Switch, but it depends on standard SP nodes for certain functions. A dependent node can use the SP Switch as an IP network, but it relies upon a standard SP node to generate its switch route table. (The communication subsystem does not send dependent nodes switch topology database updates, but rather generates and sends appropriate switch route tables.)

SNMP support provides for the communication of switch-related configuration information between the SP system and the dependent node.

A dependent node cannot be a primary or primary backup node for the SP Switch.

The **CSS_test**, **Efence**, and **Eunfence** commands operate on dependent nodes in the same way they operate on standard nodes. The **Eprimary** command returns an error message if a dependent node is specified for a primary or primary backup node for the SP Switch. The **Eannotator** command notes dependent nodes in the annotated topology file.

For more information on extension nodes and dependent nodes, refer to Chapter 17, "Managing Extension Nodes" on page 251.

SP-Attached Servers

SP-attached servers are stand-alone servers that attach to the SP and might include a switch attachment. From a switch administration point of view, these nodes are fully functional. This means, they can serve as a Switch Primary or Primary Backup node just like any other SP node that sits in an SP frame.

Selecting Appropriate Switch Clocks

One of two switches in the system provides the master switch clock to the rest of the system. If the current master clock fails, or is removed from the configuration, the alternate master clock must be selected and the switch reinitialized.

The **Eclock** command establishes a master clock source after the system is powered up or when an alternate must be selected. It can set the appropriate clock input for every switch in the system or for a single switch after power on. After **Eclock** completes, the **Estart** command must be run.

Warning: Running **Eclock** after the switch is started may cause nodes to lose their clock source. Refer to **Eclock** in the *PSSP: Command and Technical Reference* for more information.

Selecting the Switch Clock Source

To set the switch clock source using SMIT:

TYPE `smit`

- The System Management menu appears.

SELECT RS/6000 SP Systems Management

- The RS/6000 SP Systems Management menu appears.

SELECT RS/6000 SP Cluster Management

- The Cluster Management menu appears.

SELECT Perform Switch Operations

- The Perform Switch Operations menu appears.

SELECT Change/Show Switch Clock Source Settings

- The Change/Show Switch Clock Source Settings menu appears.

At this point, you can select options:

- Change/Show Switch Clock Source Settings

 - List Switch Clock Source Settings
 - Initialize Switch Clock Source Settings
 - Initialize Switch Clock Source Settings for a Switch
 - Restore the Switch Clock Source Settings
 - Select Alternate Switch Clock Source Settings
 - Create an Eclock Topology File
 - Automatic selection of Switch Clock Settings

Figure 12. The Change/Show Switch Clock Source Settings SMIT Menu

When you choose a switch clock source setting option, you are prompted for a topology file name, and then the **Eclock** command is invoked.

The fastpath invocation for this menu is:

```
smit clock_src
```

Resetting the Switch Clock

The switch clock is synchronous across all nodes active on a switch. Its value is set to zero when the primary node powers on, and is later distributed by **Estart** to all nodes on or joining the switch. The switch clock will increase monotonically unless it is reset by one of the following events:

- The switch is powered off.
- An oscillator failure occurs on the switch board designated as the master clock source. (The switch designated as the master clock source provides the clocking signal for all other switch boards within the configuration.)
- The **Eclock** command is issued.
- The **Estart** command is issued while the oncoming primary node (or the primary node on High Performance Switch systems) is not active on the switch (for example, after the primary node is powered on).
- The **Estart** command is issued after **rc.switch** or reboot has run on the primary node, but before the primary backup node has taken over. For further information, see “Primary Node Takeover” on page 206.

Note: IBM suggests that a new primary node that is active on the switch be selected before the old primary node is powered off. This will prevent the switch clock from being reset to zero or to an indeterminate value on High Performance Switch systems. Refer to **Eprimary** in the *PSSP: Command and Technical Reference* for more information on selecting a primary node.

Selecting the Primary and Primary Backup Nodes

On SP Switch systems, select two standard nodes, one to be used as the primary node and the other to be used as the primary backup node. Note that a dependent node cannot be a primary or primary backup node for the SP Switch. Until the next **Estart** is issued, these nodes are reflected as the oncoming primary and oncoming primary backup nodes (**Eprimary**). Once **Estart** completes, the primary field is updated based on the oncoming primary field and the backup field is updated based on the oncoming backup field.

The criteria to select a new primary backup node follows:

1. First, select a node on a switch assembly other than the switch assembly to which the primary node is attached.
2. Second, if no other switch assembly exists, select a node attached to a switch chip other than the one to which the primary node is attached.
3. If no other switch chip exists, select any available node on the switch chip to which the primary node is attached.

The primary and primary backup fields are updated automatically when primary node and primary backup node failures occur. Therefore, the primary and primary backup fields reflect the state of the current system.

The following scenario describes how to select a primary backup node on an SP Switch system.

Initially, the primary node and backup primary node fields are blank. In a 16-node system, the oncoming primary node field has the default value of 1 and the oncoming primary backup has the default value of 16.

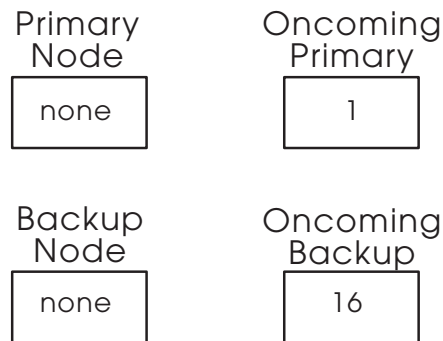


Figure 13. Initial Values

When **Estart** is executed, the node specified in the oncoming primary field becomes the primary node. The node specified in the oncoming primary backup becomes the primary backup.

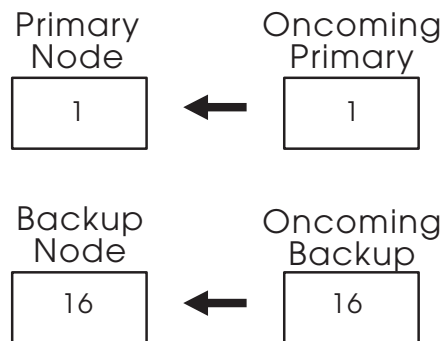


Figure 14. Estart Values

If the primary backup node fails, the primary node automatically selects a replacement.

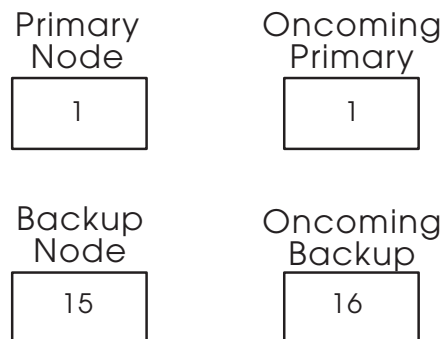


Figure 15. Backup Takeover Values

If the primary node fails, the primary backup node automatically becomes the primary node and a new primary backup is selected.

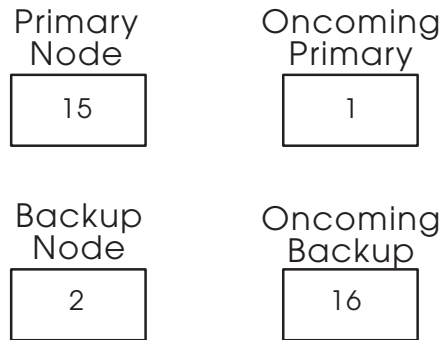


Figure 16. Primary Takeover Values

In summary, primary and primary backup fields reflect the current state of the system and the oncoming fields are not applicable until the next invocation of the **Estart** command.

To select the primary and primary backup nodes using SMIT:

TYPE **smit**

- The System Management menu appears.

SELECT RS/6000 SP Systems Management

- The RS/6000 SP Systems Management menu appears.

SELECT RS/6000 SP Cluster Management

- The RS/6000 SP Cluster Management menu appears.

SELECT Perform Switch Operations

- The Perform Switch Operations menu appears.

SELECT Set Primary/Primary backup Node

- The Set Primary/Primary Backup Node menu appears.

Using the Command Line

On SP Switch systems, to select an oncoming switch primary node and an oncoming switch primary backup node by IP address, use the command:

```
Eprimary 129.33.34.1 -backup 129.33.34.56
```

Managing the Switch Topology File

The switch topology file describes the wiring configuration for the switch; it contains node-to-switch or switch-to-switch cable information. You choose a switch topology file based on the number of **switches** in your SP system. (For more information on choosing a switch topology file, refer to the *PSSP: Installation and Migration Guide*.) You can annotate connection labels onto the selected switch topology file with the **Eannotator** command. Once you have selected and annotated the switch topology file, you store it in the System Data Repository (SDR) with the **Etopology** command. When you start the switch, the switch initialization code reads the switch topology file stored in the SDR. (Refer to the *PSSP: Command and Technical Reference* for more information on the **Eannotator** and **Etopology** commands.)

An **expected.top** file in the **/etc/SP** directory of the primary node overrides the topology file specified in the SDR. (This is used when servicing or debugging the switch. Refer to the *IBM RS/6000 SP: Maintenance Information Volume 2* for more information.)

Annotating a Switch Topology File

Before a switch topology file is stored in the SDR, it should be annotated with **Eannotator**. **Eannotator** will update the switch topology file's connection labels with their current physical locations. If the **-O yes** flag is specified, the topology file is stored in the SDR. Running **Eannotator** makes the switch hardware easier to debug because the switch diagnostics information is printed based on physical locations. The base topology files only contain logical locations. **Eannotator** is required to update jack information for switch diagnostics. For more details on **Eannotator**, refer to the *PSSP: Command and Technical Reference*.

Note that for the SP Switch, if you have annotated your topology file and any nodes have been added, you must re-annotate and store the topology in the SDR. You then must issue the **Estart** command in order to bring any new nodes into the switch network.

Using SMIT

TYPE `smit`

- The System Management menu appears.

SELECT RS/6000 SP Systems Management

- The RS/6000 SP Systems Management menu appears.

SELECT RS/6000 SP Cluster Management

- The RS/6000 SP Cluster Management menu appears.

SELECT Perform Switch Operations

- The Perform Switch Operations menu appears.

SELECT Topology File Annotator

- The Topology File Annotator menu appears.

At this menu:

- Input the name of the topology file to be annotated
- Input the name to which you want the annotated topology file stored
- Enter **yes** to storing the topology file in the SDR

Using the Command Line

To annotate a switch topology file, use the command:

```
Eannotator -F input_file -f output_file -O yes
```

For example, to annotate a topology file for a two-switch or maximum 32-node system:

```
Eannotator -F expected.top.2nsb.0isb.0 -f expected.top.2nsb.0isb.0 -O yes
```

Storing the Switch Topology File in the SDR

Use **Etopology** to store the topology file in the SDR in the current system partition:

Using SMIT

TYPE smit

- The System Management menu appears.

SELECT RS/6000 SP Systems Management

- The RS/6000 SP Systems Management menu appears.

SELECT RS/6000 SP Cluster Management

- The RS/6000 SP Cluster Management menu appears.

SELECT Perform Switch Operations

- The Perform Switch Operations menu appears.

SELECT Fetch/Store Topology File

- The Fetch/Store Topology File menu appears.

Input the topology file for your configuration.

Using the Command Line

To store a topology file, use the **Etopology** command:

```
Etopology topology_filename
```

For example, to store a topology file for a two-switch or 32-node maximum configuration:

```
Etopology expected.top.2nsb.0isb.0
```

Starting the Switch

Once all the processor nodes are running, you can start the switch.

1. Check that all processor nodes are running (hostResponds indicator).
2. Start the switch. On the control workstation, enter:

```
Estart
```

(**fault_service_Worm_RTG_SP**, the fault-handling daemon for the SP Switch, that check, initialize, and prepare the switch for operation, must be running on the primary node.)

Note: Before you issue an **Estart**, if the daemon is not running, use the **rc.switch** command to start the daemon.

When **Estart** finishes, it sends a message similar to the following example verifying that it completed:

```
Switch initialization started on tserv10.hpssl.kgn.ibm.com
Initialization successful for 16 nodes
Switch initialization completed
```

If the switch initialization process encounters any problems, other messages may appear. If this happens, refer to diagnosis information in the *PSSP*:

Diagnosis Guide and resolve any errors. See the *PSSP: Command and Technical Reference* for more information on **Estart**.

Determining Switch Connectivity

You can monitor switch connectivity to the nodes by using SP Perspectives. For specific information on how to do this, view the online help available from the SP Hardware perspective application.

Automatic Node Unfence

As of the PSSP 3.1 release, the default is for nodes to rejoin the switch communication fabric without any expressed action by the operator. This differs from the default of past releases which required the operator to run either the **Estart** or **Eunfence** commands to get nodes to talk on the switch fabric. This function is built into the fault service daemon and replaces the Emonitor daemon function. Experienced users will notice that the *autojoin* attribute of the SDR **switch_responds** class is set whenever nodes join the switch fabric, which is different than past releases. The autojoin attribute being set has the effect of signaling the switch primary node to unfence it once it is fully operational.

With automatic unfence, if you want to fence a node off the switch fabric and not have it rejoin, you must fence the node with the **Efence** command which turns off the autojoin attribute in the **switch_responds** class. If you do not have the autojoin attribute set, the fault service daemon will not unfence it during **Estart** or automatically. The node will remain fenced until either it is unfenced using the **Eunfence** command or the autojoin attribute is set in the SDR.

The following example shows all the states a node **switch_responds** object could be in and how they are treated by the primary.

node_number	switch_responds	autojoin	isolated	Description
1	1	X	X	up on switch
2	0	0	1	Fenced isolated
3	0	1	1	Fenced with autojoin

When a node is up on the switch fabric, it does not matter how the "isolated" or "autojoin" attributes are set. It will remain on the switch until it is fenced, rebooted, or shutdown. The opposite is true of a node that is fenced "isolated". It will remain off the switch fabric until it is unfenced or "autojoin" attribute is set. Nodes that are fenced with their autojoin attribute set will get unfenced automatically by the switch primary.

If you do not want to have nodes automatically join the switch, you can turn off automatic unfence. The behavior will be the same as the default of PSSP 2.4 and earlier releases. The *autojoin* attribute will be turned off whenever the node joins the switch. If the node is fenced, it will remain fenced until it is unfenced using the **Eunfence** command. You can still fence individual nodes with the autojoin option which will allow the node to be unfenced automatically when the node is rebooted or when the fault service daemon is restarted.

The default is to have automatic unfence enabled. To turn automatic unfence off (or to enable it again after having turned it off), use the **Estart** command. To turn automatic unfence off, issue the command:

```
Estart -autounfence 0
```

To turn automatic unfence on, issue the command:

```
Estart -autounfence 1
```

See the descriptions of the **Efence**, **Eprimary**, and **Estart** commands in the book PSSP: Command and Technical Reference for more information about using automatic unfence.

Fence the Node From the Switch

To keep a node off the the switch fabric that is about to undergo maintenance or service, issue the **Efence** command on the control workstation. For example:

```
Efence node1 node2
```

Once the command completes, powering the node up, powering it down, or rebooting it will not affect the switch.

Return the Node to the Switch

To bring a node previously fenced back into the switch fabric, use the **Eunfence** command on the control workstation. For example:

```
Eunfence node1
```

Alternatively, you can use the Hardware Perspective to unfence the node if it is already powered on. If the node is powered off, you can automatically bring the node back on the switch when powering it on. Display the Power On or Cluster Power On dialog for the specific node, select a power on option and set the “Enable Autojoin - automatically bring the node back in the switch after it is powered up” check box.

The rc.switch /etc/inittab Entry

The rc.switch /etc/inittab entry can be changed from the default of **once** to **wait**. This change, in combination with the autojoin function, will stop the execution of the subsequent entries in the **/etc/inittab** file until the switch's css0 IP interface comes up. This allows for entries that require the switch to be up to be put after the rc.switch entry.

The use of the **once** parameter in the following example indicates that the execution of the subsequent entries will continue in the **/etc/inittab** file until the switch's css0 IP interface comes up:

```
fsd:2:once:/usr/lpp/ssp/css/rc.switch
```

The use of the **wait** parameter in the following example indicates that the execution of the subsequent entries will wait for the switch IP interface to come up before continuing on to run **/etc/inittab**.

```
fsd:2:wait:/usr/lpp/ssp/css/rc.switch
```

The rc.switch will wait for the unfence function to bring up the switch's IP interface after starting the switch's fault service daemon. It will only wait 3 consecutive switch can periods or approximately 6 minutes. After 6 minutes, the rc.switch will return allowing the rest of inittab to be executed.

Preparing an SP Switch for Repartitioning

Use the **Eunpartition** command to prepare the current system partition for repartitioning. For example:

```
Eunpartition
```

Optionally, you can use a SMIT panel to prepare a switch for repartitioning.

Using SMIT

TYPE **smit**

- The System Management menu appears.

SELECT RS/6000 SP Systems Management

- The RS/6000 SP Systems Management menu appears.

SELECT RS/6000 SP Cluster Management

- The RS/6000 SP Cluster Management menu appears.

SELECT Perform Switch Operations

- The Perform Switch Operations menu appears.

SELECT Prepare a switch network for repartitioning

- The Prepare a switch network for repartitioning menu appears.

See the *PSSP: Command and Technical Reference* for more information on the **Eunpartition** command.

Monitoring Nodes on a Switch

You can monitor and maximize the availability of nodes on a switch with the **Emonitor** daemon. **Emonitor** is controlled by the System Resource Controller (SRC). One instance of the daemon exists for each system partition and is named **Emonitor.partition_name**. A system-wide configuration file, **/etc/SP/Emonitor.cfg** lists all node numbers (one per line) on the system to be monitored.

Emonitor is invoked with **Estart -m**. Once invoked, it is SRC controlled and so will restart if halted abnormally. To end monitoring, run **/usr/lpp/ssp/bin/emonctrl -s** to stop the daemon in the system partition.

Refer to *PSSP: Command and Technical Reference* for more information on **Emonitor**.

Switch Admin Daemon (cssadm)

The **cssadm** daemon runs on the control workstation and it interacts with the Event Management subsystem to obtain information regarding node and node adapter status. The daemon also provides node recovery on the switch. The daemon is started from **/etc/inittab** on the control workstation, or HACWS, and is controlled by SRC. The SRC subsystem name for this daemon is **swtadm**.

Based on information it obtains from the Event Management subsystem regarding the state of nodes and node adapters, the daemon handles the following events:

- Node joins Host membership or node leaves Host membership (IBM.PSSP.Response.Node.state)

If a node comes up in this state, the node is checked to see if it is the oncoming primary node in its partition. If it is the oncoming primary node, `cssadm` executes an **Estart**.

If the **Estart** command fails, a message is written to the `cssadm` log file and no further action is taken.

- Node leaves Switch Adapter membership (IBM.PSSP.Membership.LANAdapter.state)

If this node is the primary node, the daemon checks to see if there is an active primary backup node. If there is an active primary backup node, the daemon will not intervene because the primary backup node will begin functioning as the backup. If an active primary backup node does not exist, the oncoming node is checked and if the node is up on host responds, `cssadm` attempts an **Estart**.

If this node is not the primary and the primary is "none" in the node's partition, `cssadm` checks to see if there is an active primary backup node. If there is an active primary backup node, the daemon will not intervene because the primary backup node will begin functioning as the backup. If an active primary backup node does not exist, the oncoming node is checked and if the node is up on host responds, `cssadm` attempts an **Estart**.

Node event handling operates on a system partition basis. Events from each of the membership groups are handled based on the effect that event would have on that system partition.

HACWS Considerations: In an HACWS environment, do not add an **/etc/inittab** entry to start the `cssadm` daemon because in the HACWS environment, the HACWS function starts the `cssadm` daemon on either the primary or backup control workstation.

Node Recovery

As stated previously, the `cssadm` daemon interacts with the Event Management subsystem. It also interacts with a configuration file to enable node recovery. The configuration file resides at **/spdata/sys1/ha/css/cssadm.cfg** and it contains one line which reads as follows:

```
Node 1
```

If the value is set to 1, the daemon is triggered by the Event Management events described previously. If you do not want the daemon to react to the events, you can disable node recovery by changing the 1 to a 0 and stopping the `cssadm` daemon by issuing the **stopsrc -s swtadmd** command. After you restart the daemon, the new value in the configuration file will be picked up.

Starting and Stopping the `cssadm` Daemon

The daemon is added to SRC as the subsystem `swtadmd` and is started from **/etc/inittab** automatically. However, if you do not want the daemon to run, issue:

```
stopsrc -s swtadmd
```

To permanently stop the daemon from running, remove the subsystem entry from the **/etc/inittab** file.

To restart the daemon, issue:

```
startsrc -s swtadmd
```

cssadm Log Files

The daemon generates the following log files all located in `/var/adm/SPlogs/css`:

- `cssadm.debug`

This file contains trace information of the actions of the daemon. It contains entries for each event received and handled, as well as how the events were handled and the results.

- `cssadm.stderr`

This file contains any unexpected error messages received by the daemon while performing commands external to the daemon.

- `cssadm.stout`

This file contains any unexpected informational messages received by the daemon while performing commands external to the daemon. In general, this log should remain empty.

css.summlog Daemon

This daemon provides a summary log of switch-related AIX error log entries from all nodes in an SP in one convenient location on the control workstation. It provides a summary of switch errors across the entire system, ordered by time and tagged with identifying information, which can serve as the starting point for switch-related diagnosis. The name of this file is `/spdata/sys1/ha/css/summlog`.

For additional information regarding switch-related diagnostic information, see the *PSSP: Diagnosis Guide*.

Starting and Stopping the css.summlog Daemon

To stop the daemon from running, issue:

```
stopsrc -s swtlog
```

To permanently stop the daemon from running, remove the subsystem entry from the `/etc/inittab` file.

To restart the daemon, issue:

```
startsrc -s swtlog
```

css.summlog Log Files

The daemon generates the following log files all located in `/var/adm/SPlogs/css`:

- `logevnt.out`

This file contains records of errors which occurred in the components running on the node which experienced the error. These components are notified when switch-related error log entries are made, and report the summary data to Event Management for transmission to the control workstation. The log is a text file and may exist on each node.

- `summlog.out`

This file contains error information for the daemon which gathers summary log information and writes it to the summary log file. This log is a text file and exists on the control workstation.

Global Shutdowns and Reboots of Nodes with a Switch

The **Equiesce** command disables switch error recovery and primary node takeover. It is used to shut off the normal error actions when global activities are performed. For example, when nodes are shut down or rebooted, they are fenced from the switch.

It is possible that the primary node may fence all the other nodes, including the primary backup node, if it is not the first node to shut down during a global shutdown/reboot of the entire system. Primary node takeover can also occur if the primary node is shut down and the primary backup node remains up. Issuing the **Equiesce** command before the shutdown will prevent these situations from occurring.

The **Equiesce** command causes the primary and primary backup nodes to shut down their recovery actions. Data still flows over the switch, but no faults are serviced and primary node takeover is disabled. Only the **Eannotator**, **Eclock**, **Eprimary**, **Estart**, and **Etopology** commands are functional after the **Equiesce** command is issued.

Estart must be issued when the global activity is complete to reestablish switch recovery and primary node takeover.

Use the **Equiesce** command to disable switch error recovery and primary node takeover. For example:

```
Equiesce
```

Optionally, you can use a SMIT panel to quiesce the SP Switch.

Using SMIT

TYPE `smit`

- The System Management menu appears.

SELECT RS/6000 SP Systems Management

- The RS/6000 SP Systems Management menu appears.

SELECT RS/6000 SP Cluster Management

- The RS/6000 SP Cluster Management menu appears.

SELECT Perform Switch Operations

- The Perform Switch Operations menu appears.

SELECT Switch Quiesce

- The **Equiesce** command is run.

See *PSSP: Command and Technical Reference* for more information on the **Equiesce** command.

NFS Mounts Over a Switch

IBM does not support NFS mounts over the switch in the root (/) directory. If a mount is necessary in the root (/) directory, place the NFS mount elsewhere in the file system with a symbolic link into the root (/) directory. For example:

```
mount -o intr,hard server:/myfs /otherdir/myfs
ln -s /otherdir/myfs /myfs
```

Chapter 15. Managing System Partitions

System partitioning is a method for organizing the SP system into non-overlapping groups of nodes for various purposes such as testing new software and creating multiple production environments. In this chapter we present information on:

- Understanding system partitioning
- Partitioning the system
- Displaying system partition configuration information
- Verifying system partition configuration
- Managing system partition-sensitive subsystems using the **syspar_ctrl** command

For information on accounting and system partitioning, refer to “Accounting and System Partitioning” on page 118.

Before proceeding with this chapter, you should have installed the IBM Parallel System Support Programs for AIX (PSSP) software. In doing so you have automatically created a single default system partition which consists of all the nodes in your SP system. If you now wish to select an alternate system partition configuration, the information in this chapter will guide you in selecting, customizing, and applying a system partition configuration.

Understanding System Partitioning

System partitioning is a method for organizing an SP system into non-overlapping groups of nodes that can be dedicated to specific purposes. A group of such nodes (not including the control workstation) is a *system partition*. For example, you may wish to configure a system partition to use for non-disruptive testing when you migrate to a new release of software. Or you may wish to configure system partitions for multiple isolated production environments so that the workload in one environment will not interfere with the workload in another environment. Isolation of production environments is achieved by partitioning the switch in such a way that communication paths for different system partitions do not cross each other.

Note that SP-attached servers and dependent nodes can be part of a system partition along with standard nodes. Partitioning is performed according to the switch port to which the server or dependent node is connected.

System partitions appear to most subsystems and for most user tasks as logical SP systems. It is important to understand that from an administrative point-of-view, each system partition is a logical SP system within one administrative domain.

On the control workstation, the administrator is in an environment for *one system partition at a time*. The SP_NAME environment variable identifies the system partition to subsystems. (If this environment variable is not set, the system partition is defined by the primary: stanza in the **/etc/SDR_dest_info** file.) Most tasks performed on the control workstation that get information from the SDR will get the information for *that particular system partition*.

In managing multiple system partitions it is helpful to open a window for each system partition. You can set and export the SP_NAME environment variable in

each window and set up the window title bar or shell prompt with the system partition name. The following script is an example:

```
sysparenv:
# !/bin/ksh
for i in 'splst_syspars'
do
    syspar='host $i | cut -f 1 -d".'"
    echo "Opening the $syspar partition environment"
    sleep 2
    export SP_NAME=$syspar
    aixterm -T "Work Environment for CWS 'hostname -s' - View: $syspar" -ls -sb &
done
exit
```

```
| .profile addition:
| # Added for syspar environment setup
| if [ "'env | grep SP_NAME | cut -d= -f1'" = SP_NAME ]
| then
|     PS1="[`hostname -s`<p>$SP_NAME] [``$PWD]> '
| else
|     PS1="[`hostname -s`][``$PWD]>'
| fi
| export ENV
```

As a user, you can check what system partition you're in with the command:

```
spget_syspar -n
```

You can achieve system partitioning either by applying one of a fixed set of supplied configurations or by creating a different system partition configuration and applying it. (See the book *IBM RS/6000 SP: Planning Volume 2, Control Workstation and Software Environment* for specific information on supplied configurations). The **ssp.top** option in the **pssp** install package contains a directory structure with all relevant files for partitioning the system with supplied system partition configurations. In addition, the **ssp.top** option contains the System Partitioning Aid, a tool for creating customized system partitions to suit your needs. Start the System Partitioning Aid from the Perspectives Launch Pad or with the **spsyspar** command. This option must be installed on the control workstation if you are partitioning your system into more than one system partition.

SP Switch Usage Note

If you are repartitioning your system and it has an SP Switch, you must run **Eunpartition**. Failure to do this will produce unpredictable **Estart** results in the new system partitions. The **Estart** command results can range from only the old system partition portion of the new system partition being accessible on the switch, to a complete **Estart** timeout failure. Various recovery scenarios are available. A guaranteed recovery can be accomplished by:

1. Issuing **Eclock** to reset the switch
2. Rebooting all the nodes or issuing a **css_restart_node** on all nodes
3. Issuing **SDRChangeAttrValues switch_responds isolated=0**
4. Issuing **Estart** in each of the system partitions

This recovery scenario is disruptive to all system partitions, even those unaffected by the repartition.

Before You Define System Partitions

In the beginning, after you have installed the PSSP software, but before you have configured system partitions, there is one system partition corresponding to the name and IP address of the control workstation returned by the **hostname** command. This is the *default* or *persistent* system partition. It always exists. When you configure more than one system partition, one of them must be defined as the default system partition with this name and IP address.

When configuring more than one system partition, you must define alias IP addresses and names for all but the default system partition. To configure an alias IP address, you use the **ifconfig** command with the alias parameter to establish an additional IP address for the interface to which the control workstation's **hostname** map.

Note: The alias must be defined on the same subnet as the interface that the control workstation **hostname** maps to.

You should also add alias names that map to these alias IP addresses through your name resolution mechanism. The alias IP addresses should be added to **rc.net** so that they are automatically configured when the control workstation reboots.

The steps for defining alias IP addresses are:

1. In the Part II - Traditional Configuration section of the **/etc/rc.net** file, a template is provided for changing **inet0** to an alias. Follow the template and edit the file to define alias IP addresses and names. (After editing the **/etc/rc.net** file, ensure that you have execute permission on the file or the control workstation will not boot correctly.) For example:

```
/usr/sbin/ifconfig tr0 alias 129.40.127.101 netmask 255.255.255.0  
up >>$LOGFILE 2>&1;
```

2. Execute the **/etc/rc.net** script. Alternatively, you could enter the **ifconfig** command directly. For example:

```
ifconfig tr0 alias 129.40.127.101 netmask 255.255.255.0 up
```

3. Validate that the alias is defined correctly (that is, that the same interface is defined for the alias and the control workstation host name). For example, to list the IP addresses and aliases on your system:

```
netstat -in
tr0    129.40.127    129.40.127.43
tr0    129.40.127    129.40.127.101
```

Note: When you add an alias for the control workstation for the purpose of defining a name for a new system partition, you must run the **setup_server** command on the control workstation to properly define the new **rcmd** principal associated with the new host name alias.

Note: To delete the alias we defined in the previous example, the command is:

```
/usr/sbin/ifconfig tr0 delete 129.40.127.101
```

Remember to comment out or delete any entries you made to define aliases in the **/etc/rc.net** file.

In the next section we discuss how you select, customize, and apply a system partition configuration.

Partitioning the SP System

You partition your SP system by choosing a configuration for your size system. The steps involved in partitioning an SP system are:

1. Check if the desired system partition configuration is supplied.
2. If the desired system partition configuration is not supplied, use the System Partitioning Aid to generate and save a system partition configuration.
3. Archive the System Data Repository.
4. Select a system partition configuration and layout to apply.
5. Customize (name and specify attributes) the system partitions.

Note: If you are repartitioning your system, and your system contains an SP Switch, you must run **Eunpartition** in each of the partitions that are going to change. This prepares the switch in those partitions for repartitioning.

6. Shut down all nodes in changing system partitions.
7. Run the **setup_server** command on the control workstation.
8. Apply the configuration.
9. Reboot all nodes in changed system partitions.

The previous steps are referred to as *partitioning* the SP system. However, if you wish to change or add system partitions, that is, to *repartition*, the steps are the same. For example, if you wish to modify an existing system partition or create a new partition for a new production environment, you would follow these steps to repartition the system.

How to Start

If you know that the desired system partition configuration exists, start with “Step 3: Archive the System Data Repository” on page 227; otherwise, start with “Step 1: Check if the Desired System Partition Configuration is Supplied” on page 226.

For partitioning or repartitioning the system, you have the option of using SMIT panels or commands. SMIT panels can be accessed through the **smit** command or through the SP Perspectives Launch Pad. (For more information, and further examples, of the commands used in this chapter, refer to the *PSSP: Command and Technical Reference*.)

Note

Some of the commands used in system partitioning (**spdisplay_config**, **splstdata -p**, **spapply_config -v**, for example) produce lengthy output. It is helpful to log the output of such commands. For example, to execute the **spapply_config** command while simultaneously logging a copy of the output to the file **/tmp/syspar.log**, enter:

```
spapply_config -v config.4_4_8/layout.3 2>&1 | tee /tmp/syspar.log
```

Using SMIT

- TYPE** **smit**
- The System Management menu appears.
- SELECT** RS/6000 SP System Management
- The RS/6000 SP System Management menu appears.
- SELECT** RS/6000 SP Configuration Database Management
- The RS/6000 SP Configuration Database Management menu appears.
- SELECT** Enter Database Information
- The Enter Database Information menu appears.
- SELECT** System Partition Configuration
- The System Partition Configuration menu appears.

Using the SP Perspectives GUI to Access SMIT

- TYPE** **perspectives &**
- The SP Perspectives Launch Pad appears.
- SELECT** smit config_data on CWS
- The RS/6000 SP Configuration Database Management menu appears in a separate AIX window.
- SELECT** Enter Database Information
- The Enter Database Information menu appears.

SELECT System Partition Configuration

- The System Partition Configuration menu appears.

Step 1: Check if the Desired System Partition Configuration is Supplied

Use the System Partitioning Aid to check if the desired system partition configuration is supplied. The System Partitioning Aid can be invoked from the command line or from the SP Perspectives Launch Pad. This section provides examples using the command line interface. (For information on the SP Perspectives GUI to the System Partitioning Aid, invoke the Aid with the **spsyspar** command and view the online help that is available.) Refer to *IBM RS/6000 SP: Planning Volume 2, Control Workstation and Software Environment* for more information on using the System Partitioning Aid.

To check if the desired system partition configuration is supplied:

1. Enter information for your desired system partition configuration into a file. The template for this file is **/spdata/sys1/syspar_configs/bin/infile.template**.
2. Once you have created the input file, enter:

```
sysparaid input_file
```
3. If the desired system partition configuration is found, layout directory information is displayed. Go to “Step 3: Archive the System Data Repository” on page 227.
4. If the desired system partition configuration is not found, a message is displayed indicating so and stating whether the desired system partition configuration is possible or not possible. If the system cannot be partitioned as desired, an error message is output indicating the reason for the failure. If the desired system partition configuration is possible, a snapshot of the switch chips in the system and performance data for each system partition is output to a set of files.
 - a. If the system cannot be partitioned as desired, analyze the error message and create another input file specifying an alternative system partition configuration and rerun the **sysparaid** command.
 - b. If the desired system partition configuration is possible, review the performance data that is output and, if it is acceptable, proceed to “Step 2: Generate and Save the Desired System Partition Configuration.”

Refer to the *PSSP: Command and Technical Reference* for more information on the **sysparaid** command.

Step 2: Generate and Save the Desired System Partition Configuration

Once you have determined that the system partition configuration you desire is possible, generate and save it:

```
sysparaid -s layout_name input_file
```

The newly generated system partition configuration layout will be saved to the **spdata/sys1/syspar_configs** directory.

The fastpath invocation for the System Partition Configuration menu is:

```
smit syspar
```


Step 3: Archive the System Data Repository

Always archive the System Data Repository before partitioning (or repartitioning) your SP system. If you change your mind once you have committed a system partition configuration, or if applying a system partition configuration fails, you should use the archived SDR to recreate the previous system partition configuration. (To do this, select the Restore System Partition Configuration option from the SMIT menu.)

Using SMIT

At the System Partition Configuration menu:

SELECT Archive System Data Repository

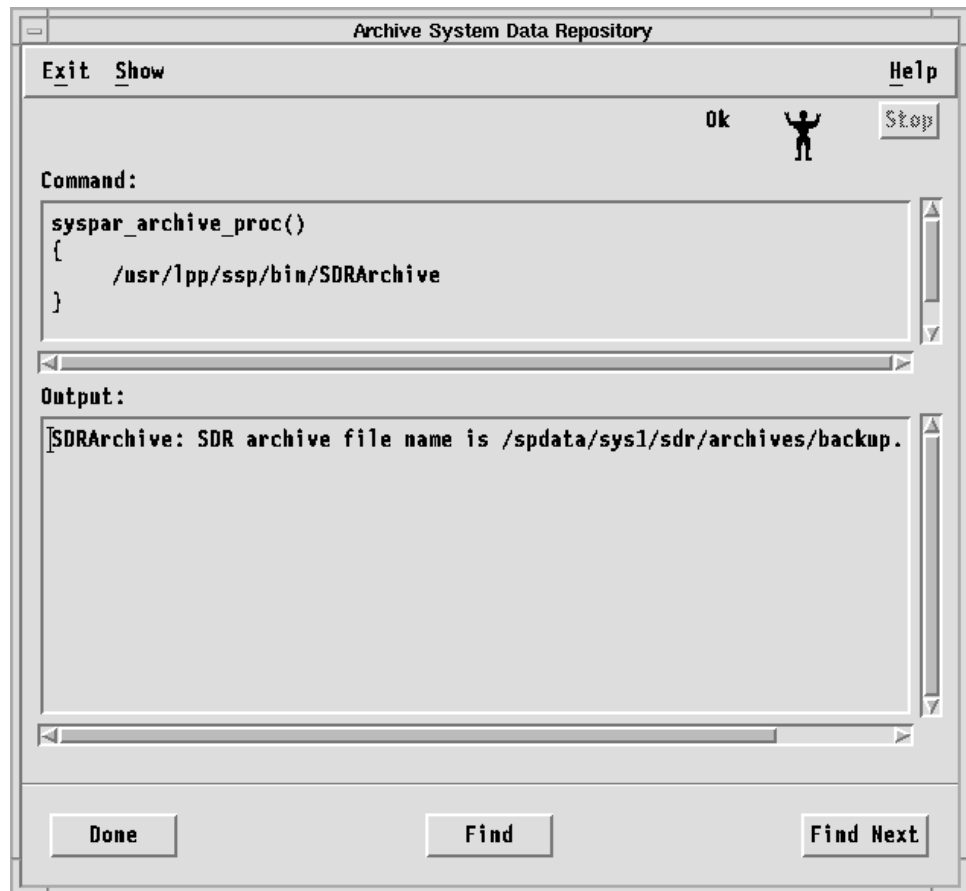


Figure 17. Archive the System Data Repository

Using the command line

Enter:

SDRArchive

Note: When you archive the SDR from either SMIT or by using the **SDRArchive** command, the archive produced is in **tar** format. Issuing **tar -x** for this archive does not restore the system partition configuration. To restore the system partition configuration, always use the SMIT Restore System Partition Configuration option.

Step 4: Select a System Partition Configuration to Apply

Refer to the chapter “Planning for System Partitions” in *IBM RS/6000 SP: Planning Volume 2, Control Workstation and Software Environment* for detailed information on partition schemes for various size SP systems.

To select a system partition configuration to apply:

1. Select a configuration from a list of supported system partition configurations for your SP system
2. Display layout information for the selected system partition configuration
3. Select a system partition layout

Select a System Partition Configuration

To select a supported system partition configuration for your SP system,

Using SMIT: At the System Partition Configuration menu:

SELECT Select System Partition Configuration

- A list of system partition configurations appears. (If you have created layouts using the System Partitioning Aid, such layouts will also be listed on the SMIT panel.)

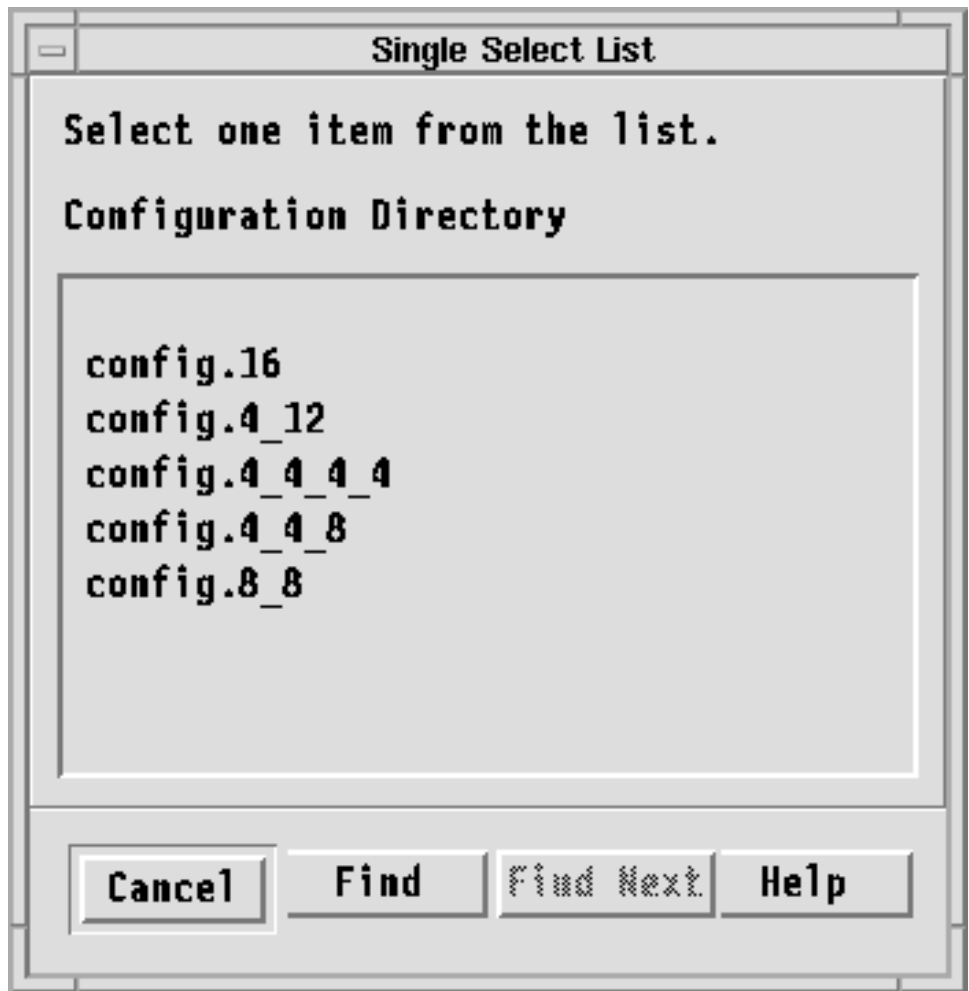


Figure 18. Select System Partition Configuration

SELECT A system partition configuration

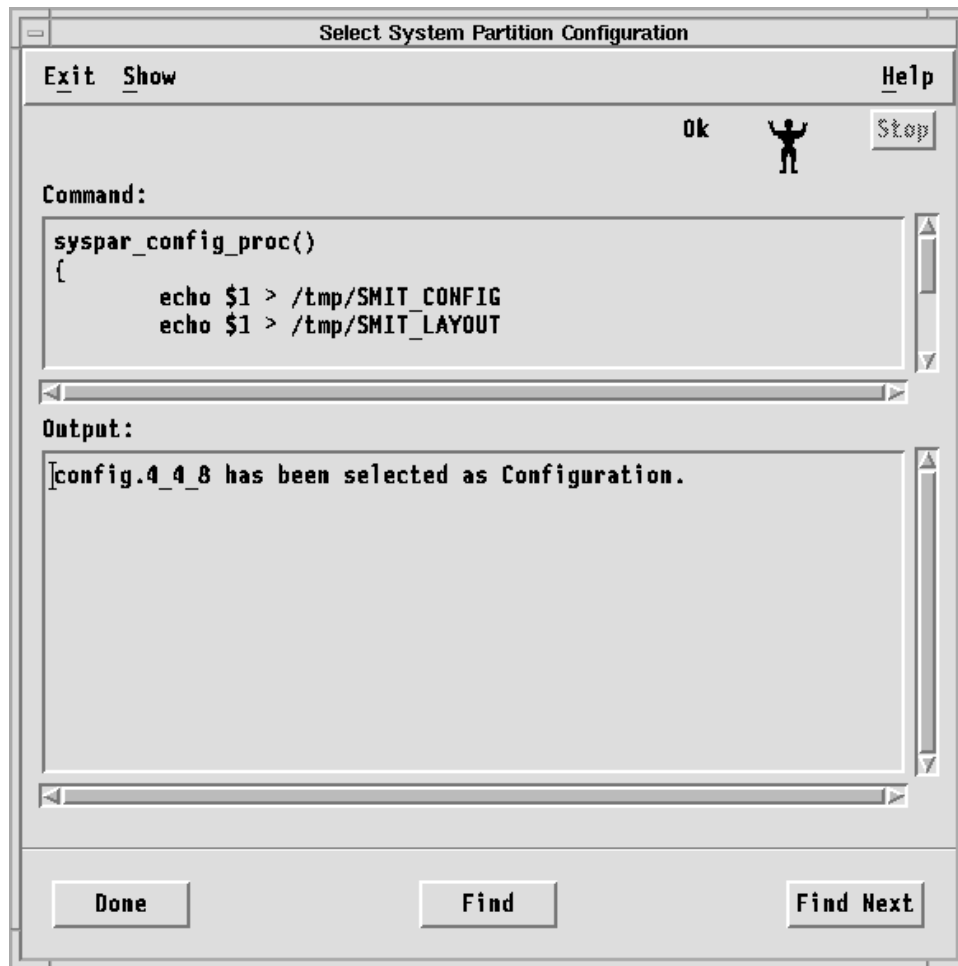


Figure 19. Select System Partition Configuration

Using the command line: To display all supported system partition configurations for your SP system, enter:

```
spdisplay_config
```

Display Information for the Selected Configuration

To display information about the supported layouts for the system configuration you have selected,

Using SMIT: At the System Partition Configuration menu:

SELECT Display Information for Given Configuration or Layout

- The Display Information for Given Configuration or Layout menu appears.

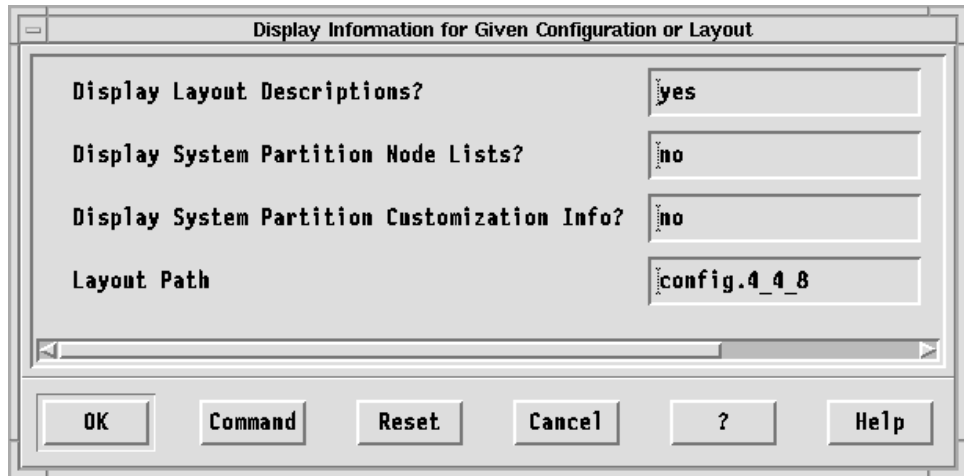


Figure 20. Display Information for Given Configuration or Layout menu

SELECT OK

- System partition layout information for the chosen system partition configuration appears.

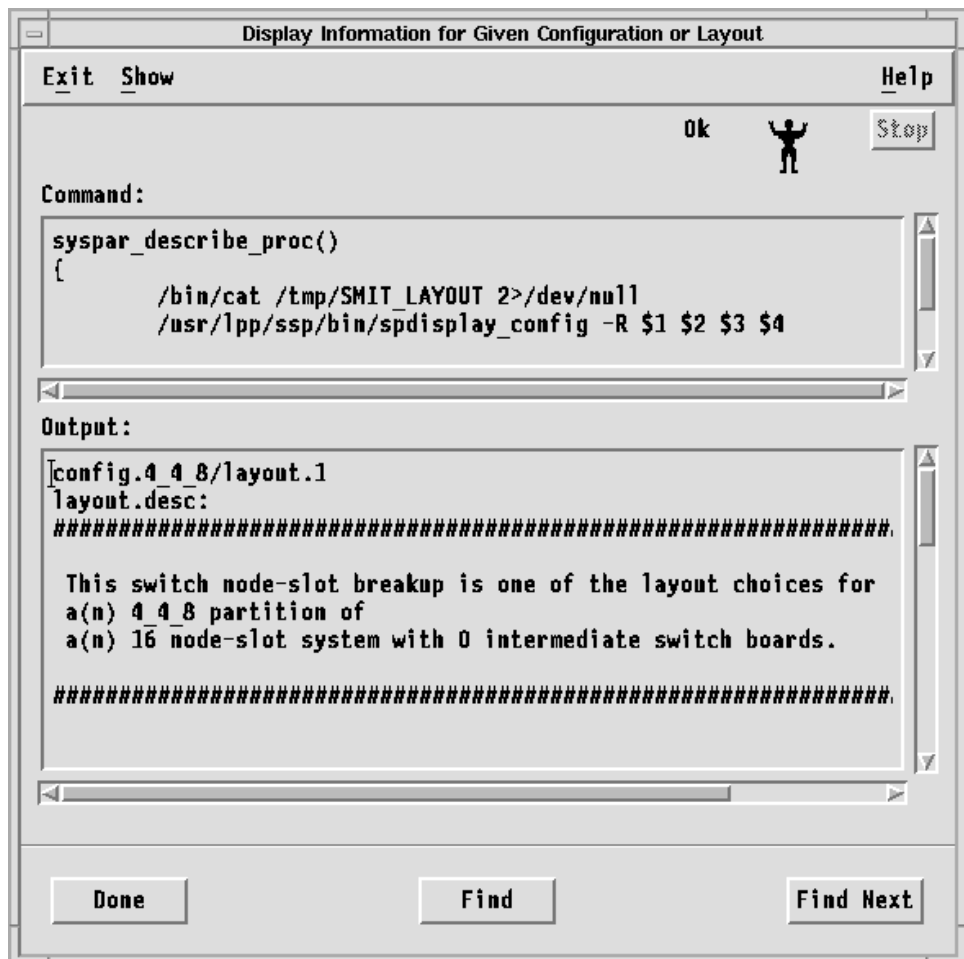


Figure 21. Display Information for the Selected Configuration

(This example shows just the first screen of the layout information displayed for our selected 4-4-8 system partition configuration.)

Using the command line: To display information for the supported layouts for the 4-4-8 system partition configuration we've chosen, enter:

```
spdisplay_config -R -d config.4_4_8
```

Select a System Partition Layout

Based on the layout information you have displayed for the selected system partition configuration, select a layout,

Using SMIT: At the System Partition Configuration menu:

SELECT Select System Partition Layout

- A list of system partition layouts for the chosen system partition configuration appears. (If you have created layouts using the System Partitioning Aid, such layouts will also be listed on the SMIT panel.)



Figure 22. Select a System Partition Layout

SELECT A system partition layout

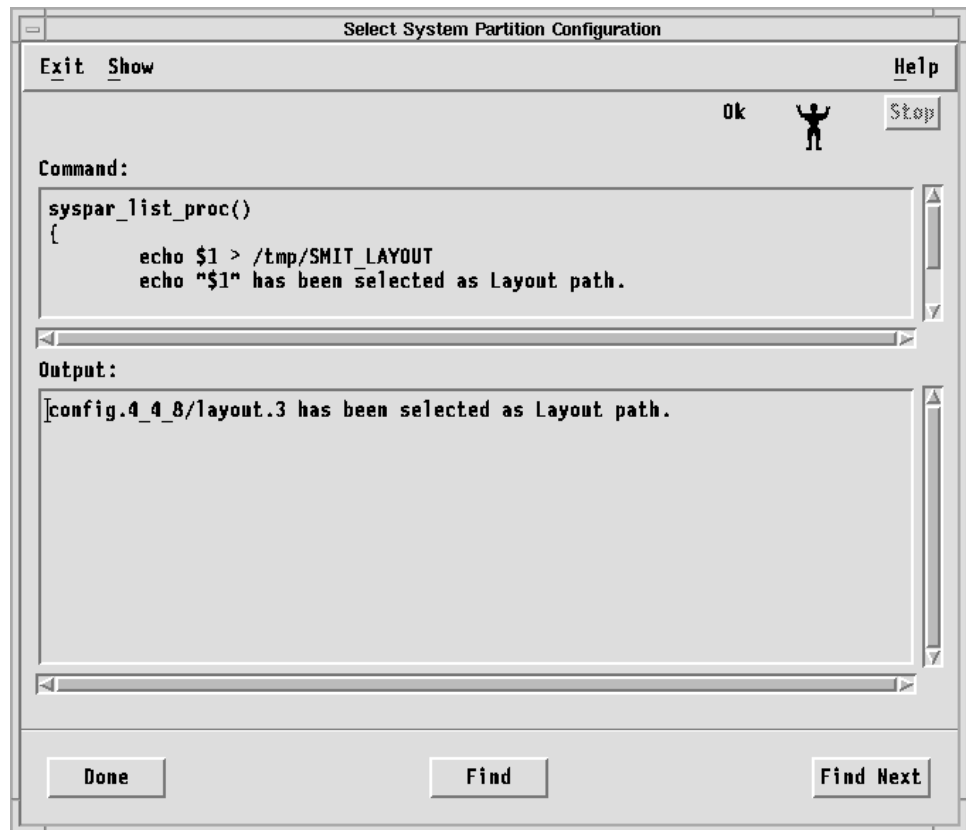


Figure 23. Select a System Partition Layout

In this example, we have chosen layout 3 for our selected 4-4-8 system partition configuration.

Step 5: Customize a System Partition

You customize a system partition by specifying information for:

- System partition name or IP address
- PSSP code level
- Partition default installation image (optional)
- Primary node (optional)

The customization information is retained in the **custom** file for the partition. Because the customization information is retained in the **custom** file, the customization step would not have to be repeated when doing a reconfiguration if the same system partition is being used and the customization information does not need to be changed. It is imperative that all new IP addresses and host names being used will be resolvable on your SP system.

Using SMIT

At the System Partition Configuration menu:

SELECT Enter Customization Information for a Selected System Partition

- A list of the system partitions for the chosen configuration layout appears.

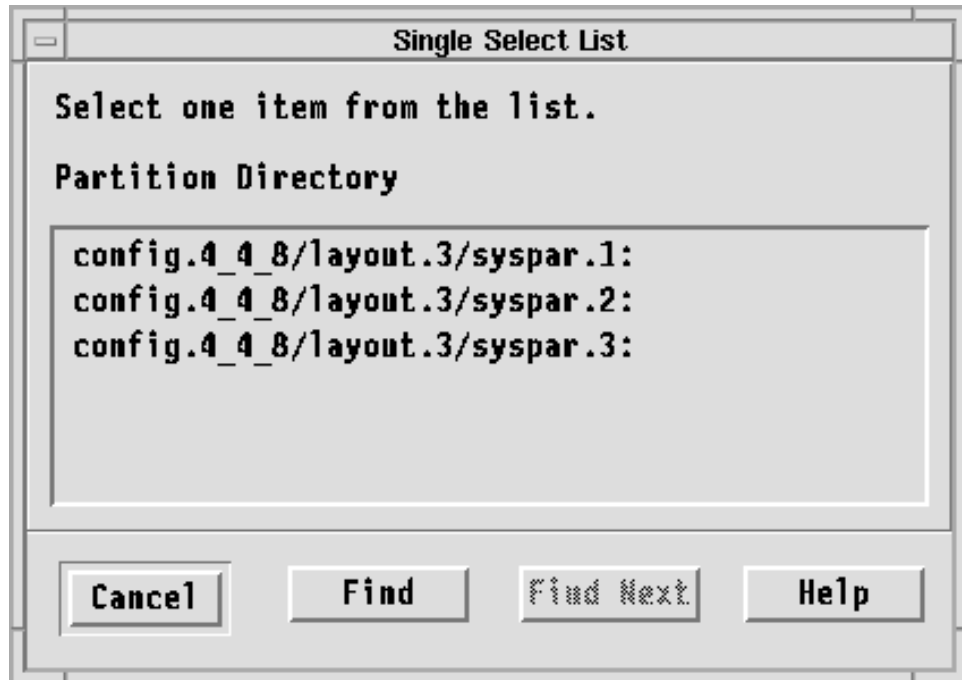


Figure 24. Customize a System Partition

SELECT A system partition and enter customization arguments

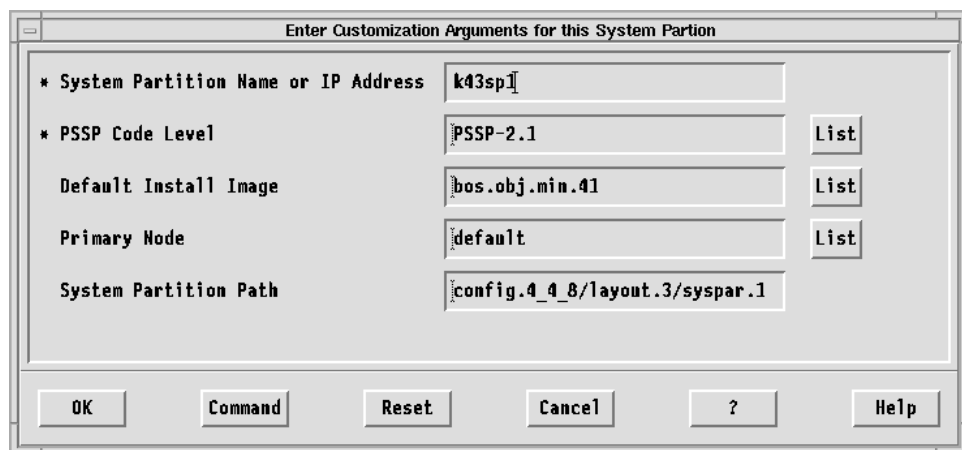


Figure 25. Customize a System Partition

Note: If you have already customized this system partition this panel displays those values. Change the values as desired.

The customization arguments and their meanings are:

System Partition Name or IP Address

The name of the system partition is either the real host name (in the case of the default system partition) or the alias host name of the control workstation. The IP address of the system partition is either the real IP address (in the case of the default system partition) or the alias IP address of the control workstation.

PSSP Code Level

The IBM Parallel System Support Programs for AIX (PSSP) version and release: **PSSP-2.2**, **PSSP-2.3**, **PSSP-2.4**, **PSSP-3.1**, or **PSSP-3.1.1**. For system partitions that contain mixed levels of PSSP, this should be set to the minimum, or earliest, level installed in this system partition.

Default Install Image

The name of the **mksysb** image that is used to install a node in this system partition (if the node itself doesn't have an installation image value). For system partitions that contain mixed levels of PSSP, this default install image should be associated with the PSSP code level specified above.

Primary Node

The node on the switch that initializes, recovers, provides diagnosis, and performs other operations to the switch network. The default is the first node in the **nodelist** file.

Backup Primary Node

Available only on SP Switch systems, the node designated as the oncoming switch primary node.

System Partition Path

The full path name for the configuration, layout, and system partition name.

Using the command line

To list the system partitions for our chosen 4-4-8 system partition configuration, layout 3, enter:

```
spdisplay_config config.4_4_8/layout.3
```

To enter customization information (as shown in our example SMIT panel) for system partition 1 of our chosen 4-4-8 system partition configuration, layout 3, enter:

```
spcustomize_syspar -n k43sp1 -l PSSP-2.2 -d bos.obj.min.41\  
-e default config.4_4_8/layout.3/syspar.1
```

Repeat this step, using the appropriate values, for each system partition in the configuration.

Step 6: Shut Down All Nodes in Changing Partitions

When you are partitioning or repartitioning the system, system partitions are considered the same if all of the following are identical:

- Node list files
- Topology files
- IBM Parallel System Support Programs for AIX code level
- Default install image
- IP address
- Partition name

If any of the above items are not identical, the system partition is being changed and nodes in that partition should be shut down before applying the system partition configuration. If your system contains an SP Switch, you must invoke **Eunpartition** in each system partition that is being changed. See the **Eunpartition** command in the *PSSP: Command and Technical Reference* for more information.

Note: Modifying only the primary node will not result in a changed system partition. If you want to change the primary node, use **Eprimary** followed by **Estart**. See the **Eprimary** command in the *PSSP: Command and Technical Reference* for more information.

You can use the Verify option on the Apply System Partition Configuration SMIT panel to determine what nodes are changing. The output for the Verify option will list changing system partitions as well as any nodes in those system partitions that are not shut down.

Using SMIT

At the System Partition Configuration menu:

SELECT Apply System Partition Configuration

- The previously selected system partition configuration and layout appears.

SELECT The system partition configuration and layout

- The Apply System Partition Configuration menu appears.

SELECT Verify only from the list of System Partition Apply Options

SELECT OK

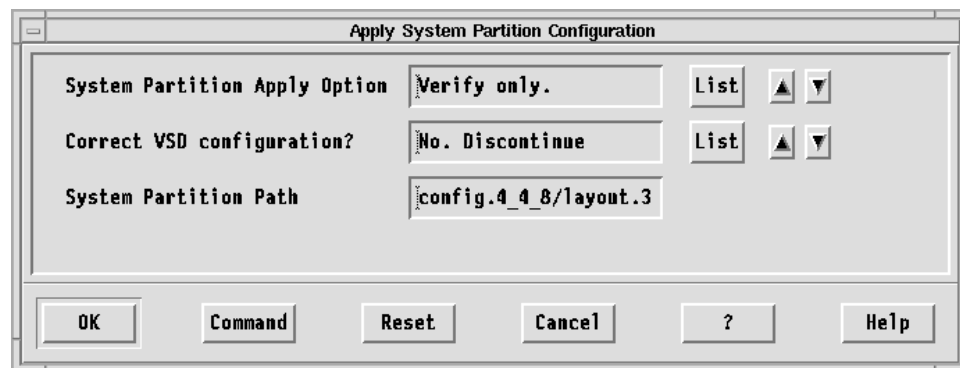


Figure 26. Apply System Partition Configuration menu

The Apply System Partition Configuration arguments and their meanings are:

System Partition Apply Option

The options are provided to either only verify the system partition configuration or to actually apply the system partition configuration.

Correct VSD configuration?

The option to correct or discontinue when nonfatal errors are encountered in the virtual shared disk subsystem in the application of the specified system partition configuration.

System Partition Path

The full path name for the configuration, layout, and system partition name.

Using the command line

Enter:

```
spapply_config -v config.4_4_8/layout.3
```

Step 7: Run the `setup_server` Command on the Control Workstation

If you have added an alias for the control workstation for the purpose of defining a name for a new system partition (see “Before You Define System Partitions” on page 223), you must run the `setup_server` command on the control workstation to properly define the new `rcmd` principal associated with the new host name alias.

Step 8: Apply a System Partition Configuration

When you apply or commit a system configuration, the following occurs:

- The existence of all required files (**nodelist**, **topology**, **custom**) for each system partition in the specified layout directory is verified. In addition, the contents of the **custom** file is checked for completeness.
- Any new system partitions are created on the control workstation.
- The system partition object for each system partition is created in the SDR.
- Affected node objects in the SDR (**Node**, **Adapter**, **host_responds**, **switch_responds**) and affected extension node objects in the SDR (**DependentNode** and **DependentAdapter**) are moved to the newly created partition.
- New **sdrd** daemons are created and started.
- New system partition-sensitive subsystems are created and started via the **syspar_ctrl** command.
- The **Syspar_map** information in the SDR is set for each node slot in the partition.
- The **verparvsd** command is invoked to verify the virtual shared disk data for system partitioning. The command extracts all virtual shared disk data from nodes involved in the system partitioning and writes SDR commands to update the virtual shared disk data to an output file. The output file reloads the virtual shared disk SDR data into the correct new partitions. (Note that the **verparvsd** command can also be used prior to applying a system partition configuration to test the desired layout for the existing virtual shared disk configuration defined in the SDR. See the *PSSP: Command and Technical Reference* for more details.)
- Old unused system partitions on the control workstation, along with their **sdrd** daemons, are deleted and old system partition-sensitive subsystems are deleted.
- Updated SDR information is checked for accuracy against the information in the supplied system partition configuration files.

To apply a system partition configuration,

Using SMIT

At the System Partition Configuration menu:

SELECT Apply System Partition Configuration

- The previously selected system partition configuration and layout appears

SELECT The system partition configuration and layout

- The Apply System Partition Configuration menu appears.

SELECT Apply this config.

SELECT OK

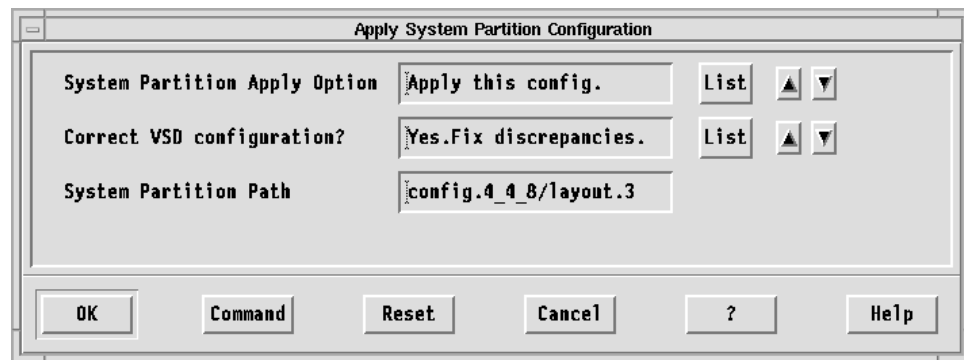


Figure 27. Apply System Partition Configuration SMIT panel

Using the command line

To apply our chosen 4-4-8 system partition configuration, layout 3, enter:

```
spapply_config -F config.4_4_8/layout.3
```

Important

If a failure occurs at this step, your system is put into an inconsistent state. Use the **sprestore_config** command or Restore System Partition Configuration SMIT menu option to restore the backed-up SDR and appropriate system partition-sensitive subsystems.

Note: Due to system partitioning changes, your SP_NAME environment variable may no longer be set to a valid system partition name. To get a list of valid system partition names, enter the **splst_syspars -n** command. Then verify that your SP_NAME environment variable is either unset or set to one of the partition names in the list.

Step 9: Configure Authentication Methods for all New Partitions

You will need to create all necessary authorization files and enable selected authentication methods for each new partition created. To create the authorization files for each partition do the following:

Using SMIT

At the RS/6000 SP Security menu

SELECT Select Authorization Methods

SELECT System Partition name

- Move cursor to the desired system partition and press Enter

SELECT Authorization Methods

- Select one or more authorization methods using PF7, then Press Enter. (k4 is required)

PRESS Enter

Using the command line

Enter:

```
spsetauth -d -p <partition_name> k4 [std]
```

To enable selected authentication methods for each partition do the following:

Using SMIT

At the RS/6000 SP Security menu

SELECT Enable Authentication Methods

SELECT Enable on Control Workstation Only

SELECT Force change on nodes

SELECT System Partition names

- Move cursor to the desired system partition and press Enter

SELECT Authentication Methods

- Select one or more authentication methods using PF7, then press Enter (k4 is required)

PRESS Enter

Using the command line

Enter:

```
chauthpar [-c] [-f] -p <partition_name> [k5] k4 [std]
```

Perform the previous step for each new system partition. When the nodes are rebooted in the next step, they will pick up the new or changed methods.

Step 10: Reboot All Nodes in Changed Partitions

Reboot any nodes that were shut down in “Step 6: Shut Down All Nodes in Changing Partitions” on page 235.

Displaying System Partition Configuration Information

You can display the following information for the system partitions in your current configuration:

- A list of the system partitions in your current configuration
- For each system partition in your current configuration:

- System partition name
- IP address
- Install image
- Layout directory name
- IBM Parallel System Support Programs for AIX code version
- Authorization Methods set for root access to AIX authenticated remote commands
- Authentication Methods enabled

Using SMIT

- TYPE** `smit`
- The System Management menu appears.
- SELECT** RS/6000 SP System Management
- The RS/6000 SP System Management menu appears.
- SELECT** RS/6000 SP Configuration Database Management
- The RS/6000 SP Configuration Database Management menu appears.
- SELECT** List Database Information
- The List Database Information menu appears.
- SELECT** List System Partition Database Information
- The list of system partitions along with information for each of the system partitions appear.

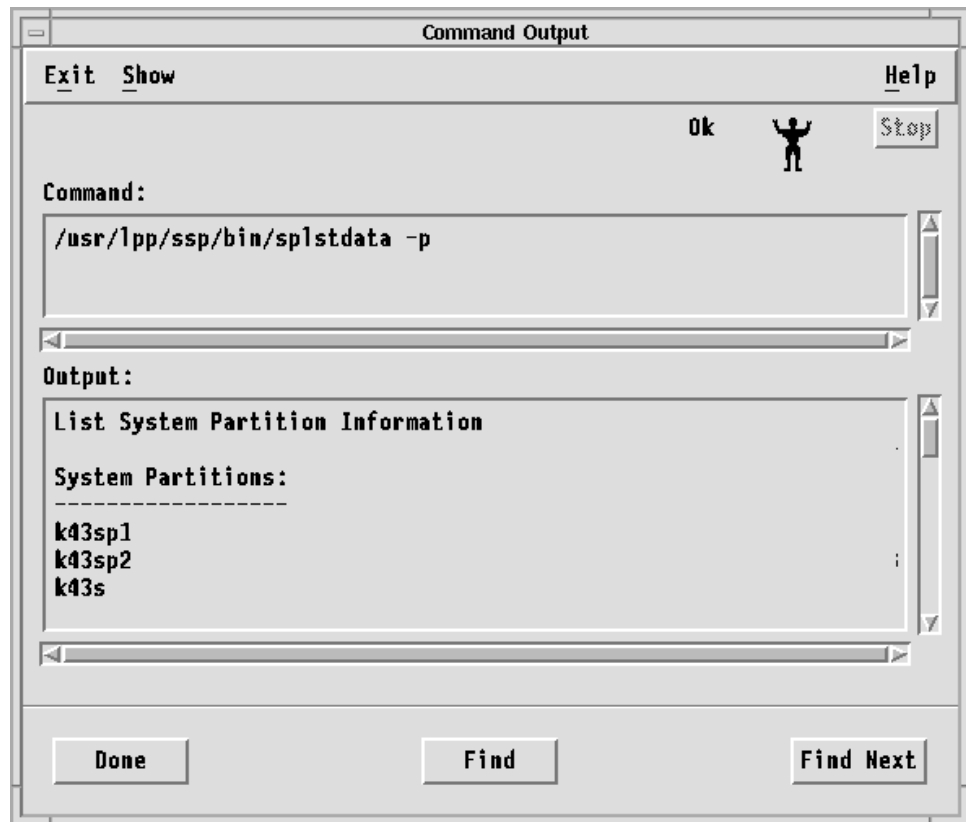


Figure 28. List System Partition Database Information Output

(This example shows just the first screen of the system partition configuration information for the system partition configuration we have applied.)

The fastpath invocation for the List System Partition Configuration Information menu is:

```
smit list_syspar
```

Using the command line

To display information for the system partitions in your current configuration, enter:

```
splstdata -p
```

Verifying System Partition Configuration

You can verify that the system partition information in the SDR is logically consistent.

Using SMIT

TYPE smit

- The System Management menu appears.

SELECT RS/6000 SP System Management

- The RS/6000 SP System Management menu appears.

SELECT RS/6000 SP Installation/Configuration Verification

- The RS/6000 SP Installation/Configuration Verification menu appears.

SELECT System Partition Configuration

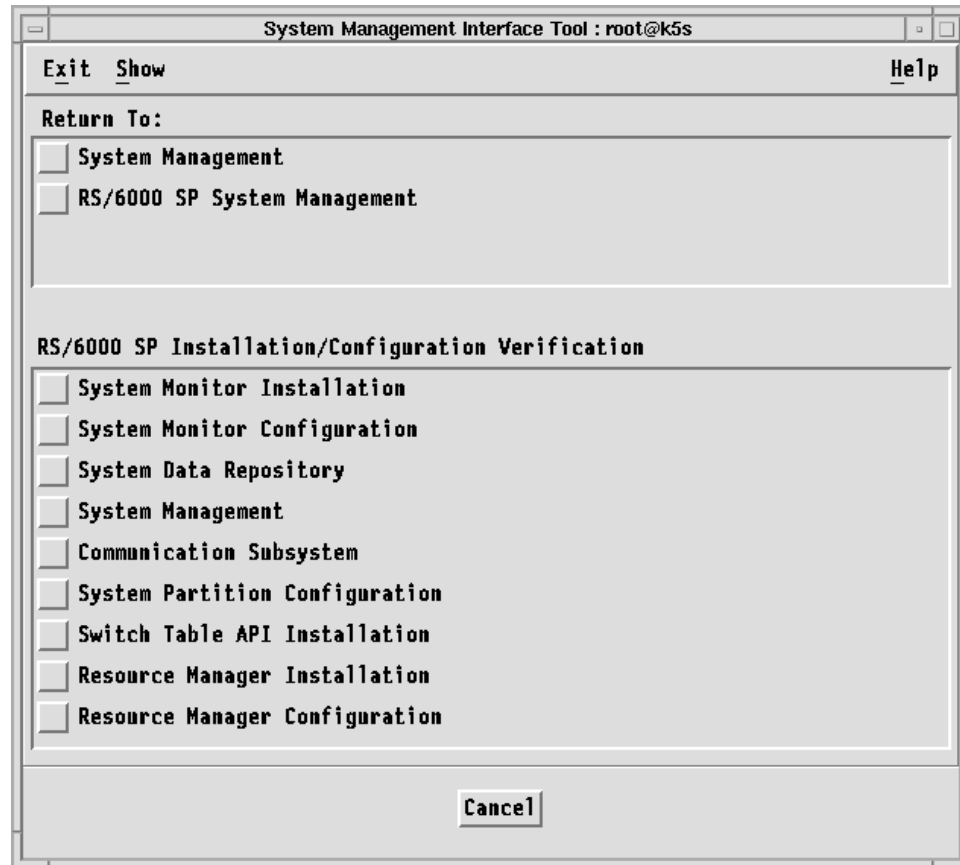


Figure 29. Verify System Partition Configuration SMIT Panel

The fastpath invocation for the Verify System Partition Configuration SMIT menu is:

```
smit SP_verify
```

Using the command line

Enter:

```
spverify_config
```

If the **spverify_config** command runs successfully, the SMIT status indicator will display **OK**.

Managing System Partition-Sensitive Subsystems Using `syspar_ctrl`

Some of the subsystems that run in the SP environment operate within the domain of a system partition, rather than the domain of the system as a whole. These subsystems are said to be **system partition-sensitive**.

To help you manage these subsystems, PSSP includes the Syspar Controller, which operates through the **syspar_ctrl** command. With this command you can

add, start, stop, refresh, and delete system partition-sensitive subsystems, as well as control other aspects of their operation, such as tracing.

The **syspar_ctrl** command provides a single interface to the control scripts for the system partition-sensitive subsystems, which perform the real work of configuring and operating the subsystems.

Listing the System Partition-Sensitive Subsystems

To list the system partition-sensitive subsystems that are managed by the Syspar Controller and their control scripts, enter:

```
syspar_ctrl -E
```

In response, the system displays information similar to the following:

Syspar Controller managed subsystems and control scripts:

hats	/usr/sbin/rsct/bin/hatsctrl
hags	/usr/sbin/rsct/bin/hagsctrl
haem	/usr/sbin/rsct/bin/haemctrl
hr	/usr/lpp/ssp/bin/hrctrl
pman	/usr/lpp/ssp/bin/pmanctrl
emon	/usr/lpp/ssp/bin/emonctrl
sp_configd	/usr/lpp/ssp/bin/sp_configdctrl
emcond	/usr/lpp/ssp/bin/emconditionctrl

Chapter 16. Managing Node Groups

Node grouping is a persistently saved naming facility for sets of nodes: nodes can be grouped together so that they can be acted on as an individual entity. Many system management commands can accept a node group as an argument, causing the action to be performed on each node in the node group. Node groups can be created and managed through a command line interface, SMIT panels, and the SP Hardware Perspective.

This chapter provides information on understanding and working with node groups, as well as using node group commands.

Understanding Node Groups

A node group is a set of nodes whose members may be individual nodes and other node groups. Individual nodes that belong to a node group are *member nodes*; node groups that belong to a node group are *member node groups*. Node groups containing only nodes are *flat node groups*; node groups containing node groups are *hierarchical node groups*. Any node group can be resolved into a flat version that contains only valid member nodes.

Dependent nodes cannot be part of a node group but SP-attached servers can be part of a node group just like any other standard SP node.

Commands, except those that directly manage the node groups, issued against a node group have the same effect as issuing the commands against the set of nodes that represent the flat version of the node group.

Node Group Information in the SDR

Node groups are stored persistently in the SDR. A node group in the SDR is made up of two components: a name and a definition list. A node group name must start with an alphabetic character (A-Z, a-z), but can then contain a sequence of alphanumeric characters (A-Z, a-z, 0-9), a period (.) or an underbar (_). (Thus named, node groups can be easily differentiated from node objects, whose names are a string of numeric characters representing the node number.) A node group name may not contain spaces. The definition list of a node group is made up of a list of nodes and node groups that belong to the node group directly, (not through member node groups).

Node Group Modes

Some node groups exist within a system partition (*partition-bound* node groups or *local* node groups) and some node groups ignore system partition boundaries (*global* node groups). Partition-bound or local node groups can be accessed only within the context of the system partition they belong to. Global node groups are not bound to any system partition, but can be acted on only by commands that accept the **-G** flag or when using the SP Perspectives GUI with a Global view of the SP system. Partition-bound and global refer to the *mode* of the node group.

Partition-bound node groups (local node groups) are stored in the partitioned SDR class **NodeGroup**. Global node groups are stored in the system SDR class **SysNodeGroup**. Valid member nodes of a partition-bound node group must be

node numbers that are defined in that particular system partition. Valid member nodes of a global node group are node numbers that are defined in that particular SP system (the union of all nodes in each system partition). Any node number or node group can be added to the definition list of a node group, but only valid member nodes or valid member node groups will be used when resolving (flattening) the node group.

Note: The global name space and the partition name spaces are mutually exclusive for node groups.

Most SP system end users work within system partitions and, therefore, will use partition-bound node groups.

System Administrators, however, will often find it necessary, and advantageous, to work with global node groups. The use of global node groups allows a System Administrator to perform certain tasks, such as installing a daemon on specific nodes or powering off and reconfiguring all nodes on which a given application is running, without regard to system partitions.

Partition-bound node groups and global node groups are different, but they both function to provide a naming facility for sets of nodes.

Considerations for Creating and Resolving Node Groups

When creating a node group or adding to an existing node group's definition list, you can use the command line interface or SMIT panels to add any node number or node group. The member node group does not need to have been defined at the time it is added to another node group's definition list. The member node does not have to be a valid node number for a system partition or even the SP system at the time it is added to another node group's definition list. In this way, you can create node groups to accommodate future growth or changes, such as adding nodes or repartitioning your SP system.

Only users that have the appropriate authority to modify objects in the SDR can create, modify or delete node groups. Any user that has the appropriate authority to read objects from the SDR can reference existing node groups. Any user can modify or reference node groups created by other users, provided they have the appropriate SDR authority. For more information on the authority needed to access the SDR, refer to "Authorization" on page 479.

When a node group is resolved into its flat version, only valid member nodes will appear in the flat version and only valid member node groups will be resolved recursively to produce the flat version. If a node group is a partition-bound node group, when it is flattened all of its member nodes and member node groups must be defined in the same system partition; if they are not, they will not be used to produce the flat version of this node group. If the node group is a global node group, when it is flattened all of its member nodes must be node numbers for the system and member node groups must also be defined as global node groups. Member nodes or member node groups that do not correspond to entities in persistent storage will be ignored when producing a flat version of the node group. A node group's flat version can be empty.

The following is a simple example of how a partition-bound node group will be resolved into a flat list of node numbers.

```
system partition P1 contains nodes 1, 2, 5, 6, 9, 10, 13, 14
system partition P1 contains node groups A, B, C
Node Group A name = A; definition list = 1, 2, B, C, D
Node Group B name = B; definition list = 1, 2, 3, 4, 5, 6
Node Group C name = C; definition list = 13, 14
flat(A) = nodes 1, 2, 5, 6, 13, 14
flat(B) = nodes 1, 2, 5, 6
flat(C) = nodes 13, 14
```

Notes:

1. Nodes 3, 4 are not in flat(A) or flat(B) because they are not part of partition P1.
2. Node group D has no effect on the flat(A) because node group D is not defined in partition P1.

Working with Node Groups

You can use a command line interface, SMIT panels, and the SP Hardware Perspective for working with node groups. Refer to *PSSP: Command and Technical Reference* for a complete description of commands. For information on using the SP Hardware Perspective to create and manage node groups, refer to the online help.

You can use the following commands to create, modify, and delete node groups:

ngcreate	Creates and optionally populates a named node group.
ngnew	Creates new node groups in persistent storage.
ngaddto	Adds specified nodes and node groups to named node group.
ngdelfrom	Removes specified nodes and node groups from named node group.
ngdelete	Removes node groups from persistent storage.
ngclean	Removes nodes and node groups that do not currently exist from specified node groups or every node group in the current system partition.

You can use the following commands to display node group information to standard output:

ngresolve	Displays a list of nodes in the node group.
nglist	Writes a list of all persistent node groups.
ngfind	Writes a list of all node groups containing the supplied node or node group.

Using SMIT

To access SMIT panels (using the fastpath invocation) for creating, modifying, and deleting node groups:

- TYPE** **smit nodegroups**
- The Node Group Information menu appears.

SELECT Partition Node Groups OR System Node Groups

- The Partition Node Groups or System Node Groups panel appears.

To access SMIT panels (using the fastpath invocation) for examining existing node groups:

TYPE **smit list_nodegroup**

- The List Node Group Database Information menu appears.

SELECT Examine Partition Node Groups OR Examine System Node Groups

- The Examine Partition Node Groups or Examine System Node Groups panel appears.

To access SMIT panels (using the fastpath invocation) for deleting node groups:

TYPE **smit delete_nodegroup**

- The Delete Node Group Information menu appears.

SELECT Delete Partition Node Groups OR Delete System Node Groups

- The Delete Partition Node Groups or Delete System Node Groups panel appears.

Using SP Hardware Perspective

To create a node group:

1. Ensure that the Node Groups pane is displayed and is current.
2. Select Actions → Create Node Group. The Create Node Group window appears.
3. Add the name of the node group, member nodes, and other requested information.
4. Click Apply.

To modify a node group:

1. Select a node group from the Node Groups pane.
2. Select the Notebook icon from the tool bar. The notebook for that node group appears.
3. Modify the information in the notebook for that node group.

To delete a node group:

1. Select a node group from the Node Groups pane to remove.
2. Select Actions → Remove Node Group.

To view node group properties:

1. Select a node group from the Node Groups pane.
2. Select the Notebook icon from the tool bar. The notebook for that node group appears.
3. View the information in the notebook for that node group.

Using Node Groups with Commands

The following commands will accept a node group as an argument. The node group may be supplied as an operand, if this is what is most consistent with the existing syntax. Some commands use the **-N** flag to indicate a node group. Refer to *PSSP: Command and Technical Reference* for actual syntax for each command.

- **cshutdown**
- **cstartup**
- **dsh** (can be used to run a command on every node in a flattened node group)
- **Efence**
- **Eunfence**
- **hostlist**
- **pmandef**
- p-commands

The following commands all pass arguments to **hostlist**. **hostlist** accepts a node group as an argument, so these commands get node group support through **hostlist**:

- **p_cat**
- **pcp**
- **pdf**
- **pexec**
- **pexecr**
- **pfck**
- **pfind**
- **pfps**
- **pls**
- **pmv**
- **ppred**
- **pps**
- **prm**
- **spacctnd**
- **spadaptrs**
- **spbootins**
- **spdeladap**
- **spdelnode**
- **spethernt**
- **sphostnam**
- **sphrdwrad**
- **splst_versions**
- **splstdata**

Chapter 17. Managing Extension Nodes

This chapter provides the following information:

- Understanding extension nodes and extension node adapters
- Implementing extension nodes and extension node adapters
- Transferring configuration information for extension node adapters
- Adding extension nodes and extension node adapters
- Changing extension nodes and extension node adapters
- Removing extension nodes and extension node adapters
- Reconfiguring extension nodes and extension node adapters
- Resetting extension nodes and extension node adapters
- Listing extension node and extension node adapter information

For switch-related information on extension nodes, refer to “Extension Nodes and the SP Switch” on page 207.

Understanding Extension Nodes

Extension nodes are non-standard nodes that extend the SP system's capabilities or scope, but cannot be used in all the same ways as standard SP nodes. Extension nodes are attached to the SP Switch via a special **extension node adapter**.

A specific type of extension node is a **dependent node**. A dependent node depends on SP nodes for certain functions, but implements much of the switch-related protocol that standard nodes use on the SP Switch.

SMIT panels and commands are provided for adding configuration information about extension nodes to the SDR. Once this information has been entered into the SDR, a Simple Network Management Protocol (SNMP) Manager daemon that runs on the control workstation (the SP SNMP Manager) allows the transfer of extension node configuration information to the SNMP Agent running on the extension node.

Specific Implementations of Extension Nodes and Extension Node Adapters

An RS/6000 SP Switch Router can be connected to the SP Switch via the RS/6000 SP Switch Router Adapter. The RS/6000 SP Switch Router Adapter provides a high performance, 100 MB/s, full duplex interface between the SP Switch and the SP Switch Router.

When the RS/6000 SP Switch Router Adapter is installed in an SP Switch Router, it allows the switched IP router to be used as a networking gateway for the SP. The SP Switch Router may be populated with additional adapters for standard network interfaces, including Ethernet 10/100BaseT, FDDI, ATM OC3c, SONET OC3c, ATM OC12c, HIPPI and HSSI. More than one RS/6000 SP Switch Router Adapter may be installed and connected to the same SP system or system partition or to other

SP systems. When multiple RS/6000 SP Switch Router Adapters are installed and connected to more than one SP system or system partition, they may be used to provide a high bandwidth link between SP systems or system partitions and to provide the SP systems or system partitions with a shared set of interfaces to external networks.

Each RS/6000 SP Switch Router Adapter requires one available node switch port on the SP Switch that meets the criteria for valid extension node ports as described in *IBM RS/6000 SP: Planning Volume 2, Control Workstation and Software Environment*. A 10 meter SP Switch cable and a 10 meter ground strap are provided for connecting the RS/6000 SP Switch Router Adapter, located in the SP Switch Router chassis, to the SP Switch.

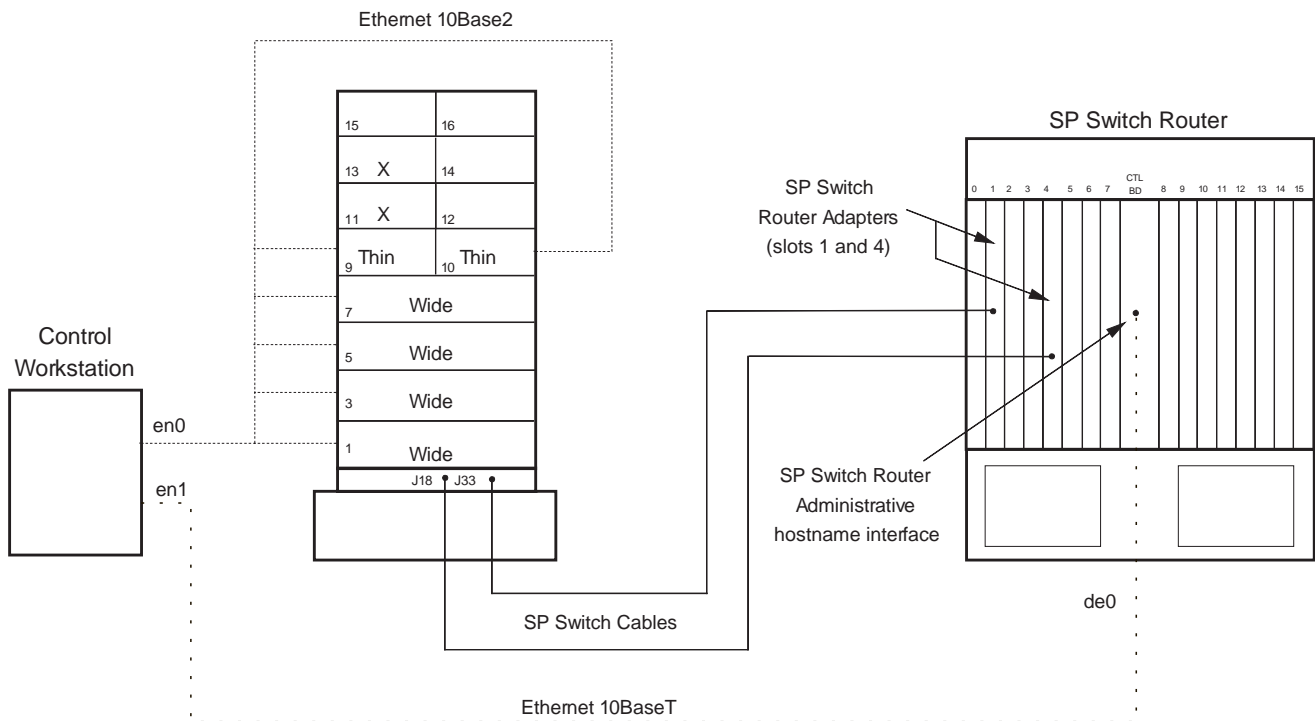


Figure 30. The RS/6000 SP Switch Router and SP Switch Router Adapter Extension Node Implementation

Extension Node Attribute Values

When you add information to the SDR for an SP Switch Router and the RS/6000 SP Switch Router Adapter (see “Adding Extension Nodes and Extension Node Adapters” on page 253), specify the following extension node attributes as described:

- Administrative Hostname** The fully-qualified host name of the administrative Ethernet network on the SP Switch Router.
- Extension Node Identifier** The 2-digit slot number on the SP Switch Router where the RS/6000 SP Switch Router Adapter is located.
- SNMP Agent Hostname** The fully-qualified host name of the administrative Ethernet network on the SP Switch Router.

For more information on configuring the SP Switch Router and RS/6000 SP Switch Router Adapter as extension nodes for the SP system, see:

- *IBM RS/6000 SP: Planning Volume 2, Control Workstation and Software Environment*
- *PSSP: Installation and Migration Guide*
- *SP Switch Router Adapter Guide*

Transferring Extension Node Configuration Information

Once you have used SMIT panels or commands to enter extension node information into the SDR, the SP SNMP Manager running on the control workstation allows the transfer of extension node configuration information to the SNMP Agent running on the extension node.

Refer to “Understanding Network Management” on page 374 for a high-level overview of Simple Network Management Protocol (SNMP).

The SP SNMP Manager daemon, **spmgrd**, runs on the control workstation and treats each extension node as a managed node, with a corresponding SNMP Agent. The SP SNMP Manager waits for trap messages from extension node SNMP Agents reporting state changes or requesting configuration information and sends configuration information to the SNMP Agents when requested. For more information on **spmgrd**, See the book *PSSP: Command and Technical Reference*.

An SNMP Agent may be responsible for one or several extension nodes attached to the SP Switch. The SNMP Agent maintains configuration information for each of its extension nodes in the SNMP Management Information Base (MIB). When an SNMP Agent needs configuration information for one of its extension nodes, it sends a trap message to the SP SNMP Manager responsible for that extension node. The SP SNMP Manager extracts the requested information from the SDR and sends it to the SNMP Agent.

See “ibmSPDepNode.my” on page 446 for a copy of the MIB for dependent nodes.

Adding Extension Nodes and Extension Node Adapters

You can add information for extension nodes and extension node adapters into the SDR by using either the SMIT or command line interface.

From SMIT

1. Access the Extension Node Database Information menu:

ENTER **smit enter_extdata**

- The Extension Node Database Information menu appears.

2. Add information to the SDR for extension nodes:

SELECT **Enter Extension Node Information**

- The Enter Extension Node Information menu appears.

- a. Enter values for the following extension node attributes:

Administrative Hostname	The fully qualified host name associated with the extension node's network interface on its system's administrative network. This name must resolve to an IP address. <i>Required.</i>
SNMP Community Name	The Simple Network Management Protocol (SNMP) community name for the authentication field in SNMP messages exchanged by the extension node's SNMP Agent and the SP SNMP Manager. This value can be 1 to 255 ASCII characters.
Extension Node Identifier	The string assigned to the extension node in its system's administrative environment. This string uniquely identifies the extension node to its system and can be 1 to 255 ASCII characters. <i>Required.</i> (See "Specific Implementations of Extension Nodes and Extension Node Adapters" on page 251 for more information.)
SNMP Agent Hostname	Host name that resolves to an IP address that is the source or destination of SNMP packets exchanged with the control workstation via the SP administrative network. <i>Required.</i>
Reconfigure the extension node? yes or no. yes	notifies the SNMP Agent on the extension node that extension node information has been added or changed. This notification causes the SNMP Agent on the extension node to request configuration information from the SP SNMP Manager on the control workstation.
Node Number	The node number that corresponds to the valid unused switch port assigned to the extension node. <i>Required.</i> See <i>IBM RS/6000 SP: Planning Volume 2, Control Workstation and Software Environment</i> for information on switch node numbering.

- b. Select **OK**.
- 3. Return to the Extension Node Database Information menu.
- 4. Add information to the SDR for extension node adapters:

SELECT Enter Extension Node Adapter Information

- The Enter Extension Node Adapter Information menu appears.

- a. Enter values for the following extension node adapter attributes:

Network Address	The IP network address for the extension node adapter.
------------------------	--

Network Netmask

The network mask for IP addresses on the attached network on which the extension node adapter resides.

Reconfigure the extension node? yes or no. yes

notifies the SNMP Agent on the extension node that extension node information has been added or changed. This notification causes the SNMP Agent on the extension node to request configuration information from the SP SNMP Manager on the control workstation.

Node Number

The node number that corresponds to the valid unused switch port assigned to the extension node. *Required. See IBM RS/6000 SP: Planning Volume 2, Control Workstation and Software Environment* for information on switch node numbering.

- b. Select **OK**.

From the Command Line

1. Use the **endefnode** command to add extension node information to the SDR.
2. Use the **endefadapter** command to add extension node adapter information to the SDR.

See the book *PSSP: Command and Technical Reference* for information on these commands.

If you are adding an extension node and extension node adapter after the SP Switch has already been started, you must run **Estart**.

Changing Extension Nodes and Extension Node Adapters

You can change information for extension nodes and extension node adapters that you've already defined in the SDR by using either the SMIT or command line interface.

From SMIT

1. Access the Extension Node Database Information menu:

ENTER smit enter_extdata

- The Extension Node Database Information menu appears.

2. Change information in the SDR for extension nodes:

SELECT Enter Extension Node Information

- The Enter Extension Node Information menu appears.

- a. Enter new values for the extension node attributes you wish to change. (Refer to "Adding Extension Nodes and Extension Node Adapters" on page 253 for a description of these values.)

Note: You cannot change the node number of the extension node. You must remove the extension node number from the SDR (by using the SMIT menus described in “Removing Extension Nodes and Extension Node Adapters” on page 256 or the **enrmnode** command) and add it back with a new node number (by using the SMIT menus described in “Adding Extension Nodes and Extension Node Adapters” on page 253 or the **endefnode** command).

b. Select **OK**.

3. Return to the Extension Node Database Information menu.

4. Change information in the SDR for extension node adapters:

SELECT Enter Extension Node Adapter Information

- The Enter Extension Node Adapter Information menu appears.

a. Enter new values for the extension node adapter attributes you wish to change. (Refer to “Adding Extension Nodes and Extension Node Adapters” on page 253 for a description of these values.)

Note: You cannot change the node number of the extension node. You must remove the extension node number from the SDR (by using the SMIT menus described in “Removing Extension Nodes and Extension Node Adapters” or the **enrmnode** command) and add it back with a new node number (by using the SMIT menus described in “Adding Extension Nodes and Extension Node Adapters” on page 253 or the **endefnode** command).

b. Select **OK**

From the Command Line

1. Use the **endefnode** command with the appropriate flags and operand to change the definition for an extension node in the SDR.
2. Use the **endefadapter** command with the appropriate flags and operand to change the definition for an extension node adapter in the SDR.

See the book *PSSP: Command and Technical Reference* for details on these commands.

You can use the **splstnodes** and **splstadapters** commands to verify that the changes you've specified have been made.

See the book *PSSP: Command and Technical Reference* for details on these commands. Also, see “Listing Extension Node and Extension Node Adapter Information” on page 259.

Removing Extension Nodes and Extension Node Adapters

You can remove information for extension nodes and extension node adapters from the SDR by using either the SMIT or command line interface.

Note: You must **reset** the extension node (using either **enadmin -a reset** or **enrmnode -r**) *before* removing the extension node adapter or extension node.

From SMIT

1. Access the Delete Database Information menu:

ENTER `smit delete_data`

- The Delete Database Information menu appears.

2. Delete extension node information:

SELECT `Delete Extension Node Information`

- The Delete Extension Node Information menu appears.

- a. Enter values to delete extension node information:

Reset the Extension Node? `yes` or `no`. `yes` notifies the SNMP Agent on the extension node to remove the node from the switch network.

Node Number The number ID assigned to the extension node.

- b. Select **OK**.

3. Return to the Delete Database Information menu.

4. Delete extension node adapter information:

SELECT `Delete Extension Node Adapter Information`

- The Delete Extension Node Adapter Information menu appears.

- a. Enter value to delete extension node adapter information:

Node Number The number ID assigned to the extension node.

- b. Select **OK**.

From the Command Line

1. Use the `enrmnode` command to remove extension node information from the SDR.
2. Use the `enrmadapter` command to remove extension node adapter information from the SDR.

See the book *PSSP: Command and Technical Reference* for information on these commands.

Reconfiguring Extension Nodes and Extension Node Adapters

Once extension node and extension node adapter information has been changed in the SDR, the extension node SNMP Agent must be notified.

From SMIT

Use the SMIT panels for adding or changing extension node information as described in “Adding Extension Nodes and Extension Node Adapters” on page 253 and “Changing Extension Nodes and Extension Node Adapters” on page 255. Specify **yes** for **Reconfigure the extension node?**

SMIT panels are also provided that invoke the `enadmin` command.

1. Access the RS/6000 Cluster Management menu:

ENTER `smit cluster_mgmt`

- The RS/6000 Cluster Management menu appears.

2. Run the **enadmin** command:

SELECT **Run enadmin Command**

- The Extension Node Management menu appears.

a. Enter values to manage the extension node:

Action to be performed on the extension node
reconfigure or **reset**.

Node Number The number ID assigned to the extension node.

b. Select **OK**.

From the Command Line

1. Use the **endefnode -r** command to reconfigure extension nodes.
2. Use the **endefadapter -r** command to reconfigure extension node adapters.
3. As an alternative to using **endefnode -r** and **endefadapter -r**, you can use the **enadmin** command to notify the SNMP Agent managing the extension node to request updated configuration information from the SP SNMP Manager (the **spmgrd** daemon) running on the control workstation.

See the book *PSSP: Command and Technical Reference* for details on these commands.

For more general information on reconfiguring extension nodes and extension node adapters, see “Transferring Extension Node Configuration Information” on page 253.

Resetting Extension Nodes

Resetting an extension node notifies the SNMP Agent managing the extension node to remove the extension node from the switch network.

Note: You must **reset** the extension node (using either **enadmin -a reset** or **enrmnode -r**) *before* removing the extension node adapter or extension node.

From SMIT

Use the SMIT panels for removing extension node information as described in “Removing Extension Nodes and Extension Node Adapters” on page 256. Specify **yes** for **Reset the extension node?**

You can also use the SMIT panels for running the **enadmin** command as described in “Reconfiguring Extension Nodes and Extension Node Adapters” on page 257. Specify **reset** for **Action to be performed on the extension node**.

From the Command Line

Use the **enrmnode -r** or **enadmin -a reset** commands to notify the SNMP Agent managing the extension node to remove the node from the switch network.

See the book *PSSP: Command and Technical Reference* for details on these commands.

Listing Extension Node and Extension Node Adapter Information

You can display information for extension nodes and extension node adapters from the SDR by using either the SMIT or command line interface or through the SP Hardware Perspective.

From SMIT

1. Access the List Database Information menu:

ENTER `smit list_data`

- The List Database Information menu appears.

2. List extension node database information:

SELECT `List Extension Node Database Information`

- The List Extension Node Database Information menu appears.

- a. Enter values to list extension node database information:

Allow Nodes Outside Current System Partition? yes or no.

Node Type

all specifies the query for both standard and dependent nodes. **standard** or **dependent** restricts the query to those types of nodes.

Node Group

The name of the node group to which the query is restricted. Only the nodes in the named node group will be queried. *Not supported for extension nodes.*

Sort Attribute

The name of a valid attribute of one of the node classes in the SDR. (See “System Data Repository Classes and Attributes” on page 486 for more information.) Listing output will be sorted by this attribute.

Query Attribute

The name and value (specified as *attribute==value*) of a valid attribute of one of the node classes in the SDR. (See “System Data Repository Classes and Attributes” on page 486 for more information.) Node information will be listed by the specified attribute. (The default is **node_number**, in which case information is listed by node number.)

Output Attribute

The attributes you want displayed.

- b. Select **OK**.

3. Return to the List Database Information menu.
4. List extension node adapter database information:

SELECT List Extension Node Adapters Database Information

- The List Extension Node Adapters Database Information menu appears.

- a. Enter values to list extension node adapter information:

Allow Adapters Outside Current System Partition? yes or no.

Node Type **all** specifies the query for both standard and dependent nodes. **standard** or **dependent** restricts the query to those types of nodes.

Query Attribute The name and value (specified as *attribute==value*) of a valid attribute of one of the node classes in the SDR. (See “System Data Repository Classes and Attributes” on page 486 for more information.) Node information will be listed by the specified attribute. (The default is **node_number**, in which case information is listed by node number.)

Output Attribute The attributes you want displayed.

- b. Select **OK**.

From the Command Line

1. Use the **splstnodes** command to list information for standard and extension nodes.
2. Use the **splstadapters** command to list information for standard node adapters and extension node adapters.

See the book *PSSP: Command and Technical Reference* for information on these commands.

From the Hardware Perspective

Note that in Perspectives, extension nodes are also referred to as IP nodes or dependent nodes. In the Hardware Perspective, you can:

- View attributes of extension nodes in the properties notebooks by selecting the node and then selecting the notebook tool bar icon.
- Monitor Switch Responds

See the online help for more details on setting up monitoring.

Monitoring and Controlling the SP System

Chapter 18. Using SP Perspectives

This chapter presents overview information for SP Perspectives covering the following topics:

- Getting started
- Understanding the SP Perspectives interface
- Accessing help for SP Perspectives
- Performing common administrative tasks with SP Perspectives that were previously performed using System Monitor GUI

Information for the tasks you can perform using the various SP Perspectives applications is available through an extensive online help system (see “Accessing Help” on page 274) and is not repeated in this chapter.

SP Perspectives is a set of applications, each of which has a graphical user interface (GUI), that enables you to perform monitoring and system management tasks for your SP system by directly manipulating icons that represent system objects.

You simply select an SP system object (a managed SP system resource such as a frame, node, or switch) by clicking on it with the mouse and then select an action to perform on that object from the menu bar or tool bar. Use this pattern of selecting an object, and then selecting an action to accomplish numerous system management tasks with SP Perspectives. Many actions are common to each Perspective such as: view and modify properties, open TTY, and run command. Table 12 lists high-level tasks, some common and some unique, for the Perspectives. You can get help on each of these tasks in the online help which can be started from the Launch Pad.

<i>Table 12 (Page 1 of 2). Perspective High-Level Tasks</i>	
This Perspective:	Enables you to perform these high-level tasks:
Hardware Perspective	<ul style="list-style-type: none"> • Select a hardware object • Set the current system partition • View hardware attributes • Monitor hardware objects • Control system partitions • Control nodes • Control node groups • Control frames and switches • Control SP Expansion I/O Units
Event Perspective	<ul style="list-style-type: none"> • Create an event definition or condition • View or modify an event definition or condition • Register or unregister event definitions or conditions • Set notification of events or conditions • Delete event definitions or conditions

<i>Table 12 (Page 2 of 2). Perspective High-Level Tasks</i>	
VSD Perspective	<ul style="list-style-type: none"> • Create, or Designate as IBM VSD Node and Define, VSDs • Create or Define IBM HSDs • Control IBM RVSD Subsystem • Configure IBM VSDs and IBM HSDs • Change IBM VSDs State • Unconfigure IBM VSDs and IBM HSDs • Undefine IBM VSDs • Remove IBM VSDs • Undefine IBM HSDs • Remove IBM HSDs • Remove IBM VSD Node Designation • Display IBM VSD Diagnostics • Change Owner and Group of IBM VSDs
System Partitioning Aid Perspective	<ul style="list-style-type: none"> • Define a new system partition • Add nodes to a system partition • Verify the system partition configuration • Generate and save a new system partition • Delete a system partition
Performance Monitoring Perspective	<ul style="list-style-type: none"> • View your system's monitoring hierarchy • Display performance data • Configure your system's hierarchy • Start and stop performance data collection • Start and stop performance data archiving

SP Perspectives also provides access to system management tools such as SMIT panels. You can also create your own applications and make them accessible from the SP Perspectives Launch Pad.

SPMON GUI Equivalency: If you previously used the System Monitor GUI to perform hardware administrative tasks, you now can use the Hardware Perspective to perform the equivalent functions. See “Performing Common Administration Tasks with SP Perspectives Previously Performed Using System Monitor GUI” on page 275 for more information.

Getting Started

This section discusses how to authorize users for SP Perspectives and how to start SP Perspectives.

Authorizing Users for SP Perspectives

Special authorization is not required to use the Perspective applications available from the Launch Pad. However, certain functions available within the Perspectives may require special authorization. Table 13 lists each Perspective and the authorization required, if any, to access and use the Perspective.

Table 13 (Page 1 of 2). Authorization required for each Perspective

Perspective:	Authorization Required:
Hardware Perspective	<p>To control the SP system hardware or open a serial link from the Hardware Perspective, an SP Perspectives user needs the appropriate hardmon ACLs (access control lists) set up on the control workstation in /spdata/sys1/spmon/hmacls (see the step on initializing authentication services in <i>PSSP: Installation and Migration Guide</i>). Then the user must have a Kerberos ticket with the hardmon authority that is necessary to control the SP system hardware.</p> <p>There are five commands that root can issue on the control workstation to which a non-root user can be given access:</p> <ul style="list-style-type: none"> • cshutdown • cstartup • Efence • Eunfence • Estart <p>These commands must be run by root on the control workstation unless sysctl Access Control Lists (ACLs) have been set up on the control workstation for the principal that the SP Perspectives user is using. Also, in order to be able to issue these commands through sysctl, the principal must have hardmon authority to the frames for which the command is issued. If you have POWER3 SMP high nodes with SP Expansion I/O Units in another frame, you need authorization for that frame also. To set up the sysctl ACL for these commands, edit the file /etc/sysctl.rootcmds.acl and add the principal that you want to be authorized to issue these commands.</p> <p>To create, modify, or delete node groups from the Hardware Perspective, you must be root on the control workstation.</p>
Event Perspective	<p>Special authorization is not required to access this Perspective. However, to use the Problem Management subsystem from SP Perspectives (available under the Event Perspective), the administrator or root user needs to add your Kerberos principal to the /etc/sysctl.pman.acl file on the control workstation and nodes. The administrator or root user must add the principal to the file on the control workstation and copy it to the nodes by issuing:</p> <pre>pcp -a /etc/sysctl.pman.acl /etc</pre> <p>If you wish, you can use another way to distribute the ACL file, such as file collections. You may also wish to be more selective about what nodes will be authorized for Problem Management. For more information on security in Problem Management, refer to "Security Considerations" on page 365.</p>

Table 13 (Page 2 of 2). Authorization required for each Perspective

<p>IBM Virtual Shared Disk Perspective</p>	<p>You may receive one of the following messages if you do not have the proper authority to use this Perspective:</p> <ul style="list-style-type: none"> • You do not have update authority to the SDR. Actions that require write access to the SDR, such as creating and removing IBM VSDs and IBM HSDs, will not be available during this IBM VSD Perspective session. To perform actions that require write access to the SDR, exit this session, log on the control workstation as a root user and restart the IBM VSD Perspective. • An error occurred while attempting to retrieve your Kerberos principal. Many of the Actions and Properties notebook pages will not be available during this IBM VSD Perspective session. Possible reasons that your Kerberos principal cannot be retrieved are: <ul style="list-style-type: none"> – You do not have a Kerberos principal; establish one properly. – The sysctld daemon used to retrieve Kerberos principals is not running; start up the daemon by running the sysctld command daemon from the command line. – The sysctl command used to retrieve the Kerberos principal is missing; install the command properly. <p>For more information, see the online help for the section on "Security Considerations for the IBM VSD Perspective" . To perform actions that require Kerberos credentials, exit this session, verify your Kerberos principal, and restart the IBM VSD Perspective.</p> • You do not have proper authority to the IBM VSD subsystem. Many of the Actions and Properties notebook pages will not be available during this IBM VSD Perspective session. To perform actions that require IBM VSD subsystem authority, exit this session, update your access in the /etc/sysctl.vsd.acl file, and restart the IBM VSD Perspective. • You need the same hmacls as the Hardware Perspective requires to open a serial link or perform other hardware control functions available on the IBM VSD Perspective. <p>For more information, refer to the <i>Managing Shared Disks document</i>.</p>
<p>System Partitioning Aid Perspective</p>	<p>To save a configuration file to the /spdata/sys1/syspar_configs directory, you must be root on the control workstation.</p>
<p>Performance Monitor Perspective</p>	<p>To use this Perspective as a non-root user, the user must be in the perfmon group. Follow these steps:</p> <ol style="list-style-type: none"> 1. As root, use smitty users and add the perfmon group to the non-root user id 2. Exit from the root id 3. From the non-root id, issue: newgrp perfmon 4. Launch the Perspective by issuing: spperfmon & 5. Refer to <i>IBM Parallel System Support Programs for AIX Performance Monitoring Guide and Reference</i> for more information on the perfmon user group.

Starting SP Perspectives and an SP Perspective Application

To start SP Perspectives running as a background process:

1. Export your display
2. Enter **perspectives &** (presuming that you have placed **/usr/lpp/ssp/bin** in your PATH) at the command line to bring up the Launch Pad. System management applications provided by SP Perspectives appear as icons on the Launch Pad. For more information on the Launch Pad, see “The Launch Pad” on page 268.
3. To start an SP Perspective application, double-click on the icon for the application you want to run. Initially an hourglass icon appears, followed by the SP Perspectives splash screen, to let you know processing is occurring. Then, the application window (also known as the “Perspective”) will open.

Alternatively:

1. Export your display
2. Enter **perspectives &** (presuming that you have placed **/usr/lpp/ssp/bin** in your PATH) at the command line to bring up the Launch Pad.
3. Click on the icon for the application you want to run.
4. Click on the **Actions** menu bar item.
5. Click on **Launch**. Initially an hourglass icon appears, followed by the SP Perspectives splash screen, to let you know processing is occurring. Then, the application window (or “Perspective”) will open.

To start SP Perspectives applications (running as background processes) without using the Launch Pad, you can enter any of the following at the command line:

- sphardware &** The Hardware Perspective used to manage, control, and monitor hardware
- spevent &** The Event Perspective used to set up and manage events
- spsvd &** The IBM Virtual Shared Disk Perspective used to create, manage, and monitor virtual shared disks
- spsyspar &** The System Partitioning Aid Perspective used to define system partitions
- spperfmon &** The Performance Monitor Perspective used to set up and configure performance monitoring

You can also start SP Perspectives applications from the Common Desktop Environment (CDE). (Double-click on the SP Perspectives icon in the CDE Application Manager window.)

Understanding the SP Perspectives Interface

SP Perspectives consists of the following graphical elements:

- Launch Pad
- Perspectives Application Windows (also known as “Perspectives”)
- Dialog boxes
- Notebooks

- Message boxes

The Launch Pad

The Launch Pad starts applications associated with managing an SP system. It also provides options for customizing the appearance of the Launch Pad.

System management applications provided by SP Perspectives appear as icons on the Launch Pad. Other system management tools, such as SMIT panels and the SP Resource Center are also accessible from the Launch Pad. In addition, you can add your own applications to the Launch Pad. For descriptions of the applications provided by default, select Options → Show Application Details.

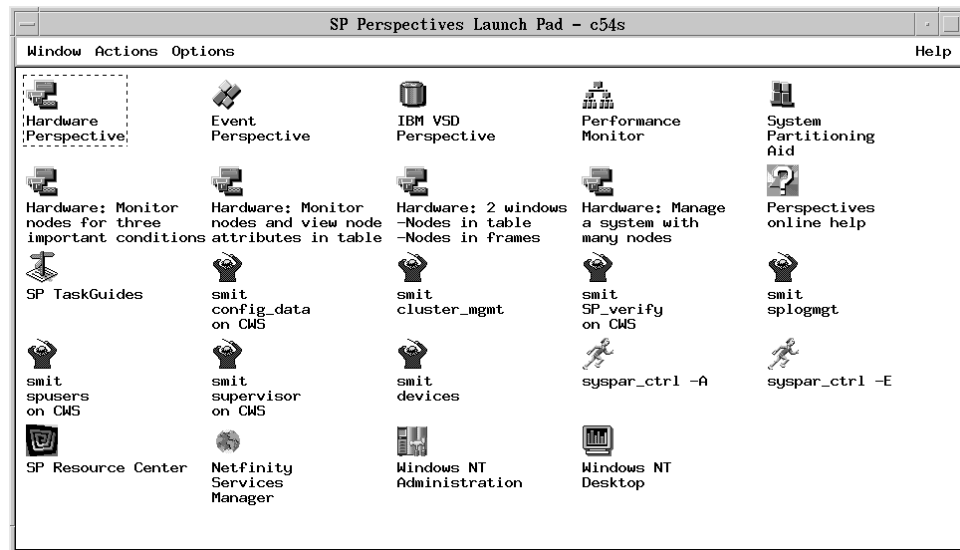


Figure 31. The SP Perspectives Launch Pad

For detailed information on the elements of the Launch Pad, see the Perspectives online help information.

SP Resource Center

The SP Resource Center provides one simple interface for all softcopy SP documentation and information resources. It consists of HTML, Java, and Javascript files and works with a Web browser. The Resource Center provides access to a variety of information including publications, READMEs, Redbooks, White Papers, SP product information, as well as up-to-date service information.

To access the SP Resource Center, double click on the SP Resource Center icon and the SP Resource Center window will appear once the Web browser has launched. The left margin of this window contains navigational aids to help you find the information for which you are looking. For example, to find out information on PSSP documentation, you would select Publications. Similarly, to find out information on SP hardware or software, select Product Information.

SP TaskGuides

SP TaskGuides are a form of advanced online assistance designed to walk you through complex or infrequently performed tasks. Each TaskGuide does not simply list the required steps. It actually performs the steps for you, automating the steps to the highest degree possible and prompting you for input only when absolutely necessary. You might recognize them as *wizards*.

The following TaskGuides are available:

- Set Site Environment Information
- Add Frames
- Configure New Nodes
- Create Node Image

Perspective Application Windows

The application window for each SP Perspectives application is called a “Perspective” because it provides a unique view of your SP system. The Hardware Perspective, for example, allows you to view the system as a set of hardware objects (for example, nodes, frames, switches). The Event Perspective allows you to view the system in terms of conditions of interest related to system resources.

These windows contain work areas known as *panes*. Panes contain icons representing system objects (a managed SP system resource such as a node, frame, or switch).

- To perform actions on the way objects are displayed in a pane:
 1. Click on the pane to give it focus.
 2. Select **View** from the menu bar.
 3. Select the desired action from the pulldown menu. (Alternatively, some of the **View** options are available from the tool bar.)
- To perform actions on the *objects themselves*:
 1. Select one or more objects.
 - To select multiple objects that are next to each other in a pane, click on the icon of the first desired object, hold the left mouse button down, and move the cursor over the icons of other desired objects in the pane.
 - To select multiple objects that are not next to each other in a pane, hold the **Ctrl** key on your keyboard down while you click on each desired object in the pane.
 2. Select **Actions** from the menu bar.
 3. Select the desired action from the pulldown menu. (Alternatively, some of the **Actions** are available from the tool bar.) Note that the choices available under the Actions menu may vary depending upon the type of objects selected.

Each Perspective consists of the same basic format as illustrated in Figure 32 on page 270.

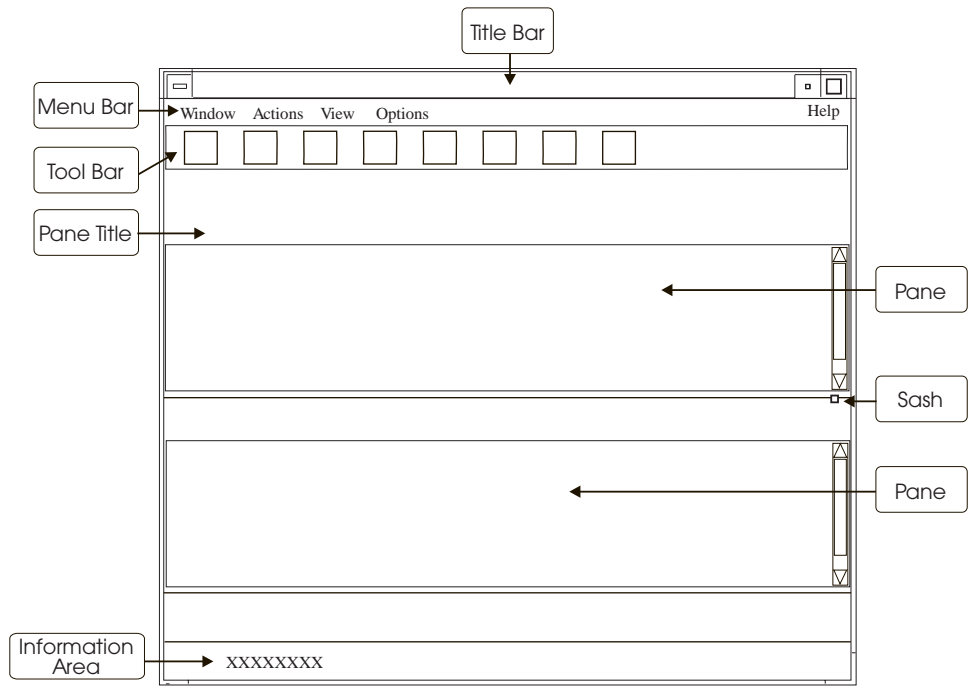


Figure 32. Elements of an SP Perspectives Window

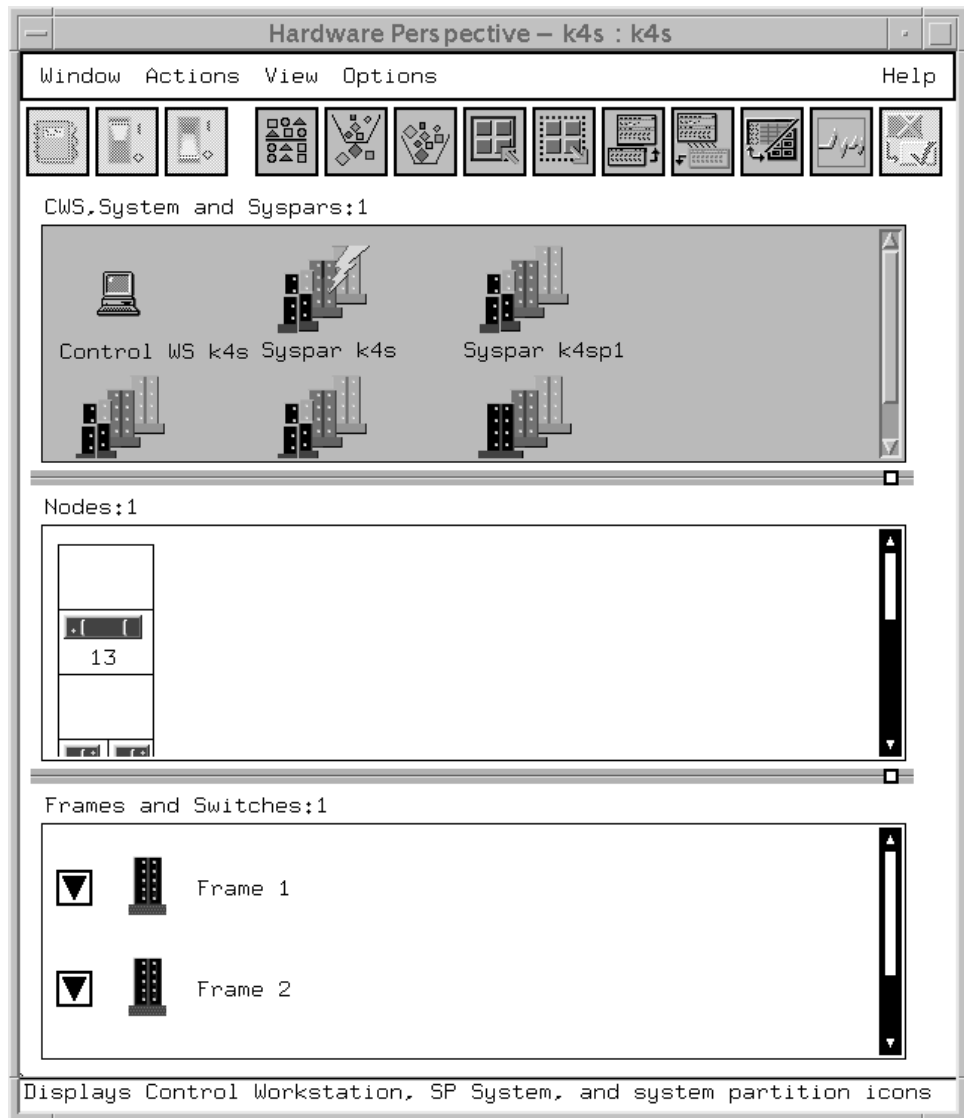


Figure 33. The SP Perspectives Hardware Perspective

For detailed information on the elements of an application window, see the Perspectives online help information.

Dialog Boxes

When you use an SP Perspectives application window to perform certain actions, a dialog appears. The dialog box solicits or displays additional information needed to complete the action. The specific information or actions contained in the dialog box depend on the application.

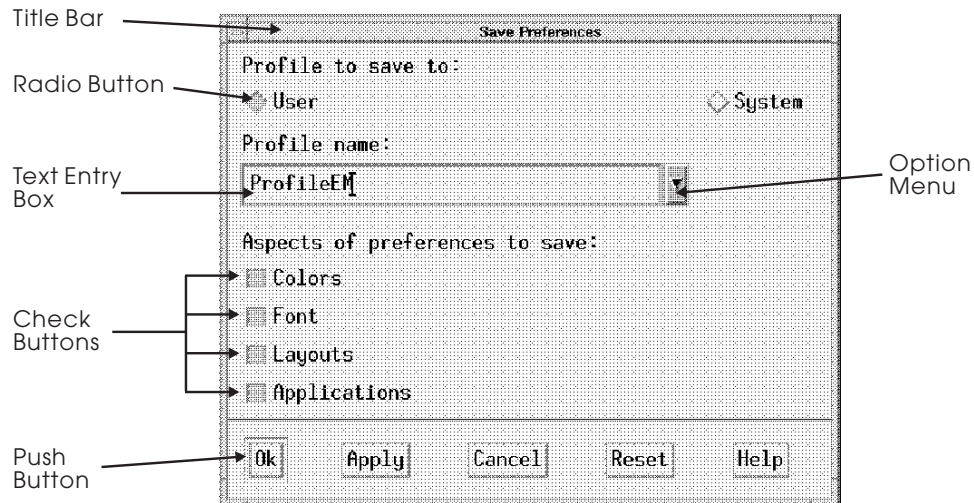


Figure 34. An SP Perspectives Dialog Box

For detailed information on the elements of a dialog box, see the Perspectives online help information.

Notebooks

A notebook lets you view or alter object properties. Each page in the notebook contains property information (defining or operational characteristics) about system objects. Similar properties are grouped on the same page and identified by tabs. The specific property information contained on a notebook page depends on the object.

Notebooks typically contain the following elements:

- The title bar, which displays the action (view, modify), the object name or type, and optionally, the system partition name.
- Text entry boxes for entering text information for object attributes that can be modified.
- Static text entry fields for displaying object attribute information that cannot be modified (view only).
- Check buttons for selecting one or more choices from a list of options that are not mutually exclusive.
- Radio buttons for making a single choice from a list of mutually exclusive options.
- Option menus for entering information and selecting a single choice from a list of mutually exclusive options.
- Buttons for selecting an action:
 - OK - updates attribute information and closes the notebook.
 - Apply - updates attribute information and leaves the notebook open.
 - Cancel - cancels any changes made to the original values and closes the notebook.
 - Reset - resets any changes made to the original values and leaves the notebook open.

- Help - displays general help about using notebooks.

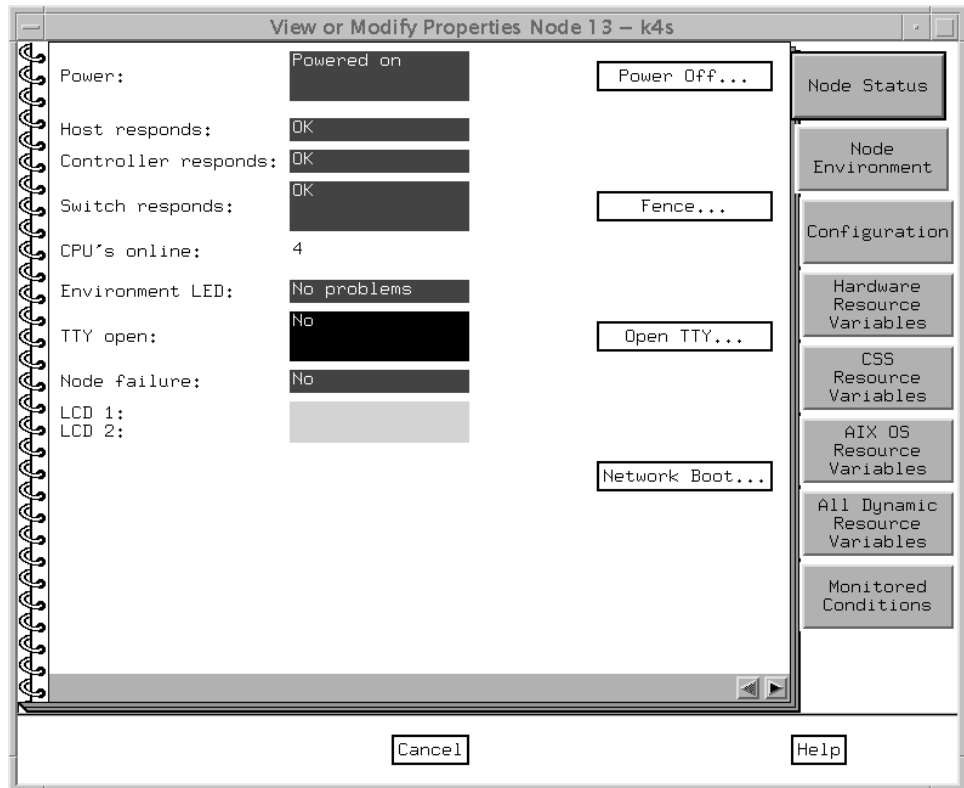


Figure 35. An SP Perspectives Notebook

Message Boxes

A message box is a type of dialog box used to convey a message to you. Following is a list of the message types:

- Informational
- Warning
- Error

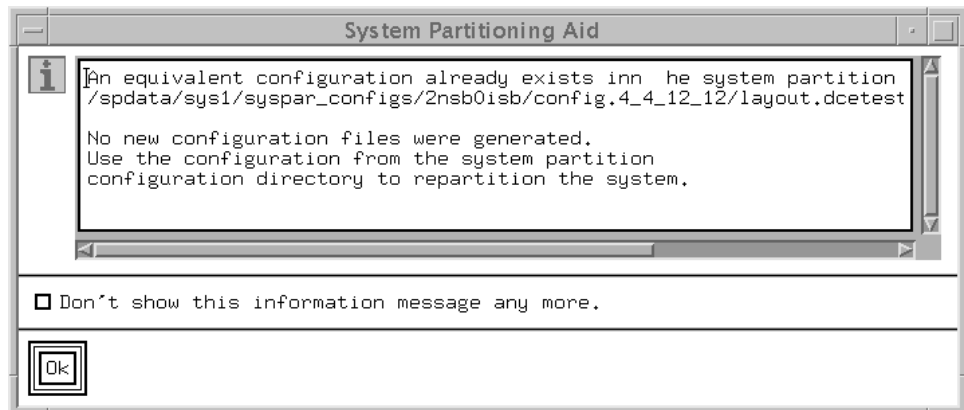


Figure 36. An SP Perspectives Message Box

For detailed information on the elements of a message box, see the Perspectives online help information.

Accessing Help

You can access help on the:

- Menu bar **Help** button
- The bubble-up text
- Information area
- Dialog box, notebook, and message box **Help** buttons

The Menu Bar

The menu bar **Help** contains items that provide information about windows, actions (tasks), and objects. To display the menu items, click on **Help** in the upper right hand corner of the menu bar.

- **Overview** activates a standalone help volume that contains overview and task-oriented information for SP Perspectives.
- **Contents** displays a list of hyper-linked help topics for SP Perspectives.
- **Tasks** accesses task-oriented help information for the Perspective that you are in when you click on **Tasks**.
- **Using Help** provides information on how to access help for SP Perspectives.
- **About SP Perspectives** displays the name and version of the application.
- **Give Feedback to IBM** provides information about how to contact the SP Perspectives team with your comments.

The Bubble-Up Text

The bubble-up text is present for all icons on the tool bar and provides a brief description of the function of that icon. Bubble-up text is also available when the mouse is placed over the title and monitoring label above a pane. This text describes pane related information such as the number of objects in the pane, number of objects filtered, the label used for the objects, and what objects are being monitored for specific conditions.

The Information Area

The Information Area is present at the bottom of application windows and displays information about the object or area at the current cursor location. You can toggle this display on and off from the **Options** menu.

The Dialog Box, Notebook, and Message Box Help Buttons

The dialog box and notebook **Help** buttons display information about the purpose of items in the dialog box or notebook. The message box provides an **Error Stack** button which displays extended information on the contents of the message and provides a way to view internal diagnostics information.

Performing Common Administration Tasks with SP Perspectives Previously Performed Using System Monitor GUI

If you previously used the System Monitor GUI to perform hardware administrative tasks, you now can use the Hardware Perspective to perform the same functions. Table 14 lists the System Monitor GUI function and the corresponding Hardware Perspective function. **Note that this table does not provide detailed instructions on performing these tasks using the Hardware Perspective. For more detailed information, see the Hardware Perspective online help.**

<i>Table 14 (Page 1 of 4). System Monitor to Perspectives Cross Reference</i>	
System Monitor GUI function	Equivalent function in Perspectives
All node summaries	<p>To monitor the conditions of nodes:</p> <ol style="list-style-type: none"> 1. Ensure that the Nodes pane is displayed and is the current pane. You may need to add a Nodes pane first. Then, click in the Nodes pane to make it the current pane. 2. Select View → Set Monitoring from the menu bar or click on the Monitoring icon on the tool bar. 3. Select one or more conditions to monitor. The most common conditions you may want to monitor include hostResponds, switchResponds, and nodePowerLED <p>The most commonly used all node summaries are available for monitoring using Perspectives. Note that the 3DigitDisplay previously available using spmon, can be seen by selecting Actions → LCD/LED Display. The Nodes pane must be the active pane in the window.</p> <p>Note that you can add multiple Nodes panes and monitor a different condition in each pane or you can monitor multiple conditions inside one pane.</p>
Node Front Panel	<p>Variables and controls displayed on the Node Front panel now appear on the Node Status page.</p> <ol style="list-style-type: none"> 1. Ensure that the Nodes pane is displayed and is the current pane. You may need to add a Nodes pane first. Then, click in the Nodes pane to make it the current pane and select a node for which to view properties. 2. Select Actions → View or Modify Properties from the menu bar or just click on the Notebook icon. 3. Select the Node Status page. This is the default page.
Node Environment Layout	<p>Variables and controls displayed on the Node Environment Layout now appear on the Node Environment page.</p> <p>To display this information:</p> <ol style="list-style-type: none"> 1. Ensure that the Nodes pane is displayed and is the current pane. You may need to add a Nodes pane first. Then, click in the Nodes pane to make it the current pane and select a node for which to view properties. 2. Select Actions → View or Modify Properties from the menu bar or just click on the Notebook icon. 3. Select the Node Environment page.

Table 14 (Page 2 of 4). System Monitor to Perspectives Cross Reference

<p>Node Detail Layout</p>	<p>Variables displayed on the Node Detail Layout now appear on the Node Environment page except for the following:</p> <ul style="list-style-type: none"> • type (node supervisor card type) and codeVersion (node supervisor code version) are shown on the Configuration page • powerLED (Power) and envLED (Environment LED) are shown on the Node Status page. <p>To display this information:</p> <ol style="list-style-type: none"> 1. Ensure that the Nodes pane is displayed and is the current pane. You may need to add a Nodes pane first. Then, click in the Nodes pane to make it the current pane and select a node for which to view properties. 2. Select Actions → View or Modify Properties from the menu bar or just click on the Notebook icon. 3. Select either the Node Environment page, the Configuration page, or the Node Status page.
<p>Frame Environment Layout</p>	<p>Variables displayed on the Frame Environment Layout now appear on the Frame Status page in the Frame notebook.</p> <p>To display this information:</p> <ol style="list-style-type: none"> 1. Ensure that the Frames and Switches pane is displayed and is the current pane. You may need to add a Frames and Switches pane first. Then click in the pane to make it the current pane and select a frame for which to view properties. 2. Select Actions → View or Modify Properties from the menu bar or just click on the Notebook icon. 3. Select the Frame Status page in the Frame notebook.
<p>Frame Detail Layout</p>	<p>Variables displayed on the Frame Detail Layout now appear on the Frame Configuration page in the Frame notebook.</p> <p>To display this information:</p> <ol style="list-style-type: none"> 1. Ensure that the Frames and Switches pane is displayed and is the current pane. You may need to add a Frames and Switches pane first. Then click in the pane to make it the current pane and select a frame for which to view properties. 2. Select Actions → View or Modify Properties from the menu bar or just click on the Notebook icon. 3. Select the Frame Configuration page in the Frame notebook.
<p>Switch Environment Layout</p>	<p>Variables displayed on the Switch Environment Layout now appear on the Switch Status page in the Switch notebook.</p> <p>To display this information:</p> <ol style="list-style-type: none"> 1. Ensure that the Frames and Switches pane is displayed and is the current pane. You may need to add a Frames and Switches pane first. Then click in the pane to make it the current pane and select a switch for which to view properties. 2. Select Actions → View or Modify Properties from the menu bar or just click on the Notebook icon. 3. Select the Switch Status page in the Switch notebook.

<i>Table 14 (Page 3 of 4). System Monitor to Perspectives Cross Reference</i>	
Switch Front Panel	<p>Variables and controls displayed on the Switch Front Panel now appear on the Switch Status page in the Switch notebook.</p> <p>To display this information:</p> <ol style="list-style-type: none"> 1. Ensure that the Frames and Switches pane is displayed and is the current pane. You may need to add a Frames and Switches pane first. Then click in the pane to make it the current pane and select a switch for which to view properties. 2. Select Actions → View or Modify Properties from the menu bar or just click on the Notebook icon. 3. Select the Switch Status page in the Switch notebook.
Switch Detail Layout	<p>Variables displayed on the Switch Detail Layout now appear on the Switch Status page in the Switch notebook except for type and codeVersion which are on the Configuration notebook page.</p> <p>To display this information:</p> <ol style="list-style-type: none"> 1. Ensure that the Frames and Switches pane is displayed and is the current pane. You may need to add a Frames and Switches pane first. Then click in the pane to make it the current pane and select a switch for which to view properties. 2. Select Actions → View or Modify Properties from the menu bar or just click on the Notebook icon. 3. Select the Switch Status page in the Switch notebook.
Switch Diagnostics Layout	<p>Variables displayed on the Switch Diagnostics Layout now appear on the Switch Status page of the Switch notebook.</p> <p>To display this information:</p> <ol style="list-style-type: none"> 1. Ensure that the Frames and Switches pane is displayed and is the current pane. You may need to add a Frames and Switches pane first. Then click in the pane to make it the current pane and select a switch for which to view properties. 2. Select Actions → View or Modify Properties from the menu bar or just click on the Notebook icon. 3. Select the Switch Status page in the Switch notebook.

Table 14 (Page 4 of 4). System Monitor to Perspectives Cross Reference

Global Controls	<p>Information displayed through the spon Global Controls function is available:</p> <ol style="list-style-type: none"> 1. Select the nodes in the Nodes pane for which you want information 2. Select Actions from the menu bar. <p>The following actions are available:</p> <ul style="list-style-type: none"> • Power Off, Reset, or Shutdown - to perform a power off with setting certain options such as a straight power off or reset, a shutdown or cshutdown, or fence or fence with autojoin. • Power On or Cluster Power On - to power on one or more nodes with the options for performing a straight power on or a cstartup. • Change Key Switch - change key switch position on nodes that have a key switch. • Fence or Unfence - fence or unfence nodes. • Open TTY - to open TTY windows to selected nodes. • Run Command - run a command on selected nodes. • Network Boot - network boot a node
Topology	Not currently supported in Perspectives.

Chapter 19. Using the SP Command Line System Monitor

The SP System Monitor allows authorized operators and administrators to operate and monitor the system hardware. For information on the SP Perspectives GUI, refer to Chapter 18, “Using SP Perspectives” on page 263.

Releases earlier than PSSP 3.1 provided a system monitor GUI. Now there is the SP Perspectives GUI. For information on using SP Perspectives to accomplish tasks previously performed using the system monitor GUI, see “Performing Common Administration Tasks with SP Perspectives Previously Performed Using System Monitor GUI” on page 275. Though the GUI has been replaced, you can still use the line commands.

The SP System Monitor is made up of the following parts:

Hardware Monitor	A set of commands and a daemon used to monitor and control the SP hardware platform.
Command interface	The spmon , hmmon , hmcmds , s1term , and nodecond commands with their flags and parameters, provide a way to monitor and control the system at an individual system partition level or at a global (all system partitions) level. These can be used from an ASCII terminal, such as those used on a dial-up line, from scripts, or from another terminal.
Logging daemon	The splogd logging daemon logs SP hardware state changes, reports SP hardware errors, and provides a state change alert.

See Figure 37 on page 280 for an overview of the SP System Monitor.

This chapter provides conceptual information and procedures for configuring and using the SP System Monitor.

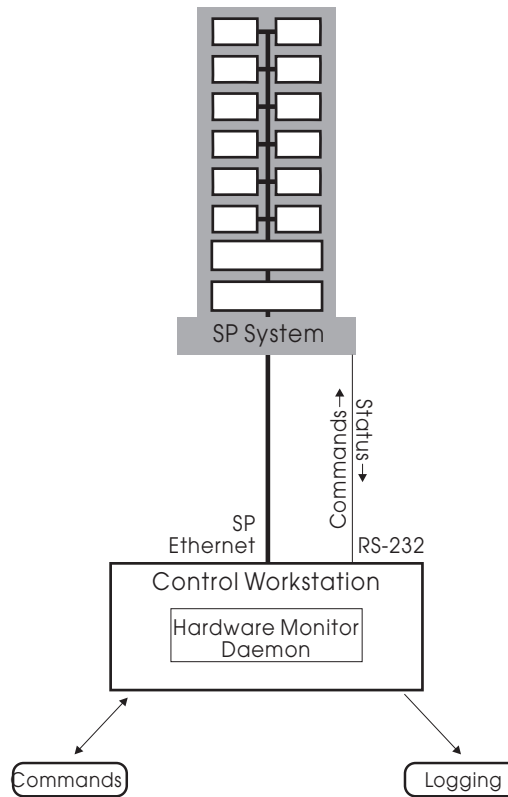


Figure 37. The SP System Monitor

Understanding the Hardware Monitor

The Hardware Monitor consists of a daemon, named **hardmon**, and a set of client commands. The daemon executes on the control workstation and, using the RS-232 lines to each frame, polls the frames for the state of the hardware within the frame. The daemon also sends hardware-level commands to each frame to change the state of the hardware within the frame in some way. The client commands, **spmon** and **hmmon**, interact with the daemon to obtain the current or changing state of the hardware. The client commands, **spmon** and **hmcmds**, also interact with the daemon to change the state of (that is, control) the hardware. The **splogd** logging daemon is also a client of the hardware monitor for logging purposes.

The **hardmon** daemon reports variables representing the state of a frame, node, SP Expansion I/O Unit, and switch hardware. These state variables are made available through the **spmon** and **hmmon** client commands. You can find a list of these variables with descriptions by using the Event Perspective GUI. This list provides variable names, display attributes and descriptions for all variables available in the SP System Monitor. The **hmmon** command, with the appropriate flag, also provides a list of variable names and descriptions. You will see these variable names in the SP System Monitor log. They also serve as parameters to the **spmon** and **hmmon** commands.

To diagnose system monitor problems including SP-attached server problems involving the **hardmon** daemon, refer to *PSSP: Diagnosis Guide*.

Frame Hardware State

The **hardmon** daemon reports variables to *monitor* the frame hardware status. This information includes:

- Switch failure
- Power LEDs
- Environment LEDs
- Temperature
- Voltage
- State of the power supply

The **hardmon** daemon allows the administrator to *control* the frame hardware state. This information includes:

- Power on/off

Note that there is no frame hardware state associated with the SP-attached servers.

Node Hardware State

The **hardmon** daemon reports variables to *monitor* the node hardware status. This information includes:

- Three-digit display
- Power and Environment LEDs
- Temperature
- Voltage
- Fan failure

Note: Some environment information, such as fan speed and voltage, is not available for a 604 High Node. Other environment information might not be available for the SP-attached servers.

The **hardmon** daemon allows the administrator to *control* the node hardware state. This information includes:

- Power on/off
- Key Mode Switch
- Reset button

Switch Hardware State

The **hardmon** daemon reports variables to *monitor* the switch hardware status. This information includes:

- Temperature
- Voltage
- Fan failure
- Power and Environment LEDs

The **hardmon** daemon allows the administrator to *control* the switch hardware state. This information includes:

- Power on/off
- Change multiplexor clock setting

Note: On SP Switch systems, you must use the **Eclock** command to change the multiplexor clock setting.

Authority

There are four classes of authority for the SP System Monitor:

1. Monitor authority permits users to display system statistics.
2. Control authority permits users to display system statistics and control the hardware.
3. S1 authority permits read and/or write access to a node's serial port.
4. Administrative authority permits a user to issue administrative commands to **hardmon**.

Configuring the SP System Monitor

The system administration tasks for configuring the SP System Monitor include authorizing users and configuring the error log.

Step 1: Authorize Users for the SP System Monitor

The SP System Monitor Access Control Lists (ACLs) are found in **/spdata/sys1/spmon/hmacls** on the control workstation. Edit this file if you wish to add users for your system. The **/spdata/sys1/spmon/hmacls** file is initially set up giving all levels of authority (including administrator) to the same user that is defined as the primary authentication services administrator by the **setup_authent** command (refer to Chapter 12, "Security Features of the SP System" on page 131 for more information). The **hardmon** principal is initially set up with monitor authority (for use by **splogd**).

The fields for each entry in the **/spdata/sys1/spmon/hmacls** file are

object name permissions

where:

object A frame number or hostname (where the **hardmon** daemon is running)

name A Kerberos principal name and optional instance

permissions

- v** Virtual Front Operator Panel (VFOP). This gives permission to issue commands to the hardware.
- s** Serial link. This gives permission to read and write to a serial port.
- m** Monitor. This gives permission to receive state changes.
- a** Administrative. This gives authority to control **hardmon**.

Invoke the **hmadm setacls** command after the ACL configuration file has been modified to update the hardware monitor daemon's internal ACL tables.

Refer to Chapter 12, “Security Features of the SP System” on page 131 for more information on security considerations.

Step 2: Configure the SP System Monitor Error Log

When the hardware supervisors indicate a warning or shutdown condition, the SP System Monitor writes a message using the AIX **syslog** facility and the AIX error log facility. For example, when the hardware supervisors determine that a fan has failed, the SP System Monitor writes a precise message into the log file that includes the time, node, type of error, variable name, and, in some cases, associated values.

The installation process creates the default syslog file **/var/adm/SPlogs/SPdaemon.log** on the control workstation. You might want to configure your system to send the system log information to other locations. For example, you might want to send the **SPdaemon.log** messages to another workstation for convenience. You can do this using the *@hostname* parameter in the **/etc/syslog.conf** file. For more details, see the book *IBM AIX Files Reference*. The facility name for the SP System Monitor is **daemon**.

Using the Command Line Interfaces

The SP system includes command line interfaces that allow you to perform the SP System Monitor functions without using a graphical user interface. With the **spmon** and **hmmon** commands you can query the value of a state variable or monitor that variable for a change in state. With the **spmon** and **hmcmds** commands you can control the nodes, frames and switches. The **hmmon** and **hmcmds** commands provide equivalent function to the **spmon** command but provide a more compact interface than **spmon** that can be appropriate in scripts. The **spmon** command is provided for compatibility with earlier SP software releases.

By default, these commands operate at a system partition level. That is, they work only on nodes *in the current system partition*. (To determine the current partition, issue the **spget_syspar** command.)

Note: Hardware Monitor queries or commands using a wildcard character do not return an error code if nodes outside of the current system partition are specified by the wildcard. Nodes outside of the current partition are treated as if they don't exist.

Each of the commands has a **-G** flag that removes system partition boundaries, allowing the command to work on any hardware in the SP system. The **-G** option is *always* needed to monitor or control frame or switch hardware. Frames and switches are considered to be outside all system partitions.

Before using any SP System Monitor command your identity must be authenticated. Use the **k4init** command to login to the SP authentication services. If your site uses an alternative to the SP authentication services, you must use authentication procedures appropriate to such an alternative. (Refer to Chapter 12, “Security Features of the SP System” on page 131 for more information on SP authentication services.)

See the book *PSSP: Command and Technical Reference* for a complete description of all SP System Monitor commands.

Chapter 20. Integrating TME 10 on the SP System

Tivoli Management Environment (TME 10) is a function that helps you to monitor SP events in a TME 10-managed environment. It allows an administrator to forward any event managed by the PSSP Event Management subsystem to the TME 10 Enterprise Console, which acts as a centralized point of control for the TME 10-managed environment. To define which events are forwarded by Event Management, you can use either the Event Management Perspective or the **pmandef** command.

Related Publications

This chapter does not provide all the necessary information for a complete and thorough understanding of TME terminology, concepts, and applications. For complete information, you should reference the following:

- *Integrating TME 10 on the RS/6000 SP*, Redbook written by the International Technical Support Organization. This Redbook document also has a section of related publications that you should reference.
- *PSSP: Installation and Migration Guide* contains information about installing the T/EC adapter.
- *PSSP: Command and Technical Reference* contains information on the syntax of the **pmandef** command.
- *PSSP: Diagnosis Guide* contains information on problem diagnosis related to TME 10.
- You should also reference Chapter 25, "The Event Management Subsystem" on page 337 for more information.

TME 10 Terminology

This chapter does not provide all the necessary information for a complete and thorough understanding of TME terminology, concepts, and applications. For complete information, you should refer to the Related Documentation section previously discussed. This section provides a brief overview of the major terms that are discussed in this chapter.

Tivoli Management Environment (TME)

The combination of the base TME 10 Framework, the distributed object databases, graphical user interface, command line interface, and all Tivoli toolkits and applications required within the enterprise that is managed by Tivoli.

TME 10 Framework

The base set of Tivoli software that is required to run any of the Tivoli management applications. The Framework provides basic system administration capabilities, and services for the management applications, including an administrator facility, scheduler facility, and notice facility. The TME 10 Framework is often referred to as the Tivoli Management Platform (TMP).

Tivoli Management Region (TMR)

The basic physical unit of Tivoli functionality. It consists of one TME server and the clients that server is managing.

Policy Regions, Policies

A policy region is a collection of TME resources that are controlled by a common set of policies or rules. Typically, policy regions define boundaries of the authority of Tivoli administrators, as well as provide a mechanism for organizing and managing system resources in a hierarchal structure.

Profiles, Profile Managers

A profile contains a collection of application-specific information. The information in a profile is specific to a particular profile type (for example, a user profile will contain user names, login ids, and so on). A profile manager contains profiles and a list of subscribers to which the profile data can be distributed. Subscribers can be managed nodes, PC managed nodes, NIS Domains, and other profile managers.

Administrators

A Tivoli administrator is a user that has been given authorization to perform management tasks in the TME. The administrator's authorization roles determine what tasks that administrator can perform against a set of resources.

Resources

A TME resource is a general term used to define systems, devices, services, and facilities in a distributed system. A managed resource can be owned by only one policy region.

TME 10 Application Extension Facility (AEF)

An interface to dynamically customize the Tivoli application by adding site-specific behavior or values to standard applications.

TME 10 Event Integration Facility (EIF)

A facility to build event adapters to map events from any application resource, or component into a format compatible with the TME 10 Enterprise Console.

TME 10 Application Developers Environment (ADE)

Programming tools for creating new custom management applications on top of the TME 10 Framework.

Understanding the TME 10 Enterprise Console

The TME 10 Enterprise Console (T/EC) is one of Tivoli's availability management applications. As a management tool, it assists in maintaining high availability of the myriad of networks, system applications, and databases found within the scope of an enterprise, and provides a centralized point of control for all critical messages stemming from these resources.

The computing resources in the distributed environment generated messages in a variety of formats. The function that captures these messages is called an event adapter. Each event adapter installed on or near the monitored computing resources is familiar with the structure of that raw data and transforms the information it receives by parsing and restructuring it before sending it to the T/EC event server in a T/EC acceptable format, a special structured format named

BAROC (Basic Representation of Object in C). BAROC is a simple event class structure that allows important elements of a message to be separated into a number of pieces of information called slots. Each event is defined as a member of a class.

A number of different types of T/EC event adapters may exist on any TCP/IP connected system, not only on TME managed nodes. Besides generating the events and sending them to the T/EC event server, the adapters also filter out extraneous information so only significant events are forwarded to the T/EC event server. This helps reduce the demand on the network and the amount of processing done by the T/EC event server when the computing resources are sending out large numbers of messages.

Defining the PSSP T/EC Adapter

The PSSP T/EC adapter is a tool that forwards events generated by the Event Manager subsystem to the TME 10 Enterprise Console. The PSSP T/EC adapter receives the PSSP events, extracts the event information, and formats it into a T/EC event notification automatically. The tool simplifies the forwarding of events, since it does not require any other program or script to parse the event strings. The adapter receives all the event information from the Event Management subsystem and formats it into T/EC events using the PSSP _EVENT classes, which are defined in the **pssp_classes.baroc file**. Using the **tecad_pssp** program allows you to subscribe events using the Event Management Perspective or the **pmandef** command and have this forwarded to T/EC.

This chapter does not provide details on how the PSSP T/EC adapter forwards events to the TME 10 Enterprise Console, how to use or install the adapter, or how to define event classes for PSSP events. For this detailed information, refer to the *Integrating TME 10 on the RS/6000 SP*, SG24-2071 Redbook.

Defining an Event and Forwarding it to the T/EC Using the Event Perspective

Using Event Management Perspective makes administering and managing the events in the SP system relatively easy. Once you start the Event Management Perspective:

SELECT Actions → Event Definitions → Create

The Event Definition window appears where you can select the parameters for the event definition.

SELECT The Response Options tab and enter the response options.

SELECT Take Actions when event occurs and enter the **tecad_pssp** command in the command window. Be sure to provide the full path for the command to supply a full path to your configuration file, if you are not using the default.

Repeat the same procedure for the **rearm** command if you wish. On the lower part, select the control workstation (Node 0) as the node where to run the command. Install the **tecad_pssp** command on all nodes, but IBM recommends that you use the control workstation as a central point for event forwarding since it is accessible by all partitions.

Defining an Event and Forwarding it to the T/EC Using the pmandef Command

Consult the *PSSP: Command and Technical Reference* for complete command syntax. A general example of this command follows:

```
pmandef -s example1
-e "AnyResourceVariable;Any InstanceVencor;AnyPredicate"
-c "$AGENT_PATH/tecad_pssp -1 $CONF_PATH/tecad_pssp.cfg"
-r "AnyRearmPredicate"
-C "$AGENT_PATH/tecad_pssp -1 $CONF_PATH/tecad_pssp.cfg"
-n 0
```

You should run this command from the control workstation to save installation efforts and keep the management of the system easier. This results in the flag `—n 0`, indicating that the command needs to be run on Node 0, the control workstation.

Debugging Problems with Event Generation and Reception by the T/EC

For debugging and problem diagnosis information, refer to the *IBM Parallel System Support Programs for AIX Diagnosis Guide*.

Chapter 21. Managing an SP-Controlled Netfinity Server

This chapter provides information to help you manage the SP-controlled Netfinity server. It briefly describes the tools that are available when you connect the Netfinity server with the control workstation and use the SP/Netfinity Server Consolidation component of PSSP.

SP-controlled Netfinity server support offers three levels of Netfinity server management:

- Direct power monitoring and control via SP Perspectives on the control workstation.
- Casual management using the Netfinity Services Manager Web interface, and optionally, the Web Administration for Windows NT software provided in the NT 4.0 Server resource kit or the Microsoft BackOffice product.
- Full native Netfinity hardware and Windows NT management displayed to the control workstation using the optional, separately installed Windows NT Terminal Server and Citrix MetaFrame products.

Netfinity Management Interfaces

These management interfaces are based on the Netfinity server:

- Netfinity Services Manager

This is the primary system management software supplied by IBM with each Netfinity server. Netfinity Services Manager provides the ability to monitor and control a set of attributes on the Netfinity server. Some functions included are: alert management, critical file monitoring, predictive failure analysis, and service processor configuration. Netfinity Services Manager also enables you to manage groups of servers from a single server. It is primarily a hardware management interface. For more information see the IBM documentation provided with your Netfinity server or at the Web address <http://www.pc.ibm.com/us/netfinity>

- Web Administration for Microsoft Windows NT Server (Web Administration for Windows NT)

The Web Administration for Windows NT is a Web-based NT administration interface. Through this add-on to NT (available in either the NT Resource Kit or as part of the BackOffice Server suite), you may view and modify basic NT operating system attributes. Among other things, this tool provides you with access to NT user management, NT services management, and file system management. Unlike Netfinity Services Manager, this tool is concerned mainly with operating system management, not hardware management.

- NT Remote Display Client Products

These applications enable you to launch native NT management applications on the Netfinity servers and display the application GUIs on the control workstation. This gives you the full functionality of any NT or Netfinity management application regardless of a given application's ability to provide a Web interface. To achieve this capability, you must purchase, install, and configure Windows NT 4.0 Terminal Server Edition on the target Netfinity server, along with a third party software package, Citrix MetaFrame by Citrix

Systems Inc. on the Netfinity server and the SP control workstation. After installation, the SP Perspectives launchpad must be configured to use the remote display client. See *Configuring Optional NT Administration Products* in the Netfinity section of the book *PSSP: Installation and Migration Guide*.

For installation instructions refer to the book *PSSP: Installation and Migration Guide*.

Managing Netfinity Nodes from Perspectives

Basic hardware control and monitoring of Netfinity nodes is provided through the SP Perspectives when you install the SP/Netfinity Server Consolidation software. This section provides information about using Perspectives to manage the Netfinity servers. For general information regarding SP Perspectives, see Chapter 18, "Using SP Perspectives" on page 263.

Three icons in the SP Perspectives Launch Pad provide access to Netfinity server management:

- Netfinity Services Manager

This icon launches the Web interface for Netfinity Services Manager, the primary system management software provided by IBM with each Netfinity server. This is a hardware management interface.

- Windows NT Administration

This icon launches the optional Web Administration for Windows NT Server, the Web-based NT administration interface. Unlike Netfinity Services Manager, this tool is concerned mainly with operating system management, not hardware management.

- Windows NT Desktop

This icon launches the optional NT remote display client that you might have chosen to configure. For a list of the supported packages, see the Netfinity information in the book *PSSP: Installation and Migration Guide*.

Bringing up Perspectives

Use the **perspectives** command to bring up the SP Perspectives Launch Pad, then double-click on the Hardware Perspective icon. You can also start the Hardware Perspective, without opening the SP Perspectives Launch Pad, by using the **sphardware** command.

The default window for Hardware Perspective with the Netfinity management software installed contains:

- The tool bar with an icon to launch Netfinity Services Manager
- The CWS, System, and Syspars pane
- The Nodes pane (in frame view)
- The Netfinity nodes pane (frame view)

For specific information regarding the other icons appearing in the tool bar, or the panes, see the online Perspectives Help, which is accessible from the Launch Pad and this window.

The Netfinity Nodes Pane

The Netfinity Nodes pane enables you to:

- View the Netfinity nodes in icon, table, and frame view
- Display the Netfinity nodes notebook by double-clicking a Netfinity node, selecting a node and clicking the notebook icon on the tool bar, or using the Action drop-down menu.
- Power a Netfinity node on and off from the Tool bar, the Action drop-down menu, or the Notebook
- Acknowledge the state of monitored Netfinity nodes
- Monitor for the nodePowerDown condition

The Netfinity Node Notebook

The Netfinity Node notebook contains all attributes of the Netfinity nodes.

Access to Netfinity Services Manager has also been provided on the Node Status Page of the notebook.

For details and information about using the Netfinity node notebook, see the online Perspectives Help.

The Frame Pane

Netfinity nodes can only be monitored as contained objects in this pane. They can be monitored for the nodePowerDown condition.

The CWS, System, Syspars Pane

The CWS, System and Syspars pane will allow monitoring and hardware control of Netfinity nodes:

- Monitoring

If the Syspars tab is selected from the Set Monitoring dialog box the nodePowerDown condition for the Netfinity nodes will appear as one of the conditions for contained objects and can be monitored.

If the System tab is selected from the Set Monitoring dialog box the nodePowerDown condition for the Netfinity nodes will appear as one of the conditions for contained objects and can be monitored.

- Hardware Control

If a system partition is selected and the Power On action is chosen from the tool bar, or the Actions menu, two dialog windows will be displayed; the SP Node Power On dialog box and the Netfinity Node Power On dialog box.

If a system partition is selected and the Power Off action is chosen from the tool bar, or the Actions menu, two dialog windows will be displayed; the SP Node Power Off dialog box and the Netfinity Node Power Off dialog box.

Availability and Problem Management

Chapter 22. Managing a High Availability Control Workstation Configuration

This chapter provides overview and task information for the High Availability Control Workstation (HACWS) function. The HACWS function provides for a backup control workstation for your SP system.

IBM assumes that you have planned for, installed, and configured the HACWS function.

For information on planning for HACWS function, refer to *IBM RS/6000 SP: Planning Volume 1, Hardware and Physical Environment* and *IBM RS/6000 SP: Planning Volume 2, Control Workstation and Software Environment*.

IBM also suggests that you refer to the documentation for AIX High Availability Cluster Multi-Processing (HACMP for AIX). HACMP is the foundation for HACWS function.

Information is provided in this chapter for the following tasks:

- Understanding HACWS function
- Modifying typical administrative tasks (maintaining the **/etc/passwd** database, setting up mail spools, setting up file collections, for example) so that subsystems will fail over to the backup control workstation in the event the primary control workstation becomes unavailable. This information is covered in the order of the related chapters in the *PSSP: Administration Guide*:
 - Starting Up and Shutting Down the SP System
 - Parallel Management Commands
 - Managing File Collections
 - Managing User Accounts
 - Managing SP Resources
 - Managing Time Synchronization
 - Managing an Automounter
 - Managing Mail Service
 - Accounting
 - Security Features of the SP System
 - Sysctl
 - Using a Switch
 - Managing System Partitions
 - Managing Extension Nodes
 - Using the SP System Monitor or SP Perspectives
- Customizing HACMP
- Verifying Frame Supervisor Cables

Understanding HACWS Function

The following definitions will help in understanding HACWS function.

Primary control workstation

The first control workstation that is installed with the SP system. The primary control workstation is a physical machine.

Backup control workstation

The standby backup control workstation that is installed during the HACWS installation. The backup control workstation is a physical machine.

Active control workstation

The control workstation that is currently running the SP control workstation applications (the SDR and **hardmon**, for example). This can be either the primary or backup control workstation.

Inactive control workstation

The control workstation that is not running the SP control workstation applications. This can be either the primary or backup control workstation.

The HACWS function improves the reliability, availability and serviceability of an SP system by providing for a backup control workstation. HACWS provides for extending the SP system configuration to include a second RS/6000 to function as a backup to the primary control workstation. Automatic failover and reintegration to the backup control workstation are provided should the primary control workstation fail or in the case of taking the primary control workstation down to perform hardware and software maintenance.

HACWS provides hardware and software features that remove a single point of failure for the SP system. HACWS does not, however, protect against double failures. You should not expect control workstation functions to work during the time that a control workstation is failing over. If a failure on the control workstation occurs while a control workstation function is being performed, the function needs to be restarted.

The following figure illustrates a typical HACWS configuration:

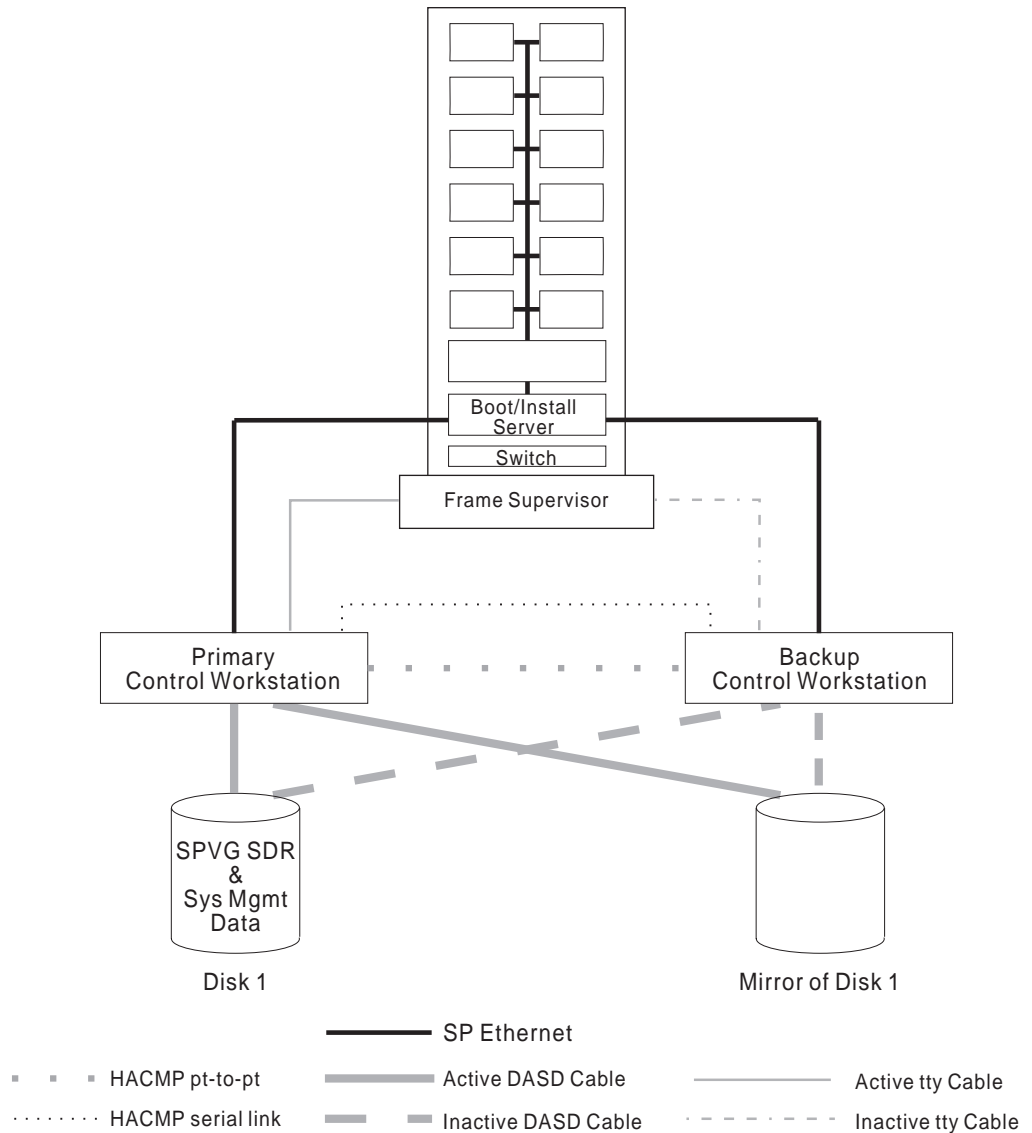


Figure 38. An HACWS Configuration

Hardware

HACWS is a two-node configuration: a primary control workstation with a backup control workstation. (Do not confuse the use of *node* in this context with an SP system node.) IBM suggests that the two RS/6000s be configured in the same way with the same model hardware and I/O configurations. This is not required, but simplifies the management of the HACWS configuration. IBM suggests that the primary and backup control workstations do not use the same power source.

External Fixed Disks and Disk Controllers

External fixed disks that provide only nonconcurrent access are used in an HACWS configuration. IBM suggests that the external fixed disks be mirrored across two disk controllers. If the external fixed disks are not mirrored, the single point of failure has not been removed from the SP system. The fixed disk becomes the single point of failure: if the disk fails the SP system will be down until the disk can be replaced. IBM suggests that the two disk controllers be on different power sources.

Frame Supervisor Cards

HACWS requires a dual RS-232 frame supervisor card per frame with a connection from each control workstation to each SP frame. Hardware feature #1245 provides this new type of frame supervisor card with an RS-232 Y-cable that connects from a frame in the SP system to the control workstation. All frames in the SP system need this new frame supervisor card and the Y-cables in order for the HACWS software to be configured. The frame supervisor connections from each frame must be connected to the same tty ports on both control workstations. If not cabled correctly, frame supervisor connections will not be activated by the hardware monitor.

SP-Attached Servers

The control workstation is attached to the S70 and S7A SP-attached servers through two serial connections; one for hardware monitoring and control and the other for serial terminal support. Only one control workstation can be connected to these servers at a time, so there is no automatic HACWS support for the SP-attached servers. When the primary control workstation fails over to the backup control workstation, hardware control and monitoring support and serial terminal support will not be available for these servers.

Network Connections

One of the following network connections must exist between the primary and backup control workstations:

- A dedicated TCP/IP network link
- An RS-232 tty link
- The target mode SCSI across the external SCSI fixed disks

Adapters

Each control workstation requires the same number of connections to the SP Ethernet on the same LAN segments. Each SP Ethernet LAN segment must be cabled to the same Ethernet (**enx**) adapter. Standby network adapters are optional in an HACWS configuration because there may not be enough adapter slots in the control workstation for a standby adapter on each service network. On a small one- or two-frame SP system the normally used control workstation has only two slots and does not have enough slots for standby adapters. In a large multi-frame SP system the number of slots required for frame supervisor and SP Ethernet LAN segments may not allow for standby adapters. However, the presence of a standby adapter may avoid the need to fail over to the inactive control workstation.

Software

The software foundation for HACWS is the separately orderable AIX High Availability Cluster Multi-Processing (HACMP for AIX). Only the high availability subsystem is required. The concurrent Resource Manager of HACMP is not supported (all the SP data on the control workstation is in an AIX Journaled File System and, therefore, does not allow concurrent access).

You may use any level of HACMP that is supported with the level of AIX that you are using. Refer to the appropriate HACMP documentation to determine what levels of HACMP are supported with the level of AIX that you are using.

Note: In an HACWS configuration, the nodes may be running any level of AIX and PSSP.

The primary and backup control workstation are configured in a two-node rotating configuration. The external fixed disks are configured in nonconcurrent access only. For information on HACMP, refer to *HACMP: Concepts and Facilities*.

Building on HACMP, the **ssp.hacws** optional installation package that is part of the IBM Parallel System Support Programs for AIX provides:

- Scripts to create a backup control workstation
- Verification programs
- HACMP pre- and post- event scripts
- Scripts to synchronize the backup and primary control workstations

When a failure of the primary control workstation takes place there is a disruptive failover (that is, the failover is noticeable) of the control workstation to the backup control workstation. This failover:

1. Switches the external fixed disks
2. Reconfigures network adapters from boot addresses to service addresses to perform IP address takeover
3. Performs hardware address takeover
4. Remounts file systems
5. Restarts the control workstation applications
6. Resumes hardware monitoring
7. Allows clients to reconnect to obtain services or update control workstation data

The backup control workstation assumes the IP address (service address) and IP aliases of the primary control workstation, resulting in only one active control workstation at a time and allowing client applications to run without changes. See Figure 39 on page 300 for an illustration of addressing in an HACWS configuration.

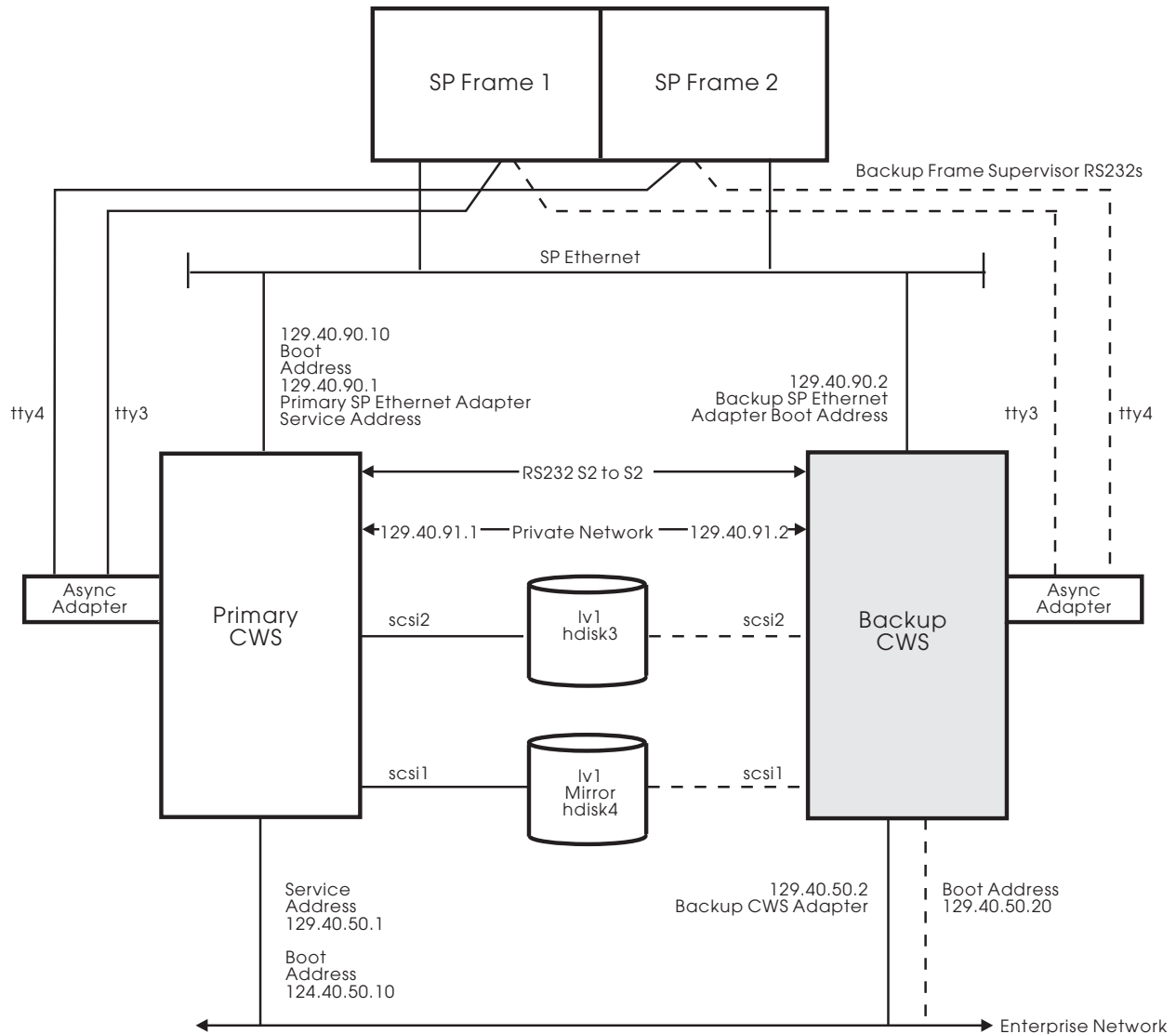


Figure 39. Addressing in an HACWS Configuration

Operating HACMP and the Control Workstation

This section describes some of the operational characteristics of the control workstation and HACMP.

Booting the Control Workstations

In a two-node rotating HACMP configuration, if you are booting both the primary and backup control workstations at the same time, the first control workstation to have the cluster manager started will acquire the shared resources. In an HACWS scenario this will cause the control workstation IP addresses (service address) and the `/spdata/sys1` file system to be configured on that control workstation. It is important that the RS/6000 that is intended to be the active control workstation have the HACMP cluster manager started on it first. To help ensure that the correct control workstation acquires the shared resources, IBM suggests that the cluster manager be started manually instead of being defined to start at system restart. This allows both control workstations to be booted in any order and any potential race conditions can be avoided during the boot process. After both control

workstations are booted the cluster manager can be started on the intended control workstation.

If the control workstation function is started on the unintended control workstation, you can perform the following sequence of events to make the intended control workstation active:

1. Boot the intended control workstation without starting the cluster manager.
2. Do a graceful shutdown without takeover on the incorrectly active control workstation.

Using SMIT:

TYPE `smit clstop`

- The Stop Cluster Services menu is displayed.

SELECT now (for Stop now option)

SELECT graceful (for Shutdown mode option)

PRESS Enter (to Do)

Using the Command Line:

Enter

```
/usr/sbin/cluster/utilities/clstop -y -N -g
```

Note: **Never** use **kill** to stop the HACMP cluster manager. Doing so will cause the node to fail.

3. Start the cluster manager on the intended control workstation.

Using SMIT:

TYPE `smit clstart`

- The Start Cluster Services menu is displayed.

SELECT now (for Start now option)

SELECT true (for Startup Cluster Information Daemon? option)

Note: Don't use the **on system restart** option.

PRESS Enter (to Do)

4. Execute the **clstat** utility to make sure HACMP is up and running on the correct control workstation.
5. Start the cluster manager on the inactive control workstation.

Avoiding False Adapter Failures on Configurations Without Standby Adapters

If a control workstation is configured without a standby adapter on one of its service networks, and the other control workstation is powered off, the HACMP software may find it difficult to determine service adapter failure. This is because the HACMP cluster manager cannot use a standby adapter to force packet traffic over the service adapter to verify its operation. This shortcoming is less of an exposure if one or more of the following is true:

1. There are network devices which answer broadcast ICMP ECHO requests. This can be verified by pinging the broadcast address and determining the number of different IP addresses that respond.

2. The service adapter is under heavy use. In this instance the inbound packet count will continue to increase over the service adapter without stimulation from the cluster manager.

If neither of these cases is true, then HACMP may report that a service adapter has failed, even though there has not been a failure; it is just that there are not any other adapters to communicate with. (For example, this can occur when all the nodes are powered off.)

An enhancement to **netmon**, the network monitor portion of the HACMP cluster manager, is described below. It can be configured to allow more accurate determination of a service adapter failure. This function can be used in configurations that require a single service adapter per network.

A **netmon** configuration file, `/usr/sbin/cluster/netmon.cf`, will specify additional network addresses to which ICMP ECHO requests can be sent. The configuration file consists of one IP address or IP label per line. The maximum number of addresses used is five. All addresses specified after the fifth will be ignored. No comments are allowed in the file.

The following is an example of a **netmon.cf** configuration file:

```
180.146.181.119
steamer
chowder
180.146.181.121
musse1
```

This file must exist at cluster startup. The cluster software will scan the configuration file during its initialization phase. When **netmon** needs to stimulate the network to verify adapter function, it will send ICMP ECHO requests to each address. After sending the request to every address, **netmon** will check the inbound packet count before determining whether there is an adapter failure.

Conditions that Cause a Failover of the Control Workstation

The following conditions will cause a failover of a control workstation:

1. An intentional failover

This occurs when the operator stops the cluster manager on a node with a shutdown mode of **graceful with takeover**.

2. A failure of the operating system on the active control workstation

If the AIX operating system fails, the cluster manager on the inactive control workstation will detect this by observing missed heartbeats. The cluster manager then declares the crashed control workstation dead and performs a takeover of the shared resources.

3. A failure of an external disk adapter that is not backed up

If HACWS is configured with only one external disk adapter per control workstation, you need to write an Error Notification Object for the failure of the disk adapter to do a graceful shutdown with takeover. The other control workstation will then take over the shared resources.

4. A service network failure occurs on the active control workstation and the inactive control workstation is available to take over

A failure of a service adapter on the active control workstation, without a standby adapter to take over, will cause some SP node subsystems to perceive that the control workstation has failed and the system will quickly become unstable. To prevent such system instability, a service network failure on the active control workstation is promoted to a node failure, causing a failover to the inactive control workstation.

Failures that Do Not Cause a Failover of the Control Workstation

The following conditions will not cause a failover of a control workstation:

1. A hung control workstation with network adapters that still heartbeat

In the case where the application software hangs and the network adapters being monitored via HACMP still respond to heartbeats from the inactive control workstation, HACMP will not detect the hang and will not fail over.

2. A service network failure occurs on the active control workstation and the inactive control workstation is not available to take over

If the inactive control workstation is not available to take over, then a service network failure on the active control workstation will not be promoted to a node failure because no failover can occur.

General Guidelines for When to Do Configuration Changes

IBM suggests that you do configuration changes on the SP system only when the primary control workstation is the active control workstation. Most configuration changes are allowed when the backup control workstation is the active control workstation. However, if changes are made when the backup control workstation is active, some configuration files will need to be updated on the primary control workstation. When changes are made on the primary control workstation, some configuration files will need to be updated on the backup control workstation. If configuration changes are done on the backup control workstation, the updates go the opposite direction (backup to primary instead of primary to backup). It is easier to manage the system when the updates go only in one direction.

You can decide to what extent you do configuration changes on the backup control workstation. Because only two machines are involved, the bidirectional updates may not be a problem.

The following table lists tasks that may be performed in an HACWS configuration and whether or not it is possible to update configuration files on the active backup control workstation. A 'No' in the 'Backup CWS Active' column means that the file cannot be updated on the active backup control workstation. In that case, you will have to update the file on the primary control workstation when it is active again.

Table 15. High Availability Control Workstation Configuration and Task Summary

Task	Primary CWS Active	Backup CWS Active
Update passwords	Yes	No
Add or change users	Yes	No
Change Kerberos keys	Yes	No
Install a node	Yes	Yes
Change or add system partitions	Yes	Yes
Add nodes to the system	Yes	No
Hardware monitoring	Yes	Yes
Reboot nodes	Yes	Yes
Run diagnostics	Yes	Yes
Shut down and restart system	Yes	Yes
Run parallel jobs	Yes	Yes
Update file collections	Yes	Yes
Accounting	Yes	Yes
Change site environment information	Yes	No

The following table lists the tasks whose support is different for SP-attached servers if the active backup control workstation has not been enabled to the servers:

Table 16. High Availability Control Workstation and Task Summary (SP-attached servers)

Task	Backup CWS Active
Install an SP-attached server	No
Reboot SP-attached servers	No
Shut down and restart the system	Yes

Note that although you can shut down and restart the system with the SP-attached servers present, the attached servers will not be included in the process.

Managing SP Subsystems for High Availability Control Workstation Function

This section presents information on the administrative tasks you must perform to ensure that the various SP subsystems will fail over to the backup control workstation.

Starting Up and Shutting Down the SP System

The sequence files need to be kept in sync between the primary and backup control workstations in order for the **cshut** command to work on the backup control workstation. The following files need to be synchronized:

- **/etc/cstartSeq**
- **/etc/shutSeq**

- `/etc/subsysSeq`

Parallel Management Commands

You can use parallel management commands with the backup control workstation as a target, even when the backup control workstation is inactive, as long as the host name of the backup control workstation is listed in the working collective or host list.

Managing File Collections

The inactive control workstation acts as a file collection client (**supper** is run using **cron**). This keeps all the file collection files and other default file collections in sync between the primary and backup control workstations. If a configuration or file collection master change is done on the active control workstation, you should perform a **supper update** for the particular file collection. Note that you can do a **supper update** for all the file collections. If the update is not performed after the change and a failover occurs, the updates will not be on the new active control workstation and such updates will not be available to the SP nodes.

If you create scan files on the active control workstation, you need to make sure that the scan files get created on the takeover control workstation after a control workstation failover. Otherwise, the scan files may not exist or may be out-of-date on the takeover control workstation, and file collections will not function as expected. To automate the creation of scan files after a control workstation failover, create a file named **start_server.post_post_event** with execute permission for the root user in the `/var/adm/hacws/events` directory on both the primary and the backup control workstation with contents similar to the following:

```
#!/bin/ksh -x
exec /var/sysman/supper scan sup.admin user.admin power_system
```

If you have additional file collections that you are using scan files with, you will need to add those to the list in the previous supper command. For more information on using scan files, refer to “Verifying File Collections Using scan” on page 51.

Managing User Accounts

IBM suggests that HACWS configurations use NIS for user management when large numbers of users need to be managed on the SP system. NIS enforces a master-slave relationship for password databases and is easier to manage than file collections for user management in an HACWS configuration.

Using NIS with HACWS

If you are using NIS in an HACWS configuration and the NIS master database is the primary or backup control workstation, there must be at least one slave database within the NIS domain. For high availability of an SP system, IBM suggests that all the nodes where users log in be NIS slave database nodes.

Using File Collections with HACWS

If you use file collections for user management, the **/etc/passwd** and associated files should be on the primary control workstation. Once a failover occurs and the backup control workstation is the active control workstation, the actual master **passwd** files are no longer available to the backup control workstation. For this reason changing your passwords, gecos information, and login shell is not allowed when the backup control workstation is active. (This is similar to the restriction with NIS when the NIS master is down.) The backup control workstation, when inactive, will act as a file collection client (similar to an SP node) and update its **passwd** files from the primary control workstation when it is active.

Controlling User Login

If you are using login control on your SP system, the **spacs_data** must be kept in sync between the primary and backup control workstations. The **block_usr_sample** script and any derivatives of it need to be kept in sync between the primary and backup control workstations.

Managing SP Resources

When a failover or reintegration occurs, running parallel jobs should not be affected. LoadLeveler does not have a dependency upon the control workstation as long as the central manager is not being run on the control workstation.

Managing Time Synchronization

Both control workstations must use the same source for time (for example, timemaster, Internet, or consensus). NTP should be running on both the inactive and active control workstations.

Managing an Automounter

Include the backup control workstation in the distribution of automounter configuration and map files.

Managing Mail Service

If the primary control workstation is the mail hub, you should use NFS to export the **/var/spool/mail** file system to the backup control workstation so that mail files for nodes are put in the same file system. When a failover occurs you should have the **/var/spool/mail** locally mounted on the backup control workstation and the primary control workstation. Use NFS to mount the file on the control workstation when it is the inactive control workstation. The **/var/spool/mail** file system must be an externally mounted file system.

Accounting

If accounting was enabled in the Site Environment Information SMIT panel and the node number **0** was selected as the Accounting Master, you need to make a directory change. Create a symbolic link from both the primary and backup control workstations to put the **cacct** directory in the **/spdata/sys1** file system.

Do the following on both the primary and backup control workstations after accounting has been enabled:

1. Enter:

```
ln -s /spdata/sys1/cacct /var/adm/cacct
```


This creates a symbolic link between the **cacct** directory so that the data in the **rootvg** is automatically copied into the external file system when **crunacct** is run using **cron**. This allows accounting reports over a period of time to have no indication of which control workstation was active.

On an inactive control workstation the **crunacct** and **cmonacct** scripts will fail because the symbolic link for the **cacct** directory on the inactive control workstation will not have a target directory.

2. To keep those scripts from failing, change the invocation in the **crontab** file to run a wrapper shell script that determines if the control workstation is the inactive control workstation. If the control workstation is the inactive control workstation, the script should exit without running the **cmonacct** or **crunacct** scripts. The **lshacws** command should be used for this. Alternatively, you may choose to let the script fail. The following is a sample script:

```
#!/bin/ksh
# This is the state of the control workstation taking
# the string output of the lshacws command.
# It takes the name either crunacct or cmonacct
# as an argument.

HACWS_STATE=`lshacws`

# Check to see if we are on an INACTIVE control workstation.
# If we are, exit quietly, if not, run the argument passed.

if [[ $HACWS_STATE = 1 || $HACWS_STATE = 16 ]] ; then
    exit 0
else

# Go run the command
$1
fi
exit 0
```

Kerberos V4 Security Features of the SP System

In an HACWS configuration the primary authentication service is **not** duplicated on the backup control workstation. If the primary control workstation is either a primary or secondary authentication server, then the backup control workstation must be a secondary authentication server. The backup control workstation cannot be a primary authentication server.

When the primary control workstation is down, and it is the primary authentication server, no updates to the authentication database can be made.

Kerberos Configuration Files

The backup control workstation needs to be added to the **.klogin** files on all the nodes and the control workstation. This **.klogin** file also needs to be kept in sync on the backup control workstation.

When control workstation services move back and forth between the two control workstations, the Kerberos service keys need to remain the same. To accomplish this, the following commands were issued on the backup control workstation when HACWS was initially configured:

```
$ /usr/lpp/ssp/rcmd/bin/rcp -p primary_name:/etc/krb-srvtab /etc/krb-srvtab.primary
$ cp -p /etc/krb-srvtab /etc/krb-srvtab.backup
$ cat /etc/krb-srvtab.primary >>/etc/krb-srvtab
```

Repeat this procedure whenever you change Kerberos service keys on either of the two control workstations.

Sysctl

When Sysctl is installed the configuration file **/etc/sysctl.conf** is created and updated on the backup control workstation. If you update this file or use a different configuration file, the file needs to be kept in sync between the primary and backup control workstation.

The Sysctl ACL files must also be kept in sync between the primary and backup control workstations. This file is shipped as **/etc/sysctl.acl**. If you add an ACL record to the **/etc/sysctl.acl** file, the file should be kept in sync between the primary and backup control workstations.

When the backup control workstation is the inactive control workstation, Sysctl commands can be used because Sysctl is installed and running on the inactive control workstation.

Using a Switch

To be sure the switch is connected to the backup control workstation for the SP Switch environment, refer to the following section.

The SP Switch

When an SP Switch event occurs during a control workstation failover, the communication subsystem continues to handle switch faults, primary node failures, and primary backup node failures. System data about the switch is inaccessible during the control workstation failover and can become out-of-date. When the backup control workstation assumes the duties of the active control workstation, it must determine if a double failure has occurred (a switch event that occurred during a control workstation failover).

Perform the following to determine if a switch event has occurred and to resynchronize the communication subsystem with the system data on the new control workstation:

1. To determine if a primary node takeover (failure) has occurred, check all nodes in each system partition for a file named **act.top.1** in the **/var/adm/SPlogs/css** directory that contain a timestamp during the control workstation failover. If you locate an **act.top.1** file containing a timestamp during the control workstation failover:
 - a. Run **/usr/lpp/ssp/css/rc.switch** on all the nodes in each system partition that has a node with an **act.top.1** file.
 - b. Run **Estart** for each system partition that has a node with an **act.top.1** file.
2. To determine if a primary backup node failed or a switch fault occurred, examine the **flt** file in the **/var/adm/SPlogs/css** directory on the primary node for entries with a timestamp during the control workstation failover. If you locate entries that contain a timestamp during the control workstation failover, run **Estart** for each system partition having entries on its primary node.

IBM suggests that you use an Error Notification Object for switch faults, primary node takeovers, and control workstation failures to send mail or a message to alert you about such double failures. You can then examine the communication subsystem as described previously. Because the timeframe and probability for these double failures are very small, IBM does not suggest that you run **rc.switch** or **Estart** whenever a control workstation failover occurs.

Managing System Partitions

System partition configuration changes must be done from the active control workstation. Such changes may be done on either the primary or backup control workstation, depending on which one is active. IBM suggests that configuration changes be done from the primary control workstation, but it is not required.

When you configure HACWS you are instructed to put any IP address aliases that are required for partitioning or any other reason in a script called **/spdata/sys1/hacws/rc.syspar_aliases**. This script will be invoked by the HACWS event scripts at the correct times to enable system partitioning.

Managing Extension Nodes

The SP SNMP Manager that runs on the control workstation transfers dependent node configuration information to the SNMP Agent running on a dependent node. This SP SNMP Manager support will transfer to the active control workstation.

Using SP Perspectives

If you have installed **ssp.clients** or **ssp.gui** on a RS/6000 other than the primary or backup control workstations, carefully consider how you will monitor the SP system from other than the active control workstation. If you execute the **perspectives** commands without logging into the active control workstation and the active control workstation does a failover, this will not be represented on the SP Perspectives GUI until you take another action. Once you take another action, the **perspectives** commands will exit. You will have to reexecute the **perspectives** commands and reconnect to the currently active control workstation. This occurs because when you are monitoring a TCP/IP connection and the host fails, the client side of that connection does not know about the failure of the connection until a packet is sent from the client side of the connection. Since the users of the **perspectives** commands who are off the control workstation should be well known, an Error Notification Object can be set up to send mail or a message to the list of well known users to inform them that the control workstation has performed a failover. Such a notification list for the users that monitor the SP system is a good idea in any event. After a failover of a control workstation (or any other major recovery action) an administrator should be notified. Several options for such notification exist:

- Error Notification Objects
- NetView for AIX control desk alerts
- Trouble Ticket for AIX problem incidents
- HACMP Notify script

Viewing which Control Workstation is Active

With previous releases of PSSP, you could view which RS-232 tail was active using the Frame Environment Layout display of the SP System Monitor GUI. This information is now available using the hardware perspective. To display this information:

1. Ensure that the Frames/Switches pane is viewable. You may need to add a Frames/Switches pane first. Then, click in the pane to make it active and select a frame from the pane.
2. Select Actions → View and Modify Properties from the menu bar or just click on the Notebook icon.
3. Select the Frame Status page in the Frame notebook to view the information.

Customizing HACMP

To customize HACMP, follow the basic steps for configuring a two-node rotating configuration with nonconcurrent access in the *AIX High Availability Cluster Multi-Processing Installation Guide*.

Configuring Resources

When you get to the configuring resources step in the HACMP installation procedure, you need to define a resource group name. For an HACWS configuration, you **must** name the resource group **hacws_group1**. To define event scripts to handle failovers for HACWS configuration, the name of the resource group needs to be known. The HACWS configuration will not fail over if the **hacws_group1** resource group name is not used.

HACMP Event Processing

The HACWS software supplies HACMP pre- and post-event scripts for all HACMP events. These supplied pre- and post-event scripts are used to customize the HACMP configuration for HACWS. To permit you to add your own pre- and post-event scripts, HACWS software provides for pre- and post-event scripts in each pre- and post-event script that is supplied by HACWS. The following flow describes when the pre- and post-event scripts are called.

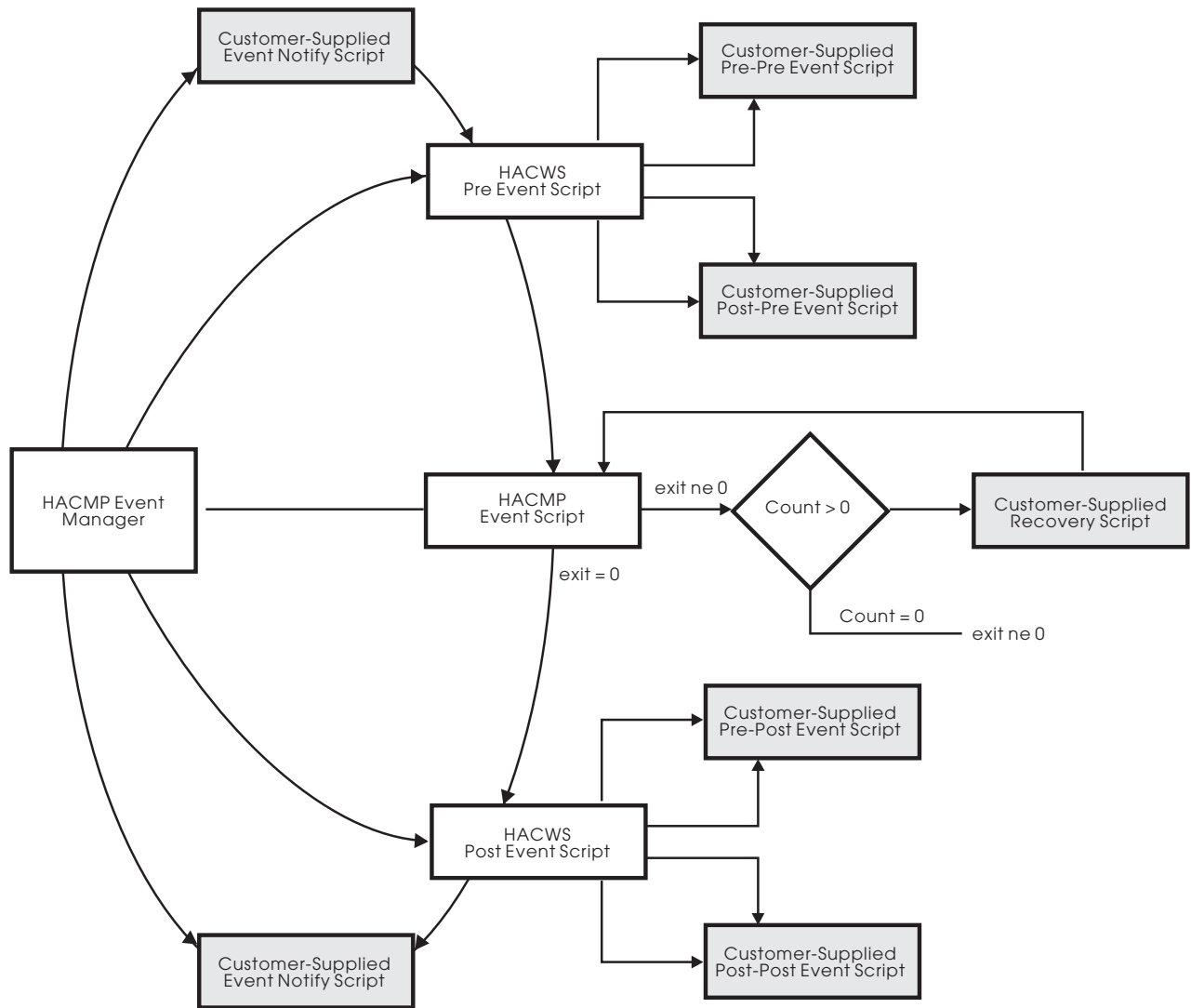


Figure 40. HACWS Pre- and Post-Events

The pre- and post-event scripts supplied with HACWS function are located in the **/usr/sbin/hacws** directory. These scripts are defined to HACMP with the **spcw_addevents** command. Do not modify these scripts. These scripts are generic and will call the scripts supplied with HACWS (found in **/usr/sbin/hacws/events**) on a per event basis. Do not modify the per event scripts.

You may provide additional pre- and post-event scripts to be called by the pre- and post-event scripts supplied with HACWS. The additional scripts you provide must be located in the **/var/adm/hacws/events** directory.

Name the scripts you supply following this naming convention:

event_name.pre_pre_event

event_name is the HACMP event name and **pre_pre_event** is the name for a pre-event script for an HACMP pre-event script. This naming convention extends to having **post_pre**, **pre_post** and **post_post** event scripts.

To enable the execution of event scripts, place the scripts in the proper directories and turn on the execution mode bits using the **chmod** command.

Verifying Frame Supervisor Cables

In an HACWS configuration the tty assignments for each frame **must** be identical on both the primary and backup control workstations. The new frame supervisor cards (Type 22 and 20) used for HACWS configurations have flash memory on the cards. This flash memory is used to write a unique ID so the frame can be identified. With previous cards the order in which the frames were cabled determined the frame numbers in the configuration. There was no method to check that the frame was actually still the same frame number.

If a frame number on the card does not match the frame number in the SDR, the hardware monitor will not bring the frame online. An error log entry is written to the AIX error log and BSD syslog indicating this. Note that in most cases of miscabling, two frames will not be brought online. An error log entry will be written for each frame. You have to correct the cabling to repair the situation.

When the **hardmon** daemon initializes it verifies that the frame supervisor cables match the information in the SDR. To check that the cables are correct from an inactive control workstation (there must be an active control workstation because the SDR information needs to be compared):

1. Enter **stopsrc -s hardmon** to ensure there is no hardmon daemon running on the inactive control workstation.
2. Issue the **hardmon -B** command from the inactive control workstation.

Any cable problems will be written to the AIX error log.

When an SP system is reconfigured and frames moved around and reassigned, new frame IDs need to be written to existing frames. Note that the system must be completely reinstalled so node numbers and system images are correct. This action should rarely, if ever, be taken.

The following example lists the steps to change the frame numbers in the flash memory of the card. In this example frame number 4 is now frame number 6 and frame number 6 is now frame number 4. A complete reinstall of the SP system is also being done. Note that it would be far easier to make the cables match and not swap frames around. Swapping frames should be done only in extreme cases.

1. Determine the tty number of the cable that is connected to the frames.
2. Verify the SDR information about the frame.

Ensure that the SDR information about the frame and tty number match how the frame is cabled. Use the **SDRGetObjects Frame** command to see which frame number matches with which tty number. Assuming that the SDR information is correct, proceed to the next step.

If the tty number in the SDR is incorrect, use the **spframe** command to make frame number 6 use the correct tty special device name. For example, if you want to make frame 6 use **/dev/tty6** (assuming no other frame is using **/dev/tty6**), enter **spframe 6 1 /dev/tty6**.

3. Set the frame ID in the flash memory.

Enter **hcmands -G setid 4,6:0** to set the frame ID to the values in the SDR frame class for frames 4 and 6.

Chapter 23. The Topology Services Subsystem

This chapter introduces you to the Topology Services subsystem. It includes information about the components of the subsystem, its configuration, other components that depend on it, and how it operates. This chapter also discusses the relationship of the Topology Services subsystem to the other PSSP subsystems. Finally, it describes a procedure you can use to check the status of the subsystem.

After reading this chapter, you will be able to manage the Topology Services subsystem and, if necessary, perform problem determination.

Introducing Topology Services

Topology Services is a distributed subsystem of the IBM RS/6000 Cluster Technology (RSCT) software on RS/6000 systems. The RSCT software provides a set of services that support high availability on your SP system. Other services in the RSCT software are the Event Management and Group Services distributed subsystems. These three distributed subsystems operate within a domain. A domain is a set of RS/6000 machines upon which the RSCT components execute and, exclusively of other machines, provide their services. On an SP system, a domain is a system partition. Note that a machine might be in more than one RSCT domain; the control workstation is a member of each system partition, and, therefore, a member of each RSCT domain. When a machine is a member of more than one domain, there is an executing copy of each RSCT component per domain.

Topology Services provides other high availability subsystems with network adapter status, node connectivity information, and a reliable messaging service. The adapter status and node connectivity information is provided to the Group Services subsystem upon request, Group Services then makes it available to its client subsystems. The Reliable Messaging Service, which takes advantage of node connectivity information to reliably deliver a message to a destination node, is available to the other high availability subsystems.

This adapter status and node connectivity information is discovered by an instance of the subsystem on one node, participating in concert with instances of the subsystem on other nodes, to form a ring of cooperating subsystem instances. This ring is known as a heartbeat ring, because each node sends a heartbeat message to one of its neighbors and expects to receive a heartbeat from its other neighbor. Actually each subsystem instance can form multiple rings, one for each network it is monitoring. Usually, each subsystem monitors two rings; the SP ethernet and the SP switch. This system of heartbeat messages enables each member to monitor one of its neighbors and to report to the heartbeat ring leader, called the Group Leader, if it stops responding. The Group Leader, in turn, forms a new heartbeat ring based on such reports and requests for new adapters to join the membership. Every time a new group is formed, it lists which adapters are present and which adapters are absent, making up the adapter status notification that is sent to Group Services.

In addition to the heartbeat messages, connectivity messages are sent around all rings. Connectivity messages for each ring will forward its messages to other rings, so that all nodes can construct a connectivity graph. It is this graph that

determines node connectivity and defines a route that Reliable Messaging would use to send a message between any pair of nodes that have connectivity.

For more detail on maintaining the heartbeat ring and determining node connectivity, see "Topology Services Components."

Topology Services Components

The Topology Services subsystem consists of the following components:

Topology Services Daemon

The central component of the Topology Services subsystem.

Port numbers

TCP/IP port numbers that the Topology Services subsystem uses for daemon-to-daemon communications. The Topology Services subsystem also uses UNIX domain sockets for server-to-client communication.

Control script

A shell script that is used to add, start, stop, and delete the Topology Services subsystem, which operates under the System Resource Controller (SRC) component.

Start-up script

A shell script that is used to obtain the configuration from the System Data Repository (SDR) and start the Topology Services Daemon. This script is invoked by the SRC component.

Files and directories

Various files and directories that are used by the Topology Services subsystem to maintain run-time data.

The sections that follow contain more details about each of these components.

The Topology Services Daemon (**hatsd**)

The Topology Services daemon is contained in the executable file **/usr/sbin/rsct/bin/hatsd**. This daemon runs on each node of an SP system partition and on the control workstation. If there is more than one system partition, then multiple daemons run on the control workstation, one for each system partition.

Note that the operational domain of the Topology Services subsystem is an SP system partition. The control workstation is considered to be part of each domain. Unless otherwise stated, a reference to "the Topology Services subsystem" is a reference to the Topology Services subsystem in a single system partition.

When each daemon starts, it first reads its configuration from a file set up by the Startup script (**hats**). This file is called the machines list file, because it has all the machines (nodes) listed that are part of the configuration and the IP addresses for each adapter for each of the nodes in that configuration. From this file, the daemon knows the IP address and node number of all the potential heartbeat ring members.

The Topology Services subsystem directive is to form as large a heartbeat ring as possible. To form this ring, the daemon on one node must alert those on the other nodes of its presence by sending a *proclaim* message. According to a hierarchy defined by the Topology Services component, daemons can send a proclaim message only to IP addresses that are lower than its own and can accept a proclaim message only from an IP address higher than its own. Also, a daemon

only proclaims if it is the leader of a ring. When a daemon first starts up, it builds a heartbeat ring for every local adapter, containing only that local adapter. This is called a singleton group and this daemon is the leader in each one of these singleton groups.

To manage the changes in these groups, Topology Services defines the following roles for each group:

Group Leader	The daemon on the node with the local adapter that has the highest IP address in the group. The Group Leader proclaims, handles request for joins, handles death notifications, coordinates group membership changes, and sends connectivity information.
Crown Prince	The daemon on the node with the local adapter that has the second highest IP address in the group. This daemon can detect the death of the Group Leader and has the authority to become the leader of the group if that happens.
Mayor	A daemon on a node with a local adapter present in this group that has been picked by the Group Leader to broadcast a message to all the adapters in the group. When a daemon receives a message to broadcast, it is a mayor.
Generic	This is the daemon on any node with a local adapter in the heartbeat ring. The role of the Generic daemon is to monitor the heartbeat of the upstream neighbor and inform the Group Leader if the correct number of heartbeats have arrived.

Each one of these roles are dynamic, which means that every time a new heartbeat ring is formed, the roles of each member are evaluated and assigned.

In summary, Group Leaders send and receive proclaim messages. If the proclaim is from a leader with a higher IP address, then the group leader with the lower address replies with a join request. The higher address group leader forms a new group with all members from both groups. All members monitor their upper neighbor for heartbeats. If a sufficient number of heartbeats are missed, a message is sent to the Group Leader and the unresponsive adapter will be dropped from the group. Whenever there is a membership change, Group Services is notified if it asked to be.

The Group Leader also accumulates node connectivity information, constructs a connectivity graph, and routes connections from its node to every other node in the SP system partition. The group connectivity information is sent to all nodes so that they can update their graphs and also compute routes from their node to any other node. It is this traversal of the graph on each node that determines which node membership notification is provided to each node. Whenever the graph changes, routes are recalculated, and a list of nodes that have connectivity is generated and made available to Group Services.

Port Numbers and Sockets

The Topology Services subsystem uses several types of communications:

- UDP port numbers for intrapartition communications, that is, communications between Topology Services daemons within an SP system partition
- UNIX domain sockets for communication between Topology Services Clients and Topology Services daemon.

Intrapartition Port Numbers

For communication between Topology Services daemons within a system partition, the Topology Services subsystem uses a single UDP port number. This port number is recorded in the **Syspar_ports** SDR class. The class contains two attributes: subsystem and port. The class object whose subsystem attribute has a value of hats contains the port number for the Topology Services subsystem. Note that the **Syspar_ports** class is a partitioned SDR class, that is there is a set of objects for each system partition. When you configure the Topology Services subsystem for a particular system partition, you use the SDR data that corresponds to that system partition.

The Topology Services port number is stored in the SDR so that, when the Topology Services subsystem is configured on each node, the port number is retrieved from the SDR. This ensures that the same port number is used by all Topology Services daemons in the same system partition. Note that because a Topology Services daemon from each system partition runs on the control workstation, the Topology Services subsystem in each system partition must have a unique port number.

This intrapartition port number is also set in the **/etc/services** file, using the service name **hats.syspar_name**, where *syspar_name* is the name of the system partition. The **/etc/services** file is updated on all nodes in the system partition and on the control workstation. The Topology Services daemon obtains the port number from the **/etc/services** file during daemon initialization.

UNIX Domain Sockets

Unix domain sockets are used for communication between Topology Services clients (Group Services and Event Management), and the local Topology Services daemon. These are connection-oriented sockets. The socket name used to connect to the Topology Services daemon is **/var/ha/soc/hats/server_socket.syspar_name**, where *syspar_name* is the name of the system partition.

The Control Script (hatsctrl)

The Topology Services control script is contained in the executable file **/usr/sbin/rsct/bin/hatsctrl**. This script or command is normally invoked by the **syspar_ctrl** script, which provides an interface to all of the system-partition-sensitive subsystems.

If necessary, you can invoke the **hatsctrl** script directly from the command line. Before you invoke this command, ensure that the **SP_NAME** environment variable is set to the appropriate system partition name and is exported.

For more information about the **hatsctrl** and **syspar_ctrl** commands, see the book PSSP: Command and Technical Reference.

The purpose of the **hatsctrl** command is to add (configure) the Topology Services subsystem to a system partition. You can also use the command to remove the subsystem from a system partition, start the subsystem, stop the subsystem, and clean the subsystem from all system partitions.

Files and Directories

The Topology Services subsystem uses the following directories:

- **/var/ha/log**, for log files
- **/var/ha/run**, for Topology Services daemon current working directory
- **/var/ha/soc**, for socket files.

The /var/ha/log Directory (Log Files)

The **/var/ha/log** directory contains trace output from the Topology Services daemon.

On the control workstation and each node, the file is called **hats.DD.HHMMSS.syspar_name**, where:

- *DD* is the Day of the Month that this daemon was started.
- *HHMMSS* is the Hour, Minute, and Second that the daemon was started.
- *syspar_name* is the name of the system partition to which the node belongs.

The Topology Services daemon limits the log size to a configured number of lines (by default, 5,000 lines). When the limit is reached, the daemon appends the string **.bak** to the name of the current log file and begins a new log with the same original name. If a **.bak** version already exists, it is removed before the current log is renamed.

The /var/ha/run Directory (Daemon Working Files)

In the **/var/ha/run** directory, a directory named **hats.syspar_name** is created, where *syspar_name* is the system partition name. This directory is the current working directory for the Topology Services daemon. If the Topology Services daemon abnormally terminates, the core dump file is placed in this directory. Whenever the Topology Services daemon starts, it renames any core file to **core.DD.HHMMSS.syspar_name**, where:

- *DD* is the Day of the Month that the daemon associated with this core file was started.
- *HHMMSS* is the Hour, Minute, and Second that the daemon associated with this core file was started.
- *syspar_name* is the name of the system partition to which the node belongs.

Components on Which Topology Services Depends

The Topology Services subsystem depends on the following components:

System Resource Controller (SRC)

An AIX feature that can be used to define and control subsystems. The Topology Services subsystem is called **hats** on SP nodes and is called **hats.syspar_name** on the control workstation, where *syspar_name* is the name of a

system partition. The subsystem name is used with the SRC commands (for example, **startsrc** and **lssrc**).

System Data Repository (SDR)

A repository of SP system configuration information.

Configuring and Operating Topology Services

The following sections describe how the components of the Topology Services subsystem work together to provide topology services. Included are discussions of Topology Services:

- Configuring Topology Services
- Initializing Topology Services Daemon
- Operating Topology Services

Configuring Topology Services

RSCT is installed as part of the installation of the PSSP product. The Topology Services subsystem is contained in the **rsct.basic.rte** and **rsct.basic.sp** filesets. After the components are installed, you need to configure it for operation using the **hatsctrl** command, which is invoked by the **syspar_ctrl** command.

The **syspar_ctrl** command configures all of the system-partition-sensitive subsystems. The person who installs PSSP issues the **syspar_ctrl** command during installation of the control workstation. The **syspar_ctrl** command is executed automatically on the nodes when the nodes are installed. The **syspar_ctrl** command is also executed automatically when system partitions are created or destroyed. For more information on using this command, see PSSP Installation and Migration Guide and PSSP Command and Technical Reference.

The **hatsctrl** command provides a number of functions for controlling the operation of the Topology Services system. You can use it to:

- Add or configure the Topology Services subsystem
- Start the subsystem
- Stop the subsystem
- Delete or unconfigure the subsystem
- "Clean" all Topology Services subsystems
- Turn tracing of the Topology Services daemon on or off
- Refresh (read and dynamically reflect a updated configuration) the subsystem.

Except for the clean function, **hatsctrl** affects the Topology Services subsystem in the current system partition, that is, the system partition that is specified by the **SP_NAME** environment variable.

The SDR stores node and data information, as well as some tunable data. The class that holds this data is called the **TS_Config** class. This class holds information that controls some of the operation of Topology Services. The following is a list of the attributes in this class and a brief description of each:

Frequency	Controls how often Topology Services sends a heartbeat to its neighbors. The value is interpreted as the number of seconds between heartbeats. The minimum and default value is 1. On a system with a high amount of paging activity, this number should be kept as small as possible.
Sensitivity	Controls the number of missed heartbeat messages that will cause a Death in Family message to be sent to the Group Leader. Heartbeats are not considered missing until it has been twice the interval indicated by the Frequency attribute.
Run_FixedPri	Run the daemon with a fixed priority. Since Topology Services is a realtime application, there is a need to avoid scheduling conflicts. The number 1 indicates that the daemon is running with fixed priority, 0 indicates that it is not.
FixedPriValue	This is the actual fixed priority level that is used. The daemon will accept values greater or equal to 31. The default is 38.
Log_Length	This is the approximate number of lines that a log file can hold before it wraps.
Pinning	This controls the memory Pinning strategy. Text causes the daemon to attempt to pin Text pages, Data attempts to pin Data Pages, Process attempts to pin all pages, and None causes no pages to be pinned. The default is Text.

On RS/6000 systems with heavy and/or unusual load characteristics, it might be necessary to adjust the Frequency and Sensitivity settings. See “Operating Topology Services Daemon” on page 321 for more information.

Adding the Subsystem

If the **hatsctrl** command is running on the control workstation, the first step in the add function is to select a Topology Services daemon communications port number and save it in the **Syspar_ports** SDR class. This port number is then placed in the **/etc/services** file. If the **hatsctrl** command is running on a node, the port number is fetched from the SDR and placed in the **/etc/services** file. Port number are selected from the range 10000 through 10100.

The second step is to add the Topology Services daemon to the System Resource Controller (SRC) using the **mkssys** command. On the control workstation, the IP address of the system partition is an argument to the **hats** command in the SRC subsystem specification.

The third step is to add an entry in the **/etc/inittab** file so that the Topology Services daemon will be started during boot. However, if **hatsctrl** is running on a High Availability Control Workstation (HACWS), an entry is not made in the **/etc/inittab** file. Instead, HACWS manages the starting and stopping of the Topology Services daemon.

Note that if the **hatsctrl** add function terminates with an error, you can rerun the command after fixing the problem. The command takes into account any steps that already completed successfully.

Starting and Stopping the Subsystem

The start and stop functions of the **hatsctrl** command run the **startsrc** and **stopsrc** commands, respectively. However, **hatsctrl** automatically specifies the subsystem argument to these SRC commands.

Deleting the Subsystem

The delete function of the **hatsctrl** command removes the subsystem from the SRC, removes the entry from **/etc/inittab**, and removes the Topology Services daemon communications port number from **/etc/services**. It does not remove anything from the SDR, because the Topology Services subsystem might still be configured on other nodes in the system partition.

Cleaning the Subsystem

The clean function of the **hatsctrl** command performs the same function as the delete function, except in all system partitions.

The clean function does not remove anything from the SDR. This function is provided to support restoring the system to a known state, where the known state is the (possibly restored copy of the) SDR database.

Tracing the Subsystem

The tracing function of the **hatsctrl** command is provided to supply additional problem determination information when it is requested by the IBM Support Center. Normally, you should not turn tracing on because it might slightly degrade Topology Services subsystem performance and can consume large amounts of disk space in the **/var** file system.

Initializing Topology Services Daemon

Normally, the Topology Services daemon is started by an entry in the **/etc/inittab** file using the **startsrc** command. If necessary, you can start the Topology Services daemon using the **hatsctrl** command or the **startsrc** command directly. The first part of initialization is done by the startup script, **hats**. During this initialization, the startup program performs the following steps:

1. The startup program obtains the number of the node on which it is running using the **/usr/lpp/ssp/install/bin/node_number** command. Node 0 is the control workstation.
2. The startup program obtains the name of the system partition from the **Syspar** SDR class. (Recall that one instance of the Topology Services daemon runs on the control workstation for each system partition to which the Topology Services subsystem was added.)
3. If running on the control workstation, the startup program builds a condensed version of the node and adapter configuration called the Machines List file. The name of the Machines List file is **machines.lst** and can be found in the Topology Services current working directory, **/var/ha/run/hats.syspar_name**, where **syspar_name** is the name of the system partition.

The Machines List file also has tunable information from the **TS_Config** SDR class. The startup script on the control workstation then compares this file with the one stored in the SDR. If they are different, then the one in the SDR is updated to reflect any changes. If this is not the control workstation, then the script retrieves the file from the SDR.

4. The startup program performs file maintenance in the log directory and current working directory to remove the oldest log and rename any core files that might have been generated.
5. The startup program executes the Topology Services daemon **hatsd**.

The daemon then continues the initialization with the following steps.

1. Read the current Machines List file and initialize internal data structures.
2. Initialize daemon-to-daemon communication, as well as client communication.
3. For each local adapter defined in the configuration, form a membership consisting of only the local adapter.

The daemon is now in its initialized state and ready to communicate with Topology Services daemons on other nodes. The intent is to expand each singleton membership group formed during initialization to contain as many members as possible. Each adapter has an offset associated with it. Only other adapter membership groups with the same offset can join together to form a larger membership group. Eventually, as long as all the adapters configured for a particular network can communicate with each other, there will exist a single group to which all adapters belong.

Merging All Adapters into a Single Group

Initially the subsystem starts out as N singleton groups, one for each node. Each of those daemons is a Group Leader of those singleton groups and knows which other adapters could join the group by the configuration information. The next step is to begin proclaiming to subordinate nodes.

The proclaim logic tries to find members as efficiently as possible. For the first 3 proclaim cycles, daemons proclaim to only their own subnet, and if the subnet is broadcast-capable, that message is broadcast. The result of this is that given the previous assumption that all daemons started out as singletons, this would evolve into M groups, where M is the number of subnets that span this heartbeat ring. On the fourth proclaim cycle, those M Group Leaders send proclaims to adapters that are outside of their local subnet. This will cause a merging of groups into larger and larger groups until they have coalesced into a single group.

From the time the groups were formed as singletons until they reach a stabilization point, the groups are considered unstable. The stabilization point is reached when a heartbeat ring has no group changes for the interval of 10 times the heartbeat send interval. Up to that point, the proclaim continues on a 4 cycle operation, where 3 cycles only proclaim to the local subnets, and one cycle proclaims to adapters not contained on the local subnet. After the heartbeat ring has reached stability, proclaim messages go out to all adapters not currently in the group regardless of the subnet to which they belong.

Operating Topology Services Daemon

Normal operation of the Topology Services subsystem does not require administrative intervention. The subsystem is designed to recover from temporary failures, such as node failures or failures of individual Topology Services daemons. Topology Services also provides indications of higher level system failures. However, there are some operational characteristics of interest to system administrators and after adding or removing nodes or adapters, you might need to refresh the subsystem.

Defaults and Limitations

The maximum node number allowed is 4097. The maximum number of networks it can monitor is 16. The default number of networks monitored is 2, the SP Ethernet and the SP Switch.

Topology Services is meant to be sensitive to network response and this sensitivity is tunable. However, other conditions can degrade the ability of Topology Services to accurately report on adapter or node membership. One such condition is the failure of AIX to schedule the daemon process in a timely manner. This can cause daemons to be late in sending their heartbeats by a significant amount. This can happen because an interrupt rate is too high, the rate of paging activity is too high, or there are other problems. Whenever these conditions exist, analyze the problem carefully to fully understand it.

Since Topology Services is a realtime process, do not intentionally subvert its use of the CPU because you can cause false indications.

Prior to AIX 4.2.1, Topology Services required you to set the network option **nonlocsrcroute** to 1 to enable IP source routing (the default is 0). AIX 4.2.1 and later releases now include network options for IP source routing as follows:

- **ipsrcroutesend** (default is 1)
- **ipsrcrouterrecv** (default is 0)
- **ipsrcrouteforward** (default is 1)

Topology Services sets all four of these options to 1 so that the reliable message, feature which utilizes IP source routing, will continue to work. Disabling any of these network options can prevent the reliable message feature from working properly.

Refreshing the Topology Services Daemon

If your system configuration changes, such as by adding or removing nodes or adapters, you need to refresh the Topology Services subsystem before it can recognize the new configuration. To refresh the subsystem, issue **hatsctrl -r** from the control workstation. Topology Services will rebuild its configuration and verify its validity. Validity checks verify that the IP addresses for the adapters have not changed since the previous configuration. If IP addresses were changed, then all the daemons are terminated and SRC will restart them. If the system configuration changes include only additions and deletions, then Topology Services will continue to operate with the new information.

Note that if there are nodes in the partition that are unreachable with Topology Services active, they will not be refreshed. Also, if the connectivity problem is resolved such that Topology Services on that node is not restarted, the node refreshes itself to remove the old configuration. Otherwise, it will not acknowledge nodes or adapters that are part of the configuration, but not in the old copy of the configuration.

For specific tuning information regarding Topology Services refer to <http://www.rs6000.ibm.com/support/sp/>.

Topology Services Procedures

Normally, the Topology Services subsystem runs itself without requiring administrator intervention. On occasion, you might need to check the status of the subsystem.

Displaying the Status of the Topology Services Daemon

You can display the operational status of the Topology Services daemon by issuing the **lssrc** command. Topology Services monitors the SP Ethernet and the SP Switch networks, however, only the SP Ethernet is monitored from the control workstation. To see the status of both networks you need to run the command on a node that is up on the switch.

On the control workstation, enter:

```
lssrc -ls hats.syspar_name
```

where *syspar_name* is the name of the system partition.

On a node, enter:

```
lssrc -ls hats
```

In response, the **lssrc** command writes the status information to the standard output. The information includes:

- The information provided by the **lssrc -s hats** command (short form)
- Two lines for each Network for which this node has an adapter and includes the following information:
 - The network name.
 - The network index.
 - The number of defined members, number of adapters that the configuration reported existing for this network.
 - The number of members, number of adapters currently in the membership group.
 - The state of the membership group, denoted by S (Stable), U (Unstable), or D (Disabled).
 - Adapter ID, the address and instance number for the local adapter in this membership group.
 - Group ID, the address and instance number of the membership group. The address of the membership group is also the address of the group leader.
- HB Interval, which corresponds to the **Frequency** attribute in the SDR **TS_Config** class. This exists both on a per network basis and a default value which could be different
- HB Sensitivity, which corresponds to the **Sensitivity** attribute in the SDR **TS_Config** class. This exists both on a per network basis and a default value which could be different
- The number of clients connected and the client process IDs
- Configuration Instance, the Instance number of the Machines List file

- The address of the Control Workstation

The following is an example of the output from the **lssrc -ls hats** on a node for the k21s system partition:

```
Subsystem      Group      PID      Status
  hats        hats          8002    active
Network Name   Indx  Defd   Mbrs   St   Adapter ID   Group ID
SPether        [ 0]    10     10    S   9.114.61.195 9.114.61.197
SPether        [ 0]                                0x3574061b 0x3574065f
HB Interval = 1 secs, Sensitivity = 4 missed beats
SPswitch       [ 1]     9      9     S   9.114.61.139 9.114.61.141
SPswitch       [ 1]                                0x3574061c 0x35740660
HB Interval = 1 secs. Sensitivity = 4 missed beats
2 locally connected Clients with PIDs:
haemd( 16554 ) hagsd( 12910 )
Configuration Instance = 895791387
Default: HB Interval = 1 secs, Sensitivity = 4 missed beats
CWS = 9.114.61.125
```

The following is an example of the output from the **lssrc -ls hats.k21s** command on the control workstation:

```
Subsystem      Group      PID      Status
 hats.k21s    hats          27716   active
Network Name   Indx  Defd   Mbrs   St   Adapter ID   Group ID
SPether        [ 0]    10     10    S   9.114.61.125 9.114.61.197
SPether        [ 0]                                0x357402b4 0x3574065f
HB Interval = 1 secs. Sensitivity = 4 missed beats
2 locally connected Clients with PIDs:
haemd( 16160 ) hagsd( 20474 )
Configuration Instance = 895791387
Default: HB Interval = 1 secs. Sensitivity = 4 missed beats
CWS = 9.114.61.125
```

The two networks being monitored are named SPether and SPswitch. The SPether has 10 defined adapters and all 10 adapters are members of the group. The Group is in the stable state and the addresses of the local adapter and the Group Leader are displayed. Likewise, in the SPswitch Network, there are 9 defined adapters and all 9 adapters are members in the group. Both networks have the same tunables and they are the same as the default. Both the node and the control workstation have two clients and the Process IDs are shown. Those are actually the Process IDs for the Group Services daemon and the Event Management Daemon.

Chapter 24. The Group Services Subsystem

This chapter introduces you to the Group Services subsystem. It includes information about the component of the subsystem, the configuration, other components on which it depends, and how it operates. It also discusses the relationship of the Group Services subsystem to the other high availability subsystems in PSSP. Finally, it describes a procedure you can use to check the status of the subsystem.

After reading this chapter, you will be able to manage the Group Services subsystem and, if necessary, perform problem determination.

For more information, see the book *RS/6000 SP High Availability Infrastructure*.

Introducing Group Services

Group Services is a distributed subsystem of the IBM RS/6000 Cluster Technology (RSCT) software on RS/6000 systems. The RSCT software provides a set of services to PSSP that support high availability on your SP system. Other services in the RSCT software are the Event Management and Topology Services distributed subsystems. These three distributed subsystems operate within a domain. A domain is a set of RS/6000 machines upon which the RSCT components execute and, exclusively of other machines, provide their services. On the an SP system, a domain is a system partition. Note that a machine might be in more than one RSCT domain; the control workstation is a member of each system partition, and, therefore, a member of each RSCT domain. When a machine is a member of more than one domain, there is an executing copy of each RSCT component per domain.

The function of the Group Services subsystem is to provide other subsystems with a distributed coordination and synchronization service. These other subsystems that depend upon Group Services are called *client subsystems*. Each client subsystem forms one or more *groups* by having its processes connect to the Group Services subsystem and use the various Group Services interfaces. A process of a client subsystem is called a *GS client*. For example, Event Management is a Group Services client subsystem. The Event Manager daemon on each node ia a GS client.

A group consists of two pieces of information:

- The list of processes that have joined the group, called the *group membership list*
- A client-specified *group state value*.

Group Services guarantees that all processes that are joined to a group see the same values for the group information, and that they see all changes to the group information in the same order. In addition, the processes may initiate changes to the group information via *protocols* that are controlled by Group Services.

A GS client that has joined a group is called a *provider*. A GS client that wishes only to monitor a group, without being able to initiate changes in the group, is called a *subscriber*.

Once a GS client has initialized its connection to Group Services, it can join a group and become a provider. All other GS clients that have already joined the group (those that have already become providers) are told as part of a join protocol about the new providers that wish to join. The existing providers can either accept new joiners unconditionally (by establishing a one-phase join protocol) or vote on the protocol (by establishing an n-phase protocol). During a vote, they can choose to approve the protocol and accept the new providers into the group, or reject the protocol and refuse to allow the new providers to join.

Group Services monitors the status of all the processes that are joined to a group. If either the process or the node on which a process is executing fails, Group Services initiates a failure protocol that informs the remaining providers in the group that one or more providers have been lost.

Join and failure protocols are used to modify the membership list of the group. Any provider in the group may also propose protocols to modify the state value of the group. All protocols are either unconditional (one-phase) protocols, which are automatically approved and not voted on, or conditional (n-phase) protocols, which are voted on by the providers.

During each phase of an n-phase protocol, each provider can take application-specific action and *must* vote to approve, reject, or continue the protocol. The protocol completes when it is either approved (the proposed changes become established in the group), or rejected (the proposed changes are dropped).

For more conceptual information about the Group Services subsystem, see *PSSP Group Services Programming Guide and Reference*.

Group Services Components

The Group Services subsystem consists of the following components:

Group Services daemon

The central component of the Group Services subsystem.

Group Services API (GSAPI)

The application programming interface that GS clients use to obtain the services of the Group Services subsystem.

Port numbers

TCP/IP port numbers that the Group Services subsystem uses for communications. The Group Services subsystem also uses Unix domain sockets.

Control script

A shell script that is used to add, start, stop, and delete the Group Services subsystem, which operates under control of the System Resource Controller (SRC) component of AIX.

Files and directories

Various files and directories that are used by the Group Services subsystem to maintain run-time data.

The sections that follow contain more details about each of these components.

The Group Services Daemon (hagsd)

The Group Services daemon is contained in the executable file **/usr/sbin/rsct/bin/hagsd**. This daemon runs on each node of a system partition and on the control workstation. If there is more than one system partition, then multiple daemons run on the control workstation, one per system partition.

Note that the operational domain of the Group Services subsystem on an SP system is a system partition. The control workstation is considered to be part of each domain. Unless otherwise stated, a reference to *the Group Services subsystem* is a reference to the Group Services subsystem in a single system partition.

A GS client communicates with a Group Services daemon that is running on the same node as the GS client. A GS client communicates with the Group Services daemon, through the GSAPI software, using a Unix domain socket. Before a GS client registers with Group Services, it must set the **HA_DOMAIN_NAME** environment variable to the name of the system partition in which it is executing. Note that for previous releases of Group Services, the GS clients needed to set the variable **HA_SYSPAR_NAME**. For PSSP 3.1, you can set either variable but to support compatibility of older clients, all new GS clients should use the variable **HA_DOMAIN_NAME** because the variable **HA_SYSPAR_NAME** might eventually be unsupported.

The Group Services API (GSAPI)

The Group Services Application Programming Interface (GSAPI) is a shared library that a GS client uses to obtain the services of the Group Services subsystem. This shared library is supplied in two versions: one for non-thread-safe programs and one for thread-safe programs. These libraries are referenced by the following path names:

- **/usr/lib/libha_gs.a** (non-thread-safe version)
- **/usr/lib/libha_gs_r.a** (thread-safe version)

These path names are actually symbolic links to the files **/usr/sbin/rsct/lib/libha_gs.a** and **/usr/sbin/rsct/lib/libha_gs_r.a**, respectively. The symbolic links are placed in **/usr/lib** for ease of use. For serviceability, the actual libraries are placed in the **/usr/sbin/rsct/lib** directory. These libraries are supplied as shared libraries, also for serviceability.

For details on the GSAPI software, see *Group Services Programming Guide and Reference*.

To allow non-root users to use Group Services:

1. Create a group named **hagsuser**.
2. Add the desired user IDs to the **hagsuser** group.
3. Stop and restart **hags** (if it was running before you created the **hagsuser** group).

Users in the created **hagsuser** group can use Group Services (**libha_gs**).

Port Numbers and Sockets

The Group Services subsystem uses several types of communications:

- UDP port numbers for intra-domain communications, that is, communications between Group Services daemons within an operational domain.
- Unix domain sockets for communication between GS clients and the local Group Services daemon (via the GSAPI).

Intra-Domain Port Numbers

For communication between Group Services daemons within an operational domain, the Group Services subsystem uses a single UDP port number. This port number is recorded in the **Syspar_ports** SDR class. The class contains two attributes: *subsystem* and *port*. The *Syspar_ports* class object with a subsystem attribute value of **hags** contains the port number for the Group Services subsystem. Note that the **Syspar_ports** class is a partitioned SDR class, that is, there is a set of objects for each system partition. When you configure the Group Services subsystem for a particular system partition, use the SDR data that corresponds to that system partition.

The Group Services port number is stored in the SDR so that, when the Group Services subsystem is configured on each node, the port number is fetched from the SDR. This ensures that the same port number is used by all Group Services daemons in the same operational domain. Note that because a Group Services daemon from each operational domain runs on the control workstation, the Group Services subsystem in each system partition must have a unique port number.

This intra-domain port number is also set in the **/etc/services** file, using the service name **hags.syspar_name**, where *syspar_name* is the name of the system partition. The **/etc/services** file is updated on all nodes in the system partition and on the control workstation. The Group Services daemon obtains the port number from the **/etc/services** file during initialization.

Unix Domain Sockets

Unix domain sockets are used for communication between GS clients and the local Group Services daemon (via the GSAPI). These are connection-oriented sockets. The socket name used by the GSAPI to connect to the Group Services daemon is **/var/ha/soc/hagsdsocket.syspar_name**, where *syspar_name* is the name of the system partition.

The Control Script (hagsctrl)

The Group Services control script is contained in the executable file **/usr/sbin/rsct/bin/hagsctrl**. This script is normally invoked by the **syspar_ctrl** command, which provides an interface to all of the system-partition-sensitive subsystems.

If necessary, you can invoke the **hagsctrl** command directly from the command line. Note that before you invoke the **hagsctrl** command, you must ensure that the **SP_NAME** environment variable is set to the appropriate system partition name.

For more information about the **hagsctrl** and **syspar_ctrl** commands, see the book *PSSP: Command and Technical Reference*.

The purpose of the **hagsctrl** command is to add (configure) the Group Services subsystem to a system partition. It can also be used to remove the subsystem from a system partition, start the subsystem, stop it, and clean the subsystem from all system partitions.

For more information, see “Configuring Group Services” on page 330.

Files and Directories

The Group Services subsystem uses the following directories:

- **/var/ha/lck**, for lock files
- **/var/ha/log**, for log files
- **/var/ha/run**, for Group Services daemon current working directories
- **/var/ha/soc**, for socket files.

The /var/ha/lck Directory (Lock Files)

In the **/var/ha/lck** directory, the **hags.tid.syspar_name** directory is used to ensure a single running instance of the Group Services daemon, and to establish an instance number for each invocation of the daemon. In the directory name, *syspar_name* is the name of the system partition. On the control workstation, there may be several instances of the Group Services daemon running, but they are in different operational domains.

The /var/ha/log Directory (Log Files)

The **/var/ha/log** directory contains trace output from the Group Services daemon.

On the nodes, the file is called **hags_nodenum_instnum.syspar_name**, where:

- *nodenum* is the node number on which the daemon is running
- *instnum* is the instance number of the daemon.
- *syspar_name* is the name of the system partition to which the node belongs.

On the control workstation and the nodes, the file

hags.default.syspar_name.nodenum_instnum contains trace output from the initial startup of the daemon.

The Group Services daemon limits the log size to a pre-established number of lines (by default, 5,000 lines). When the limit is reached, the daemon appends the string **.bak** to the name of the current log file and begins a new log. If a **.bak** version already exists, it is removed before the current log is renamed.

The /var/ha/run Directory (Daemon Working Files)

In the **/var/ha/run** directory, a directory called **hags.syspar_name** is created, where *syspar_name* is the system partition name. This directory is the current working directory for the Group Services daemon. If the Group Services daemon abnormally terminates, the core dump file is placed in this directory. Whenever the Group Services daemon starts, it renames any core file to **core_nodenum_instnum**, where *nodenum* is the node number on which the daemon is running and *instnum* is the instance number of the previous instance of the daemon.

Components on Which Group Services Depends

The Group Services subsystem depends on the following components:

System Resource Controller (SRC)

An AIX feature that can be used to define and control subsystems. The Group Services subsystem is called **hags** on SP nodes and is called **hags.syspar_name** on the control workstation, where *syspar_name* is the name of a system partition. The subsystem name is used with the SRC commands (for example, **startsrc** and **lssrc**).

System Data Repository (SDR)

A repository of SP system configuration information.

Topology Services

A PSSP subsystem that is used to determine which nodes in a system can be reached (that is, are running) at any given time. It is often referred to as **heartbeat**. The Topology Services subsystem is SRC-controlled. It is called **hats** on SP nodes and is called **hats.syspar_name** on the control workstation, where *syspar_name* is the name of a system partition.

Configuring and Operating Group Services

The following sections describe how the components of the Group Services subsystem work together to provide group services. Included are discussions of Group Services:

- Configuration
- Daemon initialization and errors
- Operation

Configuring Group Services

The RSCT software is installed as part of the installation of the PSSP product. The Group Services subsystem is contained in **rsct.basic.rte** and **rsct.basic.sp** filesets.

After the components are installed, the subsystem must be configured for operation. Group Services configuration is performed by the **hagsctrl** command, which is invoked by the **syspar_ctrl** command.

The **syspar_ctrl** command configures all of the system-partition-sensitive subsystems. The person who installs PSSP issues the **syspar_ctrl** command during installation of the control workstation. The **syspar_ctrl** command is executed automatically on the nodes when the nodes are installed. The **syspar_ctrl** command is also executed automatically when system partitions are created or destroyed. For more information on using the **syspar_ctrl** command, see the books *RS/6000 SP: Installation Guide* and *PSSP: Command and Technical Reference*.

The **hagsctrl** command provides a number of functions for controlling the operation of the Group Services system. You can use it to:

- Add (configure) the Group Services subsystem

- Start the subsystem
- Stop the subsystem
- Delete (unconfigure) the subsystem
- Clean all Group Services subsystems
- Turn tracing of the Group Services daemon on or off

Except for the clean function, **hagsctrl** affects the Group Services subsystem in the current system partition, that is, the system partition that is specified by the **SP_NAME** environment variable.

Adding the Subsystem

If the **hagsctrl** command is running on the control workstation, the first step in the add function is to select a Group Services daemon communications port number and save it in the **Syspar_ports** SDR class. This port number is then placed in the **/etc/services** file. If the **hagsctrl** command is running on a node, the port number is fetched from the SDR and placed in the **/etc/services** file. Port numbers are selected from the range 10000 through 10100.

The second step is to add the Group Services daemon to the System Resource Controller (SRC) using the **mkssys** command. The system partition name is an argument to the **hagsd** program in the SRC subsystem specification.

The third step is to add an entry to the **/etc/inittab** file so that the Group Services daemon will be started during boot. However, if **hagsctrl** is running on a workstation with the High Availability Control Workstation (HACWS) component of PSSP, no entry is made in the **/etc/inittab** file. Instead, the HACWS component manages the starting and stopping of the Group Services daemon.

Note that if the **hagsctrl** add function terminates with an error, the command can be rerun after the problem is fixed. The command takes into account any steps that already completed successfully.

Starting and Stopping the Subsystem

The start and stop functions of the **hagsctrl** command simply run the **startsrc** and **stopsrc** commands, respectively. However, **hagsctrl** automatically specifies the subsystem argument to these SRC commands.

Deleting the Subsystem

The delete function of the **hagsctrl** command removes the subsystem from the SRC, removes the entry from **/etc/inittab**, and removes the Group Services daemon communications port number from **/etc/services**. It does **not** remove anything from the SDR, because the Group Services subsystem may still be configured on other nodes in the operational domain.

Cleaning the Subsystem

The clean function of the **hagsctrl** command performs the same function as the delete function, except in all system partitions. In addition, it removes the Group Services daemon remote client communications port number from the **/etc/services** file.

The clean function does **not** remove anything from the SDR. This function is provided to support restoring the system to a known state, where the known state is the (possibly restored copy of the) SDR database.

Tracing the Subsystem

The tracing function of the **hagsctrl** command is provided to supply additional problem determination information when it is requested by the IBM Support Center. Normally, tracing should **not** be turned on, because it might slightly degrade Group Services subsystem performance and can consume large amounts of disk space in the **/var** file system.

Initializing Group Services Daemon

Normally, the Group Services daemon is started by an entry in the **/etc/inittab** file, via the **startsrc** command. If necessary, the Group Services daemon can be started using the **hagsctrl** command or the **startsrc** command directly.

During initialization, the Group Services daemon performs the following steps:

1. It gets the number of the node on which it is running using the **/usr/lpp/ssp/install/bin/node_number** command. Node 0 is the control workstation.
2. It tries to connect to the Topology Services subsystem. If the connection cannot be established because the Topology Services subsystem is not running, it is scheduled to be retried every 20 seconds. This continues until the connection to Topology Services is established. Until the connection is established, the Group Services daemon writes an AIX error log entry periodically and no clients may connect to the Group Services subsystem.
3. It performs actions that are necessary to become a daemon. This includes establishing communications with the SRC subsystem so that it can return status in response to SRC commands.
4. It establishes the Group Services domain, which is the set of nodes within the SP system partition in which a Group Services daemon is executing.

At this point, one of the GS daemons establishes itself as the GS nameserver. For details, see “Establishing the GS Nameserver.”

Until the domain is established, no GS client requests to join or subscribe to groups are processed.

5. It enters the main control loop.

In this loop, the Group Services daemon waits for requests from GS clients, messages from other Group Services daemons, messages from the Topology Services subsystem, and requests from the SRC for status.

Establishing the GS Nameserver

The Group Services subsystem must be able to keep track of the groups that its clients want to form. To do this, it establishes a GS nameserver within each domain (a domain is the set of running nodes within each SP system partition). The GS nameserver is responsible for keeping track of all client groups that are created in that domain.

To ensure that only one node becomes a GS nameserver, Group Services uses the following protocol:

1. When each daemon is connected to the Topology Services subsystem, it waits for Topology Services to tell it which nodes are currently running in this system partition.
2. Based on the input from Topology Services, each daemon finds the lowest-numbered running node in the domain. The daemon compares its own node number to the lowest-numbered node and performs one of the following:
 - If the node the daemon is on is the lowest-numbered node, the daemon waits for all other running nodes to nominate it as the GS nameserver.
 - If the node the daemon is on is not the lowest-numbered node, it sends nomination messages to the lowest-numbered node periodically, initially every 5 seconds.
3. Once all running nodes have nominated the GS nameserver-to-be and a coronation timer (about 20 seconds) has expired, the nominee sends an insert message to the nodes. All nodes must acknowledge this message. When they do, the nominee becomes the established GS nameserver, and it sends a commit message to all of the nodes.
4. At this point, the Group Services domain is established, and requests by clients to join or subscribe to groups are processed.

Note that this description is in effect when all nodes are being booted simultaneously, such as at initial system power-on. It is often the case, however, that a Group Services daemon is already running on at least one node (for example, the control workstation) and is already established as the domain's GS nameserver. In that case, the GS nameserver waits only for Topology Services to identify the newly running nodes. The GS nameserver will then send the newly running nodes proclaim messages that direct the nodes to nominate it as nameserver. Once those nodes then nominate the GS nameserver, the GS nameserver simply executes one or more insert protocols to insert the newly-running nodes into the domain.

Group Services Initialization Errors

The Group Services subsystem creates AIX error log entries to indicate severe internal problems. For most of these, the best response is to contact the IBM Support Center.

However, if you get a message that there has been no heartbeat connection for some time, it could mean that the Topology Services subsystem is not running.

To check the status of the Topology Services subsystem, issue the **lssrc -g hats** command. If the response indicates that the Topology Services subsystem is inoperative, try to restart it using the **hatsctrl -s** command. If you are unable to restart it, call the IBM Support Center.

Group Services Daemon Operation

Normal operation of the Group Services subsystem requires no administrative intervention. The subsystem normally recovers from temporary failures, such as node failures or failures of Group Services daemons, automatically. However, there are some operational characteristics that might be of interest to administrators.

Due to AIX per-process file descriptor limits, the Group Services subsystem can support a maximum of approximately 2000 GS clients on each node.

The maximum number of nodes that can be contained within a domain is 2048, although Group Services can support node numbers in the range 0 to 65535.

The maximum number of groups to which a GS client can subscribe or that a GS client can join is equivalent to the largest value containable in a signed integer variable.

The maximum number of groups allowed within a domain is 65,535.

These limits are the theoretical maximum limits. In practice, the amount of memory available to the Group Services daemon and its clients will reduce the limits to smaller values.

Group Services Procedures

For the most part the Group Services subsystem runs itself without requiring administrator intervention. However, on occasion, you may need to check the status of the subsystem.

Displaying the Status of the Group Services Daemon

You can display the operational status of the Group Services daemon by issuing the **Issrc** command.

On the control workstation, enter:

```
Issrc -l -s hags.syspar_name
```

where *syspar_name* is the name of the system partition of interest.

On a node, enter:

```
Issrc -l -s hags
```

In response, the **Issrc** command writes the status information to standard output. The information includes:

- The information provided by the **Issrc -s hags** command (short form)
- The number of currently connected clients and their process IDs
- The status of the Group Services domain
- The node number on which the GS nameserver is running
- Statistics for client groups with providers or subscribers on this node.

Note that if the **Issrc** command times out, the Group Services daemon is probably unable to connect to the Topology Services subsystem. For more information, see “Group Services Initialization Errors” on page 333.

The following is sample output from the **Issrc -l -s hags** command on a node for the **k21sp2** system partition:

```

Subsystem      Group          PID      Status
hags           hags          11938   active
4 locally-connected clients. Their PIDs:
21344 17000 18852 23486
HA Group Services domain information:
Domain established by node 9.
Number of groups known locally: 3

```

Group name	Number of providers	Number of local providers/subscribers
cssMembership	5	1
WomSchg_1	5	1
ha_em_peers	7	1

Here is a sample of the output of the **lssrc -l -s hags.k21sp2** command on the control workstation for the same domain:

```

Subsystem      Group          PID      Status
hags.k21sp2    hags          24804   active
1 locally-connected clients. Their PIDs:
44146
HA Group Services domain information:
Domain established by node 9.
Number of groups known locally: 1

```

Group name	Number of providers	Number of local providers/subscribers
ha_em_peers	7	1

In this domain, the GS nameserver is on node 9 of the system partition.

If a GS nameserver has not yet been established, the status indicates that the domain is not established. Similarly, if the GS nameserver fails, the status shows that the domain is recovering. Both of these conditions should clear in a short time. If they do not and the Topology Services subsystem is active, call the IBM Support Center.

Chapter 25. The Event Management Subsystem

This chapter introduces you to the Event Management subsystem. It includes information about the subsystem's components, its configuration, other components on which it depends, and its operation. It also discusses the relationship of the Event Management subsystem to the other high availability subsystems. Finally, it contains procedures you can use to check the status of the subsystem, or to add or change some of the Event Management configuration data.

For more information on Event Management, refer to the IBM Redbook titled *SP Monitoring: Keeping it Alive*, SG24-4873 and the *RS/6000 SP High Availability Infrastructure*, SG24-4838.

After reading this chapter, you will be able to manage the Event Management subsystem and, if necessary, perform problem determination.

Introducing Event Management

Event Management is a distributed subsystem of the RS/6000 Cluster Technology (RSCT) on the RS/6000 system. The RSCT provides a set of high availability services to the IBM Parallel System Support Programs (PSSP). The other services in RSCT are the Group Services and Topology Services (heartbeat) distributed subsystems. These three distributed subsystems operate within a **domain**. A domain is a set of RS/6000 machines upon which the RSCT components execute and, exclusively of other machines, provide their services. On the RS/6000 SP, a domain is a system partition. Note that a machine may be in more than one RSCT domain; the control workstation is a member of each system partition and, therefore, a member of each RSCT domain. When a machine is a member of more than one domain, there is an executing copy of each RSCT component per domain.

The function of the Event Management subsystem is to match information about the state of system resources with information about resource conditions that are of interest to client programs, which may include applications, subsystems, and other programs. In this chapter, these client programs are referred to as **EM clients**.

Resource states are represented by resource variables. Resource conditions are represented as expressions that have a syntax that is a subset of the expression syntax of the C programming language.

Resource monitors are programs that observe the state of specific system resources and transform this state into several resource variables. The resource monitors periodically pass these variables to the Event Manager daemon. The Event Manager daemon applies expressions, which have been specified by EM clients, to each resource variable. If the expression is true, an event is generated and sent to the appropriate EM client. EM clients may also query the Event Manager daemon for the current values of resource variables.

Resource variables, resource monitors, and other related information are specified in several System Data Repository (SDR) object classes. Information stored in these SDR classes is then translated into a form that can be easily used by the Event Management subsystem.

For more conceptual information about the Event Management subsystem, or to learn how to use the Event Management Perspective to view descriptions of resource variables, see the RS/6000 Cluster Technology *Event Management Programming Guide and Reference*.

Event Management Components

The Event Management subsystem consists of the following components:

Event Manager daemon

The central component of the Event Management subsystem.

Event Management API (EMAPI)

The application programming interface that EM clients use to obtain the services of the Event Management subsystem.

Resource Monitor API (RMAPI)

The application programming interface that resource monitors use to supply resource variables to the Event Manager daemon.

Resource monitors Programs that monitor the state of system resources. Several resource monitors are shipped with RSCT.

Port numbers TCP/IP port numbers that the Event Management subsystem uses for communications. The Event Management subsystem also uses Unix domain sockets.

Configuration data Information that defines resource variables, resource monitors, and related data to the Event Management subsystem. This information is stored in the SDR.

Configuration utility

A program that translates the Event Management configuration data that is stored in the SDR to a binary format that is used by the Event Manager daemon and the RMAPI.

Configuration database

The Event Management Configuration Database (EMCDB) is the output of the configuration utility.

Control script A shell script that is used to add, start, stop, and delete the Event Management subsystem, which operates under System Resource Controller (SRC) control.

Configuration load utility

A program that loads the default configuration data that is shipped with RSCT into the SDR.

Default configuration data

A file that is shipped with RSCT that contains the default configuration data for the Event Management subsystem.

Files and directories

Various files and directories that are used by the Event Management subsystem to maintain run-time data.

The sections that follow contain more details about each of these components.

The Event Manager Daemon (haemd)

The Event Manager daemon is contained in the executable file **/usr/sbin/rsct/bin/haemd**. This daemon runs on each node (machine) of a domain. If a node is a member of more than one domain, there is one executing copy of the daemon per domain.

Note that on the RS/6000 SP, the domain of the Event Management subsystem is a system partition. The domain name is the same as the system partition name. The control workstation is regarded as node 0 in each system partition and, therefore, in each domain. Unless otherwise stated, a reference to the "Event Management subsystem" is a reference to the Event Management subsystem in a single domain.

EM clients communicate with a single Event Manager daemon in the domain that contains the resources in which they are interested, as follows:

- If the domain contains the SP node on which the EM client is running, the client communicates with the local Event Manager daemon, through the EMAPI, using a Unix domain socket.
- If an EM client is running on the control workstation, it must specify, explicitly or implicitly, a domain. The EM client then communicates with the Event Manager daemon on the control workstation, in that domain, using a Unix domain socket.

An EM client that is running on a node that is outside of the domain that contains the resources in which it is interested uses a TCP socket to communicate with the Event Manager daemon on the control workstation that is in that domain. An EM client that is running on any RS/6000 is considered to be on a node outside of any domain. In both of these cases, the Event Management client must specify, explicitly or implicitly, the name of a domain that contains the resources of interest. For an Event Management client to run on a RS/6000, the RSCT filesets **rsct.clients.rte** and **rsct.clients.sp** and the PSSP fileset **ssp.clients** must be installed on the RS/6000.

Once an EM client has established communications with an Event Manager daemon, the EM client has access to all of the resources in that domain.

Resource monitors communicate with the Event Manager daemon, through the RMAPI, using Unix domain sockets. Resource monitors are always on the same node as the Event Manager daemon. Note that if a node is a member of multiple domains, just as there is one executing copy of the daemon per domain, there is also one executing copy of each resource monitor per domain.

The Event Management API (EMAPI)

The Event Management Application Programming Interface (EMAPI) is a shared library that an EM client uses to obtain the services of the Event Management subsystem. This shared library is supplied in two versions: one for non-thread safe programs and one for thread-safe programs. These libraries are referenced by the following path names:

- **/usr/lib/libha_em.a** (non-thread safe version)
- **/usr/lib/libha_em_r.a** (thread-safe version)

These path names are actually symbolic links to `/usr/sbin/rsct/lib/libha_em.a` and `/usr/sbin/rsct/lib/libha_em_r.a`, respectively. The symbolic links are placed in `/usr/lib` for ease of use. For serviceability, the actual libraries are placed in the `/usr/sbin/rsct/lib` directory. These libraries are supplied as shared libraries, also for serviceability.

For details on the EMAPI, see *Event Management Programming Guide and Reference*.

The Resource Monitor API (RMAPI)

The Resource Monitor Application Programming Interface (RMAPI) is a shared library that a resource monitor uses to supply resource variables to the Event Manager daemon. This shared library is supplied only in a non-thread safe version. This library is referenced through the path name

- `/usr/lib/libha_rr.a`

which is a symbolic link to `/usr/sbin/rsct/lib/libha_rr.a`. The symbolic link is placed in `/usr/lib` for ease of use. For serviceability, the actual library is placed in the `/usr/sbin/rsct/lib` directory. This library is supplied as a shared library, also for serviceability.

For details on the RMAPI, see *Event Management Programming Guide and Reference*.

Resource Monitors

A resource monitor conforms to one of the following programming models:

- The resource monitor is a daemon. When necessary, it is started by the Event Manager daemon. The Event Manager daemon connects to the resource monitor to establish communications. The resource monitor has a connection type of **server**.
- The resource monitor logic is incorporated in a subsystem that manages the resources. The Event Manager daemon connects to the resource monitor to establish communications, but the Event Manager daemon does not start the resource monitor (because it is actually another subsystem). The resource monitor has a connection type of **server**.
- The resource monitor logic is implemented in a command. The command can be used by scripts to supply resource variables to the Event Manager daemon. A command-based resource monitor connects to the Event Manager daemon to establish communications. The resource monitor has a connection type of **client**.

Note that a **server** type resource monitor may have multiple executing copies or *instances*. Each resource monitor instance may supply different instances of the same named resource variable or may supply different resource variables.

In addition, the Event Manager daemon itself performs some resource monitoring function. This function is considered to be a resource monitor with a connection type of **internal**.

RSCT supplies the following resource monitors:

IBM.PSSP.harmlid	Supplies resource variables for the CSS, VSD, and LoadLeveler subsystems. This data is also furnished to the Performance Monitor subsystem. This is a daemon with a connection type of server .
IBM.PSSP.harmpd	Supplies resource variables that represent the number of processes executing a particular program. These variables can be used to determine whether a particular system daemon is running. This is a daemon with a connection type of server .
IBM.PSSP.hmrmd	Supplies resource variables that represent the hardware state of the SP system. The resource information is obtained from the PSSP hardware monitoring subsystem (hardmon). This is a daemon with a connection type of server .
IBM.PSSP.pmanrmd	Supplies resource variables provided by the PSSP Problem Management subsystem. This is a command-based resource monitor with a connection type of client .
aixos	Supplies resource variables that represent AIX operating system resources. This is a daemon with a connection type of server .
IBM.PSSP.CSSLogMon	Supplies a resource variable that represents the state of CSS error log entries. This is a command-based resource monitor with a connection type of client .
IBM.PSSP.SDR	Supplies a resource variable that represents the modification state of SDR classes. This is a command-based resource monitor with a connection type of client .

There are also several internal resource monitors incorporated in the Event Manager daemon itself:

Membership	Supplies resource variables that represent the Host Membership and Adapter Membership states. The Event Manager daemon obtains this information directly from the Group Services subsystem by subscribing to the HostMembership , enMembership , and cssMembership system groups.
Response	Supplies resource variables that represent the information in the host_responds and switch_responds SDR classes. These resource variables are provided for compatibility with earlier releases of PSSP.

Port Numbers and Sockets

The Event Management subsystem uses several types of communications:

- UDP port numbers for intra-domain communications, that is, communications between Event Manager daemons within a domain.
- TCP port numbers for remote communications, that is, communications between Event Manager daemons and EM clients that are running outside their domains.

- Unix domain sockets for communication between EM clients and the local Event Manager daemon (via the EMAPI) and between resource monitors and the Event Manager daemon (via the RMAPI).

Intra-Domain Port Numbers

For communication between Event Manager daemons within a domain, the Event Management subsystem uses a single UDP port number. This port number is recorded in the **Syspar_ports** SDR class. The class contains two attributes: **subsystem** and **port**. The class object whose **subsystem** attribute has a value of **haem** contains the port number for the Event Management subsystem. Note that the **Syspar_ports** class is a partitioned SDR class, that is, there is a set of objects for each system partition. When you configure the Event Management subsystem for a particular domain, you use the SDR data that corresponds to that domain, that is, system partition.

The Event Management port number is stored in the SDR so that, when the Event Management subsystem is configured on each node, the port number is fetched from the SDR. This ensures that the same port number is used by all Event Manager daemons in the same domain. Note that because an Event Manager daemon from each domain runs on the control workstation, the Event Management subsystem in each domain must have a unique port number.

This intra-domain port number is also set in the **/etc/services** file, using the service name **haem.domain_name**, where *domain_name* is the name of the domain. The **/etc/services** file is updated on all nodes in the domain and on the control workstation. The Event Manager daemon obtains the port number from the **/etc/services** file during initialization.

Remote Port Numbers

The Event Management subsystem also uses a single TCP port number to accept communications from clients outside its domain. All Event Manager daemons that run on the control workstation use this port number to accept connections from EM clients outside their respective domains. Each daemon binds to a distinct IP address that is the IP address of its system partition.

The TCP port number is stored in the **SP_ports** SDR system class. The class contains the **daemon**, **hostname**, and **port** attributes. The class object whose **daemon** attribute has a value of **haemd** contains the port number for the Event Management subsystem(s). The **hostname** attribute is not used in this object.

This remote port number is also set in the **/etc/services** file on the control workstation, using the service name **haemd**.

Unix Domain Sockets

Unix domain sockets are used for communication:

- Between EM clients and the local Event Manager daemon (via the EMAPI)
- Between resource monitors and the Event Manager daemon (via the RMAPI).

These are connection-oriented sockets. The following socket names are used (*domain_name* is the name of the domain):

/var/ha/soc/em.clsrv.domain_name

Used by the EMAPI to connect to the Event Manager daemon.

/var/ha/soc/em.rmsrv.domain_name

Used by resource monitors to connect to the Event Manager daemon.

/var/ha/soc/haem/em.RMrmname.rminst.domain_name

Used by the Event Management daemon to connect to the resource monitor that is specified by *rmname* and *rminst*, where *rmname* is the resource monitor name and *.rminst* is the resource monitor instance number. This resource monitor has a connection type of **server**.

Configuration Data

Configuration information for the Event Management subsystem is stored in the SDR as objects in the following classes:

EM_Resource_Variable

Contains the definitions of all resource variables. The resources with which these variables are associated are implied in the definitions.

EM_Resource_ID

Contains the definitions of all of the resource IDs that are associated with the resources that are specified in the **EM_Resource_Variable** class.

EM_Structured_Byte_String

Contains the definitions of the structured fields for all resource variables that have a data type of structured byte string (SBS).

EM_Resource_Class

Contains the definitions for the resource classes with which each resource variable is associated.

EM_Resource_Monitor

Contains the definitions for all resource monitors.

For details about the Event Management SDR classes and attributes, see *Event Management Programming Guide and Reference*.

These SDR classes are partitioned classes. Therefore, each Event Management subsystem domain has its own configuration data. This permits a different set of resources to be monitored in each domain, as appropriate for the workload within the domain.

The Event Management SDR objects are loaded from the default configuration data when the Event Management subsystem is added to a domain using the control script.

There are several attributes that you, as the system administrator, may wish to change. They are:

- The **rvExpression** attribute of the **EM_Resource_Variable** class

Change this attribute to add or replace a default expression for a resource variable.

- The **rcObservation_interval** attribute of the **EM_Resource_Class**.

Change this attribute to increase or decrease the time between observations of a resource variable value by the Event Manager daemon.

For more information, see “Changing Configuration Data in the SDR” on page 358.

If resource monitors supplied by other IBM products or by third parties are installed on the SP system, you must load the configuration data supplied with the resource monitors into the SDR. For more information, see “Loading Non-PSSP Configuration Data into the SDR” on page 357.

The Configuration Utility (haemcfg)

The Event Management configuration utility is contained in the executable file **/usr/sbin/rsct/bin/haemcfg**. This utility is normally run on the control workstation in the environment of a system partition, that is, the value of the **SP_NAME** environment variable determines the partitioned SDR classes that are referenced. If **SP_NAME** is not set, the default system partition is assumed. Recall that a domain is equivalent to a system partition.

The **haemcfg** command converts the data stored in the Event Management SDR classes to a binary format called the Event Management Configuration Database (EMCDB). The EMCDB is placed in the staging file called **/spdata/sys1/ha/cfg/em.domain_name.cdb**, where *domain_name* is the name of the domain in which the utility is run.

A version string that uniquely identifies the EMCDB is placed both in the EMCDB itself and in the **Syspar** partitioned SDR class. The version string is set as the value of the **haem_cdb_version** attribute in the object whose **syspar_name** attribute value matches the current system partition. The version string is used to ensure that all of the Event Manager daemons and resource monitors that are running within the domain are using the same copy of the EMCDB.

If a previous copy of the **em.domain_name.cdb** file exists, it is renamed to **/spdata/sys1/ha/cfg/em.domain_name.cdb.ssss,nnnn,v**, where *ssss,nnnn,v* is the version string of the previous copy.

The **haemcfg** command is automatically invoked by the control script when the Event Management subsystem is added to a domain.

For details on the **haemcfg** command, see *PSSP Command and Technical Reference*.

The Configuration Database (EMCDB)

The Event Management Configuration Database (EMCDB) is produced by the **haemcfg** command from the information in the SDR. The format of the EMCDB is designed to permit quick loading of the database by the Event Manager daemon and the RMAPI. It also contains configuration data in an optimized format to minimize the amount of data that must be sent between Event Manager daemons and between an Event Manager daemon and its resource monitors.

When the SDR data is compiled, the EMCDB is placed in a staging file. When the Event Manager daemon on a node or the control workstation initializes, it automatically copies the EMCDB from the staging file to a run-time file on the node or the control workstation. The run-time file is called `/etc/ha/cfg/em.domain_name.cdb`, where `domain_name` is the name of the domain.

For more information, see “Reading the EMCDB” on page 353.

The Control Script (`haemctrl`)

The Event Management control script is contained in the executable file `/usr/sbin/rsct/haemctrl`. This script is normally invoked by the `syspar_ctrl` script, which provides an interface to all of the system partition-sensitive subsystems.

If necessary, you can invoke `haemctrl` directly from the command line. Note that before you invoke `haemctrl`, you must ensure that the `SP_NAME` environment variable is set to the appropriate system partition name.

For more information about `haemctrl` and `syspar_ctrl`, see *PSSP Command and Technical Reference*.

The purpose of the `haemctrl` command is to add (configure) the Event Management subsystem to a domain. It can also be used to remove the subsystem from a domain, start the subsystem, stop it, and clean the subsystem from all domains.

For more information, see “Configuring Event Management” on page 349.

The Configuration Load Utility (`haemloadcfg`)

The configuration load utility is contained in the executable file `/usr/sbin/rsct/install/bin/haemloadcfg`. The purpose of this program is to load the default configuration data into the SDR. Normally, it is invoked by the `haemctrl` script, but you can also use it to load additional configuration data into the SDR. Note that before you invoke `haemloadcfg`, you must ensure that the `SP_NAME` environment variable is set to the appropriate system partition name.

By default, this utility does not replace existing objects in the SDR. Input data is matched with existing objects based on key attributes. The key attributes for each of the Event Management SDR classes are:

SDR Class	Key Attributes
<code>EM_Resource_Variable</code>	<code>rvName</code>
<code>EM_Resource_ID</code>	<code>riResource_name</code> , <code>riElement_name</code>
<code>EM_Structured_Byte_String</code>	<code>sbsVariable_name</code> , <code>sbsField_name</code>
<code>EM_Resource_Class</code>	<code>rcClass</code>
<code>EM_Resource_Monitor</code>	<code>rmName</code>

Note that the way in which the `haemloadcfg` command handles existing SDR objects is different from the way in which the `SDRCreatObjects` command handles them. The `SDRCreatObjects` command creates a new object as long as the attributes, taken as a group, are unique.

The **haemloadcfg** command is actually a shell script that invokes the **/usr/sbin/rsct/bin/loadsd** program. The purpose of this program is to batch load any number of objects into SDR classes. However, the **loadsd** command cannot load arbitrary SDR classes. It can load only the Event Management SDR classes.

In this release of Event Management subsystem, the **EM_Resource_ID** SDR class replaces the **EM_Instance_Vector** class from prior releases. In the **EM_Resource_Variable** SDR class, the **rvExpression** and **rvIndex_element** attributes replace the **rvPredicate** and **rvIndex_vector** attributes, respectively. The **loadsd** program automatically migrates SDR data that may be present from prior releases to the new forms.

The Default Configuration Data (haemloadlist)

The default Event Management configuration data for PSSP is supplied in the **/usr/sbin/rsct/install/config/haemloadlist** file. The data in this file is loaded into each system partition to which the Event Management subsystem is added by the **haemctrl** script, using the **haemloadcfg** command.

Files and Directories

The Event Management subsystem uses the following directories:

- **/var/ha/lck**, for lock files
- **/var/ha/log**, for log files
- **/var/ha/run**, for Event Manager daemon current working directories
- **/var/ha/soc**, for socket files
- **/etc/ha/cfg**, for run-time EMCDB and related files
- **/spdata/sys1/ha/cfg**, as the staging area for EMCDB files.

The /var/ha/lck Directory (Lock Files)

In the **/var/ha/lck/haem** directory, the **em.RMrmname.domain_name** file is used to manage one or more running instances of a resource monitor. In this file name, *rmname* is the name of the resource monitor and *domain_name* is the name of the domain.

The **em.RMrmname.rminstSHM.domain_name** file in the **/var/ha/lck** directory is used by the Event Management daemon to manage a shared memory segment used by the daemon and resource monitor instance specified by *rmname* and *rminst* where *rmname* is the resource monitor name and *rminst* is the resource monitor instance number. The file contains the shared memory segment ID.

The **em.haemd.domain_name** file in the **/var/ha/lck** directory is used to ensure a single running instance of an Event Manager daemon in a domain.

The **em.RMrmnameSPMI** file in the **/var/ha/lck** directory is used by the RMAPI to create a shared memory key when the resource monitor attaches to the shared memory segment managed by the SPMI library.

The /var/ha/log Directory (Log Files)

In the **/var/ha/log** directory, the **em.trace.domain_name** file contains trace output from the Event Manager daemon. In the file name, *domain_name* is the name of the domain.

The **em.msgtrace.domain_name** file contains message trace output from the Event Manager daemon.

The **em.default.domain_name** file contains any error messages from the Event Manager daemon that cannot be written to the AIX Error Log. Normally, all daemon error messages are written to the AIX Error Log.

In addition to error messages that cannot be written to the AIX Error Log, the **/var/ha/log/em.default.domain_name** file also contains error messages that result from repetitive operational errors. Therefore, both the AIX Error Log and the **/var/ha/log/em.default.domain_name** file must be examined when performing problem determination on the Event Management subsystem.

The size of the **/var/ha/log/em.default.domain_name** file is examined every two minutes. If the size of the file exceeds 256K, the file is renamed to **/var/ha/log/em.default.domain_name.last** and a new default file is created. No more than two copies of this file are kept: the “current” **em.default.domain_name** file and the “last” file.

The **/var/ha/log/em.default.domain_name.n** file is used to record additional error information if the Event Manager daemon cannot start a resource monitor. The error information includes the name of the resource monitor that could not be started.

The /var/ha/run Directory (Daemon Working Files)

In the **/var/ha/run** directory, a directory called **haem.domain_name** is created, where *domain_name* is the domain name. This directory is the current working directory for the Event Manager daemon. If the Event Manager daemon abnormally terminates, the core dump file is placed in this directory. Whenever the Event Manager daemon starts, it renames any core file to **core.last**.

If the Event Manager daemon detects an error in the shared memory segment used by the daemon and a resource monitor instance, it creates a dump file containing the first 4096 bytes of the shared memory segment in this directory. The dump file is named **rzdump.RMrmname.rminst.time** where *rmname* is the resource monitor name, *rminst* is the resource monitor instance, and *time* is the time stamp.

This directory also contains the working directories of any resource monitors that are started by the Event Manager daemon. Each directory has the name of its resource monitor.

Finally, this directory contains the **Rcache_local** and **Rcache_remote** directories. These directories contain the registration cache for local and remote client registration requests, respectively.

For each EM client that establishes communications with the Event Manager daemon, a cache subdirectory is created in the appropriate registration cache directory. This subdirectory has a name of the form *p,s,n,q,i*, where:

- *p* is the process ID of the EM client

- *s* is the seconds portion of the timestamp recorded when the EM client established its session with the Event Manager daemon
- *n* is the nanoseconds portion of the timestamp
- *q* is a sequence number for the session
- *i* is the IP address of the host where the EM client is running. For local clients, the IP address is 0.0.0.0.

Within each subdirectory are several files with numeric names. Each file contains a registration request. The file name is the event command group ID of the group of events within the request.

The `/etc/ha/cfg` Directory (Run-Time EMCDB and Related Files)

The `/etc/ha/cfg` directory contains the run-time EMCDB file for each system partition. The file is called `/etc/ha/cfg/em.domain_name.cdb`, where `domain_name` is the name of the domain.

In addition, there is a file called `em.domain_name.cdb_vers`, where `domain_name` is the name of the domain. This file is created by the Event Manager daemon and contains the version string of the EMCDB file used by the daemon. This file is also used by the RMAPI to ensure that it is using the same EMCDB as the Event Manager daemon.

Components on Which Event Management Depends

The Event Management subsystem depends on the following components:

- | | |
|-------------------------------------|---|
| SPMI | The System Performance Monitor Interface (SPMI) is a library that is included with AIX (in the <code>perfagent.tools</code> file set). This library is used by the RMAPI and the aixos resource monitor. |
| Group Services | Another RSCT subsystem. It provides high availability membership services for coordinating activities on multiple nodes that are used by the Event Management subsystem. |
| System Data Repository (SDR) | A repository of SP system configuration information that includes the configuration information for the Event Management subsystem. |

The sections that follow contain more details about each of these components.

The System Performance Monitor Interface (SPMI) Library

The System Performance Monitor Interface (SPMI) library resides at `/usr/lib/libSpmi.a`. The RMAPI uses the shared memory technology of this library to deliver resource variable values to the Performance Toolbox for AIX product and the Performance Toolbox Parallel Extensions (PTPE) optional feature of PSSP. Thus, a resource monitor supplies data not only for event management, but also for performance monitoring.

Also, the **aixos** resource monitor uses the SPMI to obtain AIX operating system statistics that are the source of the AIX operating system resource variables that are supplied by RSCT.

The Group Services Subsystem

The Group Services subsystem provides another set of high availability services. The Event Management subsystem primarily uses Group Services to monitor the state of each Event Manager daemon in the domain, as follows. Each Event Manager daemon in the domain joins a Group Services group called **ha_em_peers**. Group Services informs each Event Manager daemon, in a synchronized fashion, when a daemon has joined or left the group.

Associated with the **ha_em_peers** group is a group state that contains the version string of the EMCDB. Thus, when an Event Manager daemon joins the **ha_em_peers** group, it can determine the version of the EMCDB that the rest of the group is using.

The Event Manager daemon also subscribes to the **HostMembership**, **enMembership**, and **cssMembership** system groups. Instances of the **IBM.PSSP.Membership.Node.state** resource variable are derived from **HostMembership** group information. Instances of the **IBM.PSSP.Membership.LANAdapter.state** resource variable are derived from **enMembership** and **cssMembership** group information. Instance of the **IBM.PSSP.Response.Host.state** resource variable are derived from **enMembership** group information.

The System Data Repository (SDR)

The Event Manager daemon uses the SDR primarily as a repository of configuration information. In addition, on the control workstation, the Event Manager daemon polls the SDR for information that it uses to create instances of the **IBM.PSSP.Response.Switch.state** resource variable. This information is taken from the **switch_responds** SDR class.

Configuring and Operating Event Management

The following sections describe how the components of the Event Management subsystem work together to provide event management services. Included are discussions of:

- Event Management configuration
- Event Manager daemon initialization
- Event Management operation.

Configuring Event Management

RSCT is installed as part of the installation of the PSSP product. The Event Management subsystem is contained in the **rsct.basic.rte** and **rsct.basic.sp** filesets. The EMAPI libraries are contained in the **rsct.clients.rte** and **rsct.clients.sp** filesets.

After the components are installed, the subsystem must be configured for operation. Event Management configuration is performed by the **haemctrl** command, which is invoked by the **syspar_ctrl** command.

The **syspar_ctrl** command configures all of the system partition-sensitive subsystems. The person who installs PSSP issues the **syspar_ctrl** command during installation of the control workstation. The **syspar_ctrl** command is

executed automatically on the nodes when the nodes are installed. The **syspar_ctrl** command is also executed automatically when system partitions are created or destroyed. For more information on using the **syspar_ctrl** command, see *PSSP Installation and Migration Guide* and *PSSP Command and Technical Reference*.

The **haemctrl** command provides a number of functions for controlling the operation of the Event Management system. You can use it to:

- Add (configure) the Event Management subsystem
- Start the subsystem
- Stop the subsystem
- Delete the subsystem
- "Clean" all Event Management subsystems
- "Unconfigure" all Event Management subsystems
- Turn tracing of the Event Manager daemon on or off

Except for the clean and unconfigure function, **haemctrl** affects the Event Management subsystem in the current system partition, that is, the system partition that is specified by the **SP_NAME** environment variable.

Adding the Subsystem

If the **haemctrl** command is running on the control workstation, the first step in the add function is to select an Event Manager daemon communications port number and save it in the **Syspar_ports** SDR class. This port number is then placed in the **/etc/services** file. If the **haemctrl** command is running on a node, the port number is fetched from the SDR and placed in the **/etc/services** file. Port numbers are selected from the range 10000 through 10100.

The second step is to add the Event Management startup program to the System Resource Controller (SRC) using the **mkssys** command. On the control workstation, the IP address of the system partition is an argument to the **haemd_SP** program in the SRC subsystem specification. The third step is to add the **aixos** resource monitor daemon **harmad** to the SRC using the **mkssys** command.

The fourth step is to add an entry to the **/etc/inittab** file so that the Event Manager daemon and the **aixos** resource monitor will be started during boot. However, if **haemctrl** is running on a High Availability Control Workstation (HACWS), no entry is made in the **/etc/inittab** file. Instead, HACWS manages starting and stopping the Event Manager daemon and the **aixos** resource monitor.

The remaining steps in the add function are performed only on the control workstation. The **haemloadcfg** program is run to load the default configuration data into the SDR. The **haemcfg** command is run to create the EMCDB and place it into the staging directory. Finally, if it is not already stored in the SDR, an Event Manager daemon remote client communications port number is selected from the range 10000 through 10100. This port number is then placed in the **/etc/services** file. This port number is used by all of the Event Manager daemons on the control workstation.

Note that if the **haemctrl** add function terminates with an error, the command can be rerun after the problem is fixed. The command takes into account any steps that already completed successfully.

Starting and Stopping the Subsystem

The start and stop functions of the **haemctrl** command simply run the **startsrc** and **stopsrc** commands, respectively. However, **haemctrl** automatically specifies the subsystem argument to these SRC commands.

Deleting the Subsystem

The delete function of the **haemctrl** command removes the subsystem from the SRC, removes the entry from **/etc/inittab**, and removes the Event Manager daemon communications port number from **/etc/services**. It does **not** remove anything from the SDR, because the Event Management subsystem may still be configured on other nodes in the domain.

Cleaning the Subsystem

The clean function of the **haemctrl** command performs the same function as the delete function, except in all system partitions. In addition, it removes the Event Manager daemon remote client communications port number from the **/etc/services** file.

The clean function does **not** remove anything from the SDR. This function is provided to support restoring the system to a known state, where the known state is the (possibly restored copy of the) SDR database.

Unconfiguring the Subsystem

The unconfigure function of the **haemctrl** command performs the same function as the clean function and then removes all port numbers from the SDR allocated by the Event Management subsystem. This function can only be performed on the control workstation and must be preceded by executing the clean function of the **haemctrl** command on all of the nodes.

The purpose of this function is to remove allocated port numbers from the SDR in a consistent manner.

Tracing the Subsystem

The tracing function of the **haemctrl** command is provided to supply additional problem determination information when it is requested by the IBM Support Center. Normally, tracing should **not** be turned on, because it may slightly degrade Event Management subsystem performance and can consume large amounts of disk space in the **/var** file system.

Initializing Event Manager Daemon

Normally, the Event Manager daemon startup program, **haemd_SP**, is started by an entry in the **/etc/inittab** file using the **startsrc** command. If necessary, you can start the startup program using the **haemctrl** command or the **startsrc** command directly. The startup program performs the following steps:

1. It gets the number of the node on which it is running using the **/usr/lpp/ssp/install/bin/node_number** command. Node 0 is the control workstation.

2. It fetches the name of the system partition and the EMCDB version string from the **Syspar** SDR class. (Recall that one instance of the Event Manager daemon runs on the control workstation for each system partition to which the Event Management subsystem was added.) It also fetches the Event Manager daemon remote client communications port number from the **SP_ports** SDR class.
3. Finally, the startup program invokes the Event Manager program **haemd**, passing the information just collected and any arguments passed to the startup program itself. Note that a new process is not started; the process image is just replaced. This permits the Event Manager daemon to be controlled by the SRC. During its initialization, the Event Manager program performs the following steps:

- a. It performs actions that are necessary to become a daemon. This includes establishing communications with the SRC subsystem so that it can return status in response to SRC commands.
- b. It removes from the registration cache all of the subdirectories for local EM clients that no longer exist. That is, if the process ID in the subdirectory name cannot be found, it removes the subdirectory.

Note that subdirectories for remote clients cannot be removed automatically, because the Event Manager daemon cannot determine if remote processes still exist.

- c. It tries to connect to the Group Services subsystem. If the connection cannot be established because the Group Services subsystem is not running, it is scheduled to be retried in 5 seconds. This continues until the connection to Group Services is established. Meanwhile, Event Manager daemon initialization continues.
- d. It enters the main control loop.

In this loop, the Event Manager daemon waits for requests from EM clients, messages from resource monitors and other Event Manager daemons, messages from the Group Services subsystem, and requests from the SRC for status. It also waits for internal signals that indicate a function that was previously scheduled should now be executed, for example, retrying a connection to Group Services.

However, EM client requests, messages from resource monitors, and messages from other Event Manager daemons (called peers) are refused until the Event Manager daemon has successfully joined the daemon peer group (the **ha_em_peers** group) and has fetched the correct version of the EMCDB.

Joining the Peer Group

After the Event Manager daemon has successfully established a connection with the Group Services subsystem, it tries to join the daemon peer group, a Group Services group called **ha_em_peers**. If this is the first Event Manager daemon to come up in the domain, it establishes the peer group. Otherwise, the other daemons in the peer group either accept or reject the daemon's join request. If an existing peer group member is still recovering from a prior termination of the joining daemon, the join request is rejected. If its join request is rejected, the daemon tries to join again in 15 seconds. This continues until the daemon's join request is accepted.

When it joins the daemon peer group, the Event Manager daemon examines the group state. The group state is the EMCDB version string.

If the group state is null, the joining daemon proposes that the group state be set to the version string that the daemon has fetched from the SDR. If several daemons try to join the group at about the same time, and the group state is null, then each daemon proposes the group state. When the group is formed, Group Services selects one of the proposals and sets the group state to it. Note that each daemon is proposing the EMCDB version string that it has fetched from the SDR. Unless the **haemcfg** command has been run at about the same time, the proposed version strings should be identical.

If the group state is not null when it is examined by the joining daemon, a group has already formed and the daemon does **not** propose a new group state.

After the daemon has successfully joined the peer group, it compares the EMCDB version string contained in the group state to the version string it fetched from the SDR. If they are different, the version that was fetched from the SDR is replaced by the version in the group state.

An Event Manager daemon is prevented from joining the peer group as long as any other Event Manager daemon, currently in the peer group, is non-responsive to “pings” from the Group Services subsystem. (When an Event Manager daemon successfully joins the peer group, Group Services requests a response from the Event Manager daemon every two minutes. If the daemon does not respond to the request within two minutes, it is considered to be non-responsive. A daemon is also considered to be non-responsive if it does not reply to the join requests of other daemons within one minute.) The Event Manager daemon status, as displayed by the **lssrc** command, indicates if a daemon cannot join the peer group. If this is the case, the **em.default.domain_name** file of any other daemon in the peer group should be examined for errors indicating that an Event Manager daemon is non-responsive. If so, and the non-responsive daemon does not terminate itself within a few minutes, perform the User Response specified for the error.

Reading the EMCDB

Once the daemon has joined the peer group and has determined the EMCDB version, it reads the run-time EMCDB file from the **/etc/ha/cfg** directory. If the file does not exist, it is copied from the staging directory on the control workstation.

Once the daemon has read the file, it compares the version string in the EMCDB to the one it fetched (from the SDR or from the group state). If the two version strings do not match, and the daemon has not just copied the EMCDB from the control workstation, then it copies the run-time EMCDB from the control workstation. If the version strings still do not match, the daemon terminates with an error.

Whenever it is necessary to copy the EMCDB from the control workstation, the EMCDB version string is used to determine how the copy is done. If the EMCDB version string was obtained from the group state then it is used to copy a back level EMCDB from the staging directory on the control workstation. Otherwise, the staging file **/spdata/sys1/ha/cfg/em.domain_name.cdb** is copied. Note that back level copies of the EMCDB should be removed from the staging directory only if their version string suffix indicates a time stamp older than the current version string found in the daemon peer group state (the current version string is found in

the Event Manager daemon status, as displayed by the **lssrc** command. See “Displaying the Status of the Event Manager Daemon” on page 355.

After the daemon has read and validated the EMCDB, it enables daemon communications. This permits EM clients to send requests to the daemon, resource monitors to connect to the daemon, and peers to send messages to the daemon. At this point, the initialization of the Event Manager daemon is complete.

To copy the EMCDB from the control workstation to the **/etc/ha/cfg** directory, the Event Manager daemon uses the **/usr/sbin/rsct/install/bin/haemrcpdb** script. This script uses the **rcp** command to perform the actual copy.

The way in which Event Manager daemons determine the EMCDB version has important implications for the configuration of the subsystem. To place a new version of the EMCDB into production (that is, to make it the run-time version that is used by the Event Management subsystem), you must stop each Event Manager daemon in the domain after the **haemcfg** command is run. Stopping the daemons dissolves the existing peer group. Once the existing peer group is dissolved, the daemons can be restarted. As they restart, the daemons form a new peer group. A new EMCDB version string can be submitted as the group state only when a peer group is formed.

Operating the Event Management Daemon

Normal operation of the Event Management subsystem requires no administrative intervention. The subsystem recovers from temporary failures automatically. However, there are some characteristics that may be of interest to administrators.

For performance reasons, the Event Manager daemon has an internal limit of 256 open file descriptors. In practice, this limits the number of EM client sessions, either local or remote, to about 225. This file descriptor limit is per daemon; it does not limit the number of EM clients in the domain.

The Event Manager daemon connects to a resource monitor of type **server** as the resource monitor starts. If a server resource monitor is running prior to the start of the Event Manager daemon, the daemon connects to the resource monitor after enabling daemon communications. If able, the daemon starts a **server** resource monitor when necessary. However, connection and start attempts are constrained under the following circumstances:

1. If it is necessary that the daemon start the resource monitor before each connection attempt, then after three attempts within two hours the resource monitor is “locked” and no further attempts are made.
2. If the resource monitor is not startable by the Event Manager daemon, then after about three successful connections within two hours the resource monitor is “locked” and no further attempts are made.

The rationale for locking the resource monitor is that, if it cannot be started and stay running or successful connections are frequently being lost, then a problem exists with the resource monitor. Once the problem has been determined and corrected, the **haemunlkrm** command can be used to unlock the resource monitor and connect to it, starting it first if necessary. Note that locking does not apply to **client** type resource monitors.

The primary function of the Event Manager daemon is to generate events, by observing resource variable values and applying expressions to those values. However, this function is performed for a resource variable only if an EM client has registered to receive events for that resource variable. The Event Manager daemon also observes a resource variable once to satisfy a query request, if the resource variable is not already being observed. When observations are necessary, the Event Manager daemon commands the appropriate resource monitor (if it has a connection type of **server**) to supply resource variable values. When observations are no longer necessary, the Event Manager daemon commands the resource monitor to stop supplying values. In this way, the Event Manager daemon performs no action for resource variables that are not of interest to clients.

Even if a resource monitor that has a connection type of **server** is running, it does not supply data to the Event Manager daemon except by command of the daemon.

The Event Manager daemon either observes a resource variable located in shared memory every X seconds, where X is the observation interval that is specified in the resource variable's resource class definition, or when the resource variable's value is sent to the Event Manager daemon by the resource monitor (transparently, via the RMAPI). The values of resource variables of value type Counter and Quantity are located in shared memory. The values of resource variables of value type State are not.

All resource variables that are located in shared memory with the same observation interval are observed on the same time boundary. This minimizes the observation overhead, no matter when the request for a resource variable is made.

Event Management Procedures

For the most part the Event Management subsystem runs itself without requiring administrator intervention. However, on occasion, you may need to check the status of the subsystem, or add or change some of the configuration data.

This section contains the procedures that you need to do these tasks, which include:

- Displaying the status of the Event Manager daemon
- Loading non-PSSP configuration data into the SDR
- Changing configuration data in the SDR
- Activating the configuration data in the SDR
- Changing resource variable instance limits

Displaying the Status of the Event Manager Daemon

You can display the operational status of the Event Manager daemon by issuing the **lssrc** command.

On the control workstation, enter:

```
lssrc -l -s haem.domain_name
```

where *domain_name* is the name of the domain of interest.

On a node, enter:

lssrc -l -s haem

In response, the **lssrc** command writes the status information to standard output. The information includes:

- The information provided by the **lssrc -s haem** command (short form)
- The names of any trace flags that are set

For information on these flags, see the **haemtrcon** command in *PSSP Command and Technical Reference*.

- The EMCDB version string and an indication as to whether the version string is taken from the SDR or from the peer group state.
- The day and time the Event Manager daemon was started.
- A report on the daemon's progress through initialization:

```
Daemon connected to group services: TRUE/FALSE
Daemon has joined peer group:      TRUE/FALSE
Daemon communications enabled :    TRUE/FALSE
```

- A count of the peer daemons that are currently in the peer group. The count does not include this daemon.
- A listing of the peer group state.
- The number and type of EM client connections. Note that when a daemon relays a request to another daemon, the sending daemon is treated as a client by the receiving daemon.
- A list of each resource monitor that is defined in the EMCDB and the current status of each, as follows.

A resource monitor may have multiple executing instances, the number of the resource monitor instance is specified in the **Inst** column. The connection type (C=client, S=server, I=internal) is found in the **Type** column. The connection status is indicated by the **FD** column; if the file descriptor is greater than or equal to 0, a connection is open. If a resource monitor has a shared memory segment used to transfer information to the Event Manager daemon, it has a shared memory ID greater than or equal to 0 in the **SHMID** column. The process ID of the resource monitor is listed in the PID column; it is interpreted as follows:

ID greater than 0	Resource monitor has been successfully started by the Event Manager daemon
ID equal to 0	A resource monitor started by the daemon has terminated (or the resource monitor forked, the parent process exited and the child process is the actual resource monitor)
ID equal to -1	The resource monitor has never been started by the Event Manager daemon
ID equal to -2	The resource monitor is not startable by the Event Manager daemon.

The **Locked** column indicates whether or not a resource monitor is locked and the current count of start attempts and successful connections, in the form *mm/nn*, where *mm* is the count of start attempts and *nn* is the count of successful connections.

If a resource monitor has more than one instance, information is present in the **PID** and **Locked** columns only for instance number 0. However, the count of successful connections is for all instances of the resource monitor.

- The highest file descriptor in use.
- The peer daemon status.

This lists the status of peer daemons by node number, in node number order. Note that this list only includes peer daemons that have joined the peer group since the local daemon started.

Following the node number are two characters. If both characters are **S**, the specified node is the number of the node on which this daemon is running. Otherwise, the characters can take on values as follows.

The first character is **I** or **O** where:

- **I** indicates that the peer on the specified node is a peer group member.
- **O** indicates that the peer is no longer a peer group member (but was at one time).

The second character is either **A** or **R**, where:

- **A** indicates that this daemon is accepting join requests from the peer on the specified node.
 - **R** means this daemon is rejecting join requests.
- A list of internal daemon counters, for use by IBM service personnel.

Loading Non-PSSP Configuration Data into the SDR

The default configuration data supplied for the PSSP is normally loaded into the SDR and compiled into its binary format automatically by the **haemctrl** script. However, if resource monitors supplied by other IBM products or by third parties are installed on the SP system, you must load the configuration data supplied with the resource monitors into the SDR and activate it. To do this:

1. Login to the control workstation. Use an ID that has **root** authority.
2. Create a file in load list format with the data to be loaded, or identify the path name of the file that has been supplied.

If you are creating a new file, use the format specified in the man page for the **haemloadlist** file.

3. Set the **SP_NAME** environment variable to the appropriate system partition name.
4. Load the data into the SDR using the **haemloadcfg** command. Enter:

```
haemloadcfg new_loadlist
```

where *new_loadlist* is the path name of the load list you previously created or identified.

5. Activate all of the Event Management data in the SDR, including your new data, using the procedure in “Activating the Configuration Data in the SDR” on page 358.

Changing Configuration Data in the SDR

With an optional flag, the **haemloadcfg** command can replace existing objects (identified by their key attributes) in the SDR. If you want to change an object that already exists, do the following:

1. Login to the control workstation. Use an ID that has **root** authority.
2. Change the object's attribute in the load list file.

For PSSP configuration data, the default configuration data is in the **/usr/sbin/rsct/install/config/haemloadlist** file. You must copy this file to another file and make the changes in the copy. If you have non-PSSP or third-party resource monitors, the object will be loaded from another load list file.

Do this step for as many objects as you are changing.

3. Set the **SP_NAME** environment variable to the appropriate system partition name.
4. Run the **haemloadcfg** command, specifying the **-r** flag and the changed load list file.

In response, the **haemloadcfg** command replaces each object in the SDR matched by an object in the load list file.

5. Activate all of the Event Management data in the SDR, including your changed data, using the procedure in "Activating the Configuration Data in the SDR."

Activating the Configuration Data in the SDR

When you have added or changed the Event Management data in the SDR, you must activate it by recompiling the EMCDB and stopping and restarting the Event Manager daemons. To do this:

1. Login to the control workstation. Use an ID that has **root** authority.
2. Set the **SP_NAME** environment variable to the appropriate system partition name.
3. Recompile the EMCDB. Enter:

```
haemcfg
```

The output is placed in the staging directory in a file whose name indicates the system partition.

4. Stop the Event Manager daemons in this system partition.

Issue the **haemctrl -k** command on the control workstation and on each of the nodes in the system partition.

You can use the **dsh** or **Sysctl** commands to run the command on multiple nodes from the control workstation. For more information on using these commands, see Chapter 3, "Parallel Management Commands" on page 31.

5. Verify that all of the Event Manager daemons in this system partition have stopped.

On the control workstation, issue the **lssrc -s haem.domain_name** command. On the nodes, issue the **lssrc -s haem** command.

You can use the **dsh** or **Sysctl** commands to run the command on multiple nodes from the control workstation. For more information on using these commands, see Chapter 3, “Parallel Management Commands” on page 31.

The status of each daemon should indicate that it is inactive.

6. Restart the Event Manager daemons in this system partition.

Issue the **haemctrl -s** command on the control workstation and on each of the nodes in the system partition.

You can use the **dsh** or **Sysctl** commands to run the command on multiple nodes from the control workstation. For more information on using these commands, see Chapter 3, “Parallel Management Commands” on page 31.

Changing Resource Variable Instance Limits

To change the limit on the number of resource variable instances accepted by the Event Management subsystem for any resource variable class, do the following:

1. Login to the control workstation. Use an ID that has **root** authority.
2. Set the **SP_NAME** environment variable to the appropriate system partition name.
3. Find the load list file that contains the resource variable class definition. Resource variable classes shipped for the PSSP are found in **/usr/sbin/rsct/install/config/haemloadlist**.
4. Find the class definition in the load list file. For example, the class definition of the Recoverable Virtual Shared Disk subsystem is:

```
EM_Resource_Class
  rcClass="IBM.PSSP.VSD"
  rcResource_monitor="IBM.PSSP.harmld"
  rcObservation_interval="60"
  rcReporting_interval="10"
```

5. Either modify the definition in the load list file or copy the definition to a new load list file (which will contain only the modified definition. This is required if the load list file is **/usr/sbin/rsct/install/config/haemloadlist**). Modify the definition by adding the attribute **rcInstance_limit** set to the desired value. For example, the Recoverable Virtual Shared Disk class definition would be changed to:

```
EM_Resource_Class
  rcClass="IBM.PSSP.VSD"
  rcResource_monitor="IBM.PSSP.harmld"
  rcObservation_interval="60"
  rcReporting_interval="10"
  rcInstance_limit="5000"
```

to limit the number of Recoverable Virtual Shared Disk resource variable instances to 5000.

6. Execute the **haemloadcfg** command, specifying the **-r** flag and the name of the modified load list file.
7. Now, follow the procedure documented in “Activating the Configuration Data in the SDR” on page 358.

Chapter 26. Using the Problem Management Subsystem

The Problem Management subsystem (**pman**) provides an infrastructure for recognizing and acting on problem events in your SP system. This infrastructure is based on an Event Management application that provides configurable access to Event Management client and resource monitor function without the necessity of writing C programs that use the Event Management APIs.

For more information, refer to the *RS/6000 SP High Availability Infrastructure*, SG24-4838.

To understand the information presented in this chapter, you should be familiar with Event Management concepts and terminology. Some important Event Management terms are:

- event** In Event Management, the notification that an expression evaluated to true. This evaluation occurs each time an instance of a resource variable is observed.
- expression** In Event Management, the relational expression between a resource variable and other elements (such as constants or the previous value of an instance of the variable) that, when true, generates an event. An example of an expression is $X < 10$ where X represents the resource variable `IBM.PSSP.aixos.PagSp.%totalfree` (the percentage of total free paging space). When the expression is true, that is, when the total free paging space is observed to be less than 10%, the Event Management subsystem generates an event to notify the appropriate application.
- instance vector** An obsolete term replaced with the term **resource identifier**.
- predicate** An obsolete term replaced with the term **expression**.
- rearm expression** In Event Management, an expression used to generate an event that alternates with an original event expression in the following way: the event expression is used until it is true, then the rearm expression is used until it is true, then the event expression is used, and so on. The rearm expression is commonly the inverse of the event expression (for example, a resource variable is on or off). It can also be used with the event expression to define an upper and lower boundary for a condition of interest.
- rearm predicate** An obsolete term replaced with the term **rearm expression**.
- resource** In Event Management, an entity in the system that provides a set of services. Examples of resources include hardware entities such as processors, disk drives, memory, and adapters, and software entities such as database applications, processes, and file systems. Each resource in the system has one or more attributes that define the state of the resource.
- resource identifier** In Event Management, a set of elements, where each element is a name/value pair of the form `name=value`, whose values

uniquely identify the copy of the resource (and by extension, the copy of the resource variable) in the system.

resource monitor A program that supplies information about the resources in the system. It can be a command, a daemon, or part of an application or subsystem that manages any type of system resource.

resource variable In Event Management, the representation of an attribute of a resource. An example of a resource variable is `IBM.AIX.PagSp.%totalfree`, which represents the percentage of total free paging space. `IBM.AIX.PagSp` specifies the resource name and `%totalfree` specifies the resource attribute.

For more information on Event Management services, refer to *RSCT: Event Management Programming Guide and Reference*.

The major components of the Problem Management subsystem are:

- A Problem Management daemon (**pmand**) that provides access to events generated by Event Management on all nodes. In addition, the following function is provided to work in conjunction with **pmand**:
 - A command, **pmandef**, that provides for subscribing to Event Management events and associating actions for those events
 - A command, **pmanquery**, that queries the SDR for a description of a Problem Management subscription
 - A script, **pmandefaults**, that causes **pmand** to register for a set of default events
 - A script, **notify_event**, that mails an event notification when an event occurs
 - A script, **log_event**, that logs a record of an event to a regular wraparound file
- A resource monitor daemon, **pmanrmd**, that provides resource variables to Event Management. The following function is provided to work in conjunction with **pmanrmd**:
 - Sixteen resource variables. The resource variables are named **IBM.PSSP.pm.User_state1** through **IBM.PSSP.pm.User_state16**.
 - A command, **pmanrminput**, that provides for associating values with the sixteen supplied resource variables.
 - A sample file, **pmanrmd.conf**, for configuring the **pmanrmd** daemon.
 - A command, **pmanrmdloadSDR**, for loading the configuration information into the System Data Repository.

This chapter provides information on each of the components of Problem Management.

Understanding the Problem Management Daemon

The **pmmand** daemon is a client of Event Management; it can be configured to register for Event Management events and perform actions when those events occur.

Event Management provides access to events throughout an SP system partition; therefore, **pmmand** can monitor and react to events on the node on which it is running as well as on all other nodes in the system partition and the control workstation.

When you install your system, a **pmmand** daemon is automatically configured on each node in a system partition. Additionally, there is a **pmmand** daemon running on the control workstation for each system partition in the SP system that is running Event Management. When running on a node, the **pmmand** daemon:

- Monitors events occurring on the node on which the daemon is running
- Monitors events on all other nodes in the system partition
- Monitors events not associated with a node, such as frame events, as supplied by Event Management

There are no restrictions on what a **pmmand** daemon can monitor:

- Any number of **pmmand** daemons can monitor and act on a single event
- A single **pmmand** daemon can monitor any number of events locally or remotely
- A single **pmmand** daemon can monitor the same event multiple times and all the actions associated with all the event registrations are taken by the daemon when the event occurs

In Figure 41 **pmmand** daemons are running on a 4 node SP system partition.

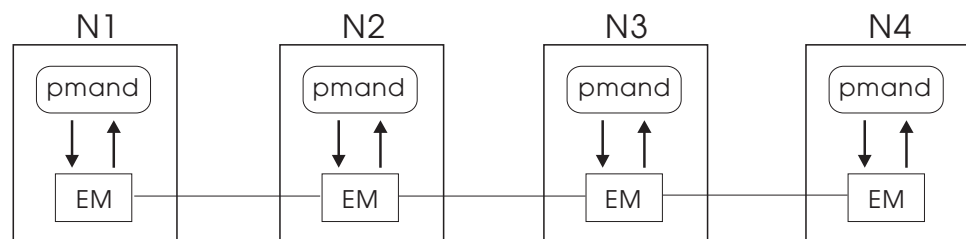


Figure 41. An Example of Problem Management Daemon Configuration

Each **pmmand** daemon has access to events on the node on which it is running, as well as to the other nodes in the system partition. The access to the events is provided by Event Management, which is able, due to its distributed nature, to monitor resource variables throughout the system partition and to generate events based on the values of the resource variables. When an event that any of the daemons has subscribed to occurs (whether that event is local or remote), all the **pmmand** daemons registered for it will perform the actions they are configured for (if any).

Because **pmmand** is a daemon, its subscriptions to events are persistent. That is, the daemon continues to subscribe to events even after the process or user who created the subscription has gone away. A system administrator, for example, can set up automated operations with unattended monitoring and recovery actions.

Controlling pmand

The **pmand** daemon is under the System Resource Controller (SRC) and can be controlled by the following commands:

- To start **pmand** running, issue **startsrc —s** on a node.

```
startsrc -s pman
```

To start **pmand** running, issue **startsrc -s** on a control workstation.

```
startsrc -s pman.system_partition_name
```

- To stop **pmand** running, issue **stopsrc -s** on a node.

```
stopsrc -s pman
```

To stop **pmand** running, issue **stopsrc -s** on a control workstation.

```
stopsrc -s pman.system_partition_name
```

- To refresh **pmand**, issue **refresh -s**.

```
refresh -s pman
```

This causes **pmand** to update its internal configuration from the SDR and start pairing actions with events as specified by the **pmandef** command. If a **pmand** refresh occurs, all currently monitored events will be unregistered from before the configuration information is reread from the SDR. The SDR contains persistent information, so that a refresh results only in configuration changes that have been put into the SDR. If you have not deleted or modified a configuration record for a particular event, refreshing the daemon results in reregistering for the same event. To refresh **pmand** on the control workstation, issue **refresh -s**.

```
refresh -s pman.system_partition_name
```

- To receive status on the **pmand** daemon, issue **lssrc -ls**. To check the status of **pmand** running on a node, enter:

```
lssrc -ls pman
```

To check the status of **pmand** running on the control workstation, enter:

```
lssrc -ls pman.system_partition_name
```

This command provides the following status information:

- When **pmand** was started.
- When **pmand** was last refreshed.
- Whether tracing (debug mode) is on or off. When debug mode is on, all SRC requests and all events are logged to the **/var/adm/SPIlogs/pman** directory.
- Events for which registrations are as yet unacknowledged.
- Events for which actions are currently being taken.
- Events currently ready to be acted on by this daemon.

Creating Problem Management Subscriptions

The **pmandef** command is the mechanism provided for creating Problem Management subscriptions for the **pmand** daemon to Event Management services. The **pmandef** command provides for defining:

- Event Manager events to register for
- Actions to take when those events occur:
 - Run a command
 - Issue an SNMP trap
 - Write to the AIX Error Log and BSD syslog facilities

The **pmandef** command also provides for:

- Activating a Problem Management subscription
- Deactivating a Problem Management subscription
- Querying a Problem Management subscription
- Removing a Problem Management subscription

For more information on **pmandef**, refer to *PSSP: Command and Technical Reference*.

Security Considerations

The **pmandef** command is based on Sysctl, which uses Kerberos for user authentication. All users of the **pmandef** command must have valid Kerberos credentials. In addition, the user's Kerberos principal must be listed in the **/etc/sysctl.pman.acl** file on the local node, in order to store the subscription in the SDR, as well as on all the nodes that are affected by the new subscription, in order for the affected Problem Management daemons to be notified of the new subscription. If the user's Kerberos principal is listed only in the **/etc/sysctl.pman.acl** file on the local node, the subscription will be stored in the SDR, but the Problem Management daemons will not act on the new subscription until the next time they are restarted.

The user's Kerberos principal is more than just a mechanism to validate access to the Problem Management subsystem. The Kerberos principal that was in effect when the subscription was created, via the **pmandef -s** command, is stored as part of the subscription definition, and it is used to establish the ownership of that subscription. Modifications to the subscription, via the **pmandef** command with the **-u**, **-d** and **-a** flags, are allowed only by a user with the same Kerberos principal that is stored in the subscription definition. The Kerberos principal that is stored in the subscription definition is also used by the Problem Management daemon to decide whether the action that is to result from the occurrence of an event should be allowed.

After **pmand** receives notification that an event has occurred, and before it performs the action for that event, it checks to see whether the Kerberos principal that is stored in the subscription definition is authorized to perform the requested action on the node where **pmand** is running. If the requested action is to write an entry in the AIX Error Log and BSD syslog or to generate an SNMP trap, then the Kerberos principal that owns the subscription must have been listed in the root user's **\$HOME/.klogin** file. If the requested action is to execute a command, then the Kerberos principal must be listed in the **\$HOME/.klogin** file of the user which will be used to run the command. The user which will be used to run the command is by default the same user who issued the **pmandef -s** command to create the subscription. A different user may be specified to **pmandef** by using the **-U** flag.

If the requested action is to execute a command, then **pmand** also checks to see whether the system administrator has imposed any AIX login restrictions for the user which will be used to run the command. The **pmand** daemon enforces the same AIX login restrictions as the authenticated rsh service, which in turn enforces the same AIX login restrictions as the AIX rsh service. The AIX user login restrictions that are checked by **pmand**, authenticated rsh and AIX rsh include the following:

- Does the user account exist?
- Has the user's account been locked?
- Is the user allowed to access the node at this time of day?
- Has the user been specifically denied access to the rsh service by including the string "IRSH" in the "ttys" attribute for this user? (You can set the user's "ttys" attribute using the AIX **mkuser** or **chuser** commands.)

Refer to the AIX security documentation for the entire list of user login restrictions which are enforced by the AIX rsh service. Since **pmand** enforces the same AIX login restrictions as authenticated rsh, and since both **pmand** and authenticated rsh require the user's Kerberos principal to be listed in the user's \$HOME/.klogin file, you can determine whether **pmand** will refuse to execute a user's command by answering a simple question: Can this user access the authenticated rsh service on the node where the **pmand** daemon would execute the user's command? If the answer is "yes", then **pmand** will allow the user's command to execute. If the answer is "no", then **pmand** will refuse to execute the command, and it will make note of this in the **pmand** daemon log file, which exists in the directory **/var/adm/SPlogs/pman**. Also keep in mind that AIX user login restrictions do not have to be the same across all nodes in the SP system. Therefore, the answer to this question can be "yes" for some nodes and "no" for other nodes in the same SP system.

Running a Command

Use **pmandef** to specify a command to run when a specified event or rearm event occurs. For example:

```
pmandef -s Program_Monitor \  
-e 'IBM.PSSP.Prog.pcount:NodeNum=12;ProgName=mycmd;UserName=bob:X@0==0' \  
-r "X@0>0" -c "echo program has stopped >/tmp/myevent.out" \  
-C "echo program has restarted >/tmp/myrearm.out"
```

Running this example on node 5 causes the command **echo program has stopped >/tmp/myevent.out** to run on node 5 whenever the number of processes named **mycmd** and owned by user **bob** on node 12 becomes 0 (the event). When this number increases back to 1 (the rearm event), the command **echo program has restarted >/tmp/myrearm.out** runs on node 5.

You can specify if you want the command to run on a node other than the one from which the **pmandef** command was issued. For example:

```
pmandef -s Program_Monitor \  
-e \  
'IBM.PSSP.Prog.pcount:NodeNum=1-5,13;ProgName=mycmd;UserName=bob:X@0==0' \  
-r "X@0>0" -c /usr/local/bin/start_recovery \  
-C /usr/local/bin/stop_recovery -n 1-3,7
```

This example causes the commands to run on nodes 1, 2, 3 and 7, whenever **bob's** program dies or gets restarted on any of nodes 1, 2, 3, 4, 5 or 13. If **bob's** program dies on node 4, then the command `/usr/local/bin/start_recovery` runs on nodes 1, 2, 3 and 7.

Any number of commands can run simultaneously.

You can specify a timeout, in seconds, for each command. The minimum timeout that can be specified is 10 seconds. If the command has not exited before the specified timeout, the command is killed.

For information on command termination status, use the `lssrc -ls` command.

The Command Environment: The Problem Management subsystem makes all of the contents of an Event Management notification available in the command's environment when the command is run:

PMAN_HANDLE

The name that identifies this subscription to the Problem Management subsystem. This name was given as the argument of the `-s` flag to `pmandef`.

PMAN_PRINCIPAL

The name of the Kerberos principal that owns this subscription.

PMAN_RVNAME

The Event Management resource variable.

PMAN_IVECTOR

The Event Management resource identifier.

PMAN_PRED Either the Event Management expression or rearm expression, depending on whether this is an event or rearm event.

PMAN_TIME The time that the event was reported to the Problem Management subsystem.

PMAN_LOCATION

The node number of the node on which the event was generated, usually (but not always) the node on which the event occurred.

PMAN_RVTYPE

One of **long**, **float** or **sbs**, depending on whether the type of the resource variable value is a long integer, a floating point value or a Structured Byte String.

Note: If the `PMAN_RVTYPE` is either **long** or **float**, then the resource variable value is stored in `PMAN_RVVALUE`, and `PMAN_RVVALUE` is to be interpreted as type `PMAN_RVTYPE`.

Note: If `PMAN_RVTYPE` is **sbs**, then the resource variable value is composed of one or more structure elements. There is no `PMAN_RVVALUE` environment variable. Instead there is a separate environment variable for each element, and the `PMAN_RVCOUNT` environment variable defines the number of elements. For example, if there are 3 structure elements within the Structured Byte String, the `PMAN_RVCOUNT` will be 3, and there will be 3 separate environment variables for the 3 structure elements: `PMAN_RVFIELD0`, `PMAN_RVFIELD1` and `PMAN_RVFIELD2`. Each of these 3 environment variables contains a `name=value` pair, where `name` is the structure element name, and `value` is the structure element value.

For example, the following command:

```
pmandef -s example \  
-e 'IBM.PSSP.Prog.pcount:NodeNum=9-11;ProgName=mycmd;UserName=root:X@0==0' \  
-c "/usr/local/bin/recovery_cmd" -n 12
```

requests the **/usr/local/bin/recovery_cmd** command to run on node 12, when the number of processes named **mycmd** and owned by root on nodes 9, 10, or 11 becomes zero. If the **mycmd** program terminates on node 10, the command **/usr/local/bin/recovery_cmd** runs on node 12, and the following environment variables are included in its environment:

- PMAN_HANDLE (**example**)
- PMAN_PRINCIPAL (**root.admin@PPD.POK.IBM.COM**)
- PMAN_RVNAME (**IBM.PSSP.Prog.pcount**)
- PMAN_IVECTOR (**ProgName=mycmd;UserName=root;NodeNum=10**)
- PMAN_PRED (**X@0==0**)
- PMAN_TIME (**Thu Aug 22 00:42:08 1996**)
- PMAN_LOCATION (**10**)
- PMAN_RVTYPE (**sbs**)
- PMAN_RVCOUNT (**3**)
- PMAN_RVFIELD0 (**CurPIDCount=0**)
- PMAN_RVFIELD1 (**PrevPIDCount=1**)
- PMAN_RVFIELD2 (**CurPIDList=**)

This information could be used by any command. Two utilities that report this information are provided as part of the Problem Management subsystem: **notify_event** and **log_event**. (These commands are provided to get you started; you may want to write more sophisticated commands.)

notify_event captures event information and mails it to the user running the command on the local node.

log_event captures event information and logs it to a wraparound file. The syntax for **log_event** is:

```
/usr/lpp/ssp/bin/log_event log_filename
```

log_event uses the AIX **alog** command to write to a wraparound file. The size of the wraparound file is limited to 64K. The **alog** command must be used to read the file. Refer to the AIX **alog** man page for more information on this command.

Issuing an SNMP Trap

Use the **pmandef** command to subscribe to an Event Management event and specify that an SNMP trap be issued for that event. For example:

```
pmandef -s Filesystem_Monitor \  
-e 'IBM.PSSP.aixos.FS.%totused:NodeNum=10;VG=myvg;LV=mylv:X>95' \  
-t 1234 -n 10
```

In this example, whenever the file system associated with the **mylv** logical volume and **myvg** volume group on node 10 becomes more than 95% full, an SNMP trap will be generated on node 10.

For complete information on how **pmand** can be configured to issue an SNMP trap when an event for which it is registered occurs, refer to Chapter 27, “Managing SP System Events in a Network Environment” on page 373.

Logging an Event

You can specify that **pmand** write event notification information, along with some optional specified text, to the AIX Error Log and BSD syslog facilities. For example:

```
pmandef -s Filesystem_Monitor \  
-e 'IBM.PSSP.aixos.FS_%totused:NodeNum=11;VG=myvg;mylv:X>95' \  
-l "filesystem is almost full" -h local
```

In this example, whenever the file system associated with the **mylv** logical volume and **myvg** volume group on node 11 becomes more than 95 % full, the text **filesystem is almost full** gets written to the AIX Error Log and BSD syslog facilities on node 11 (via the **-h local** option).

Requesting Problem Management Subscription Information

Use the **pmanquery** command to query the SDR for a description of a Problem Management subscription. The **pmanquery** command outputs the details of the subscription information in raw format, which can then be used by other applications. The following example queries all subscriptions:

```
pmanquery -n all -k all
```

For more information on **pmanquery**, refer to *PSSP: Command and Technical Reference*.

Monitoring Default Events

The Problem Management subsystem provides for a set of default events to be monitored in the **/usr/lpp/ssp/install/bin/pmandefaults** script. This script contains a series of **pmandef** commands that request an event notification to be mailed to root on the control workstation, when the specified event occurs. These are events that would interest many system administrators. Events are defined for all nodes in the current system partition, and all events are monitored from the control workstation. The list of events includes:

- The **/var** file system is more than 95 percent full.
- The **/tmp** file system is more than 90 percent full.
- An error log record of type **PERM** has been written to the AIX Error Log.
- The **inetd** daemon has terminated.
- The **sdrd** daemon has terminated (control workstation only).
- The **sysctld** daemon has terminated.
- The **hrd** daemon has terminated (control workstation only).
- The **fsd** daemon has terminated (nodes only).

This script is a suggested starting point for configuring the Problem Management subsystem on your SP system. You can choose to run the script as is, or you can make your own copy of the script and modify it to suit your needs, or you can choose not to run the script at all.

The script takes no arguments, and it should be executed as the **admin** Kerberos user (for example, **root.admin@PPD.POK.IBM.COM**). It should be executed once for each system partition; specify the system partition by setting the **SP_NAME** environment variable.

Understanding the Problem Management Resource Monitor Daemon

The **pmanrmd** daemon is a resource monitor daemon that provides resource variables to Event Management services. When you install your SP system, a **pmanrmd** daemon is automatically configured on each node in a system partition. Additionally, there is a **pmanrmd** daemon running on the control workstation for each system partition in the SP system.

Controlling pmanrmd

The **pmanrmd** daemon is under the System Resource Controller (SRC). Only the following SRC commands are available for controlling the **pmanrmd** daemon:

- To start **pmanrmd** running, issue **startsrc -s** on a node.

```
startsrc -s pmanrm
```

To start **pmanrmd** running, issue **startsrc -s** on a control workstation.

```
startsrc -s pmanrm.system_partition_name
```

- To stop **pmanrmd** running, issue **stopsrc -s** on a node.

```
stopsrc -s pmanrm
```

To stop **pmanrmd** running, issue **stopsrc -s** on a control workstation.

```
stopsrc -s pmanrm.system_partition_name
```

No command is provided to refresh the **pmanrmd** daemon while it is running. To refresh the daemon, you must stop and start the daemon (using **stopsrc** and **startsrc**).

Creating Resource Monitors

The Problem Management subsystem provides sixteen resource variables, named **IBM.PSSP.pm.User_state1** through **IBM.PSSP.pm.User_state16**. You can associate values with the provided resource variables and configure **pmanrmd** to supply the values that you associate to Event Management services.

Associating Values with Problem Management Resource Variables

Two mechanisms are provided for associating values with the Problem Management resource variables:

1. The **pmanrminput** command
2. The **pmanrmd.conf** sample configuration file

The **/usr/lpp/ssp/bin/pmanrminput** command requires root authority to use. The command can be run from within scripts. It takes as arguments:

- The name of the SRC subsystem to which to communicate the request
- A string that consists of:
 - The name of one of the Problem Management supplied resource variables
 - A delimiter (+)
 - A string to provide as the value of the Problem Management supplied resource variable
 - Another delimiter (+)

For example, if you have a server application running on node 10 and you sometimes want to tell users on nodes 1-9 to stop using the server application, you could use the Problem Management subsystem to accomplish this. Create a Problem Management subscription on nodes 1-9 to run the **wall** command to inform the users about the state of the server application:

```
pmandef -s ready_or_not \  
-e 'IBM.PSSP.pm.User_state3:NodeNum=10:X@0=="READY"' \  
-r 'X@0!="READY"' \  
-c "wall Node 10 is ready" -C "wall Stop using node 10" -n 1-9
```

When the server application is ready for users, it uses the **pmanrminput** command to communicate its state to the Problem Management subsystem:

```
pmanrminput -s pman -a "IBM.PSSP.pm.User_state3+READY+"
```

The Problem Management subsystem passes the string **READY** to Event Management as the value of the resource variable **IBM.PSSP.pm.User_state3**, and that value satisfies the event definition, so the command **wall Node 10 is ready** runs on nodes 1-9. When the server application has a problem, it can use **pmanrminput** to communicate its state change:

```
pmanrminput -s pman -a "IBM.PSSP.pm.User_state3+NOT READY+"
```

The resource value **NOT READY** satisfies the rearm event definition, so the command **wall Stop using node 10** runs on nodes 1-9.

Another way to associate a value with one of the Problem Management resource variables is to provide the name of the resource variable and a command in a configuration file, based on the **pmanrmd.conf** sample file (for more information on the **pmanrmd.conf** sample file refer to “Configuring pmanrmd to Provide Resource Variables”). The standard output of the command will be provided to Event Management services as the value for the specified Problem Management resource variable.

Configuring pmanrmd to Provide Resource Variables

In the example in “Associating Values with Problem Management Resource Variables” on page 370, the resource, an application server, was able to communicate its state change directly to the Problem Management subsystem. However, this is not always possible. For example, if you want to run a command whenever the contents of the **/etc** directory on nodes 1-5 change, the **/etc** directory is the resource, and a directory is not capable of stating when it has changed. In this case, the Problem Management subsystem needs to be configured to periodically send the name and time stamp of the last changed file in the **/etc** directory as the resource variable value to Event Management. This can be accomplished by defining a resource monitor command to the Problem Management subsystem.

A sample **pmanrmd.conf** file is provided. Copy, rename, and edit this file to specify:

- The node(s) on which **pmanrmd** is to be configured
- The name of the Problem Management resource variable to provide via **pmanrmd**
- A sampling interval (in seconds) for each resource variable

- The command to run to produce the standard output that will be provided as the value for the Problem Management resource variable

For example:

```
TargetType=NODE_RANGE
Target=1-5
Rvar=IBM.PSSP.pm.User_state2
SampInt=600
Command="/bin/ls -tl /etc | /bin/head -2 | /bin/grep -v total"
```

In this example, the **pmanrmd** daemon will run the command **/bin/ls -tl /etc | /bin/head -2 | /bin/grep -v total** every ten minutes, and it will send the output of that command as the value of the resource variable **IBM.PSSP.pm.User_state2** to Event Management. The output of this command is the name of the last file to be changed and its time stamp. You could then subscribe to changes in the **/etc** directory resource with the following command:

```
pmandef -s etc_file_changed \
-e 'IBM.PSSP.pm.User_state2:NodeNum=1-5:X@0!=X@P0' \
-c "echo somebody changed a file in /etc >/tmp/etc_change_log"
```

Whenever the most recent output of the **/bin/ls -tl /etc | /bin/head -2 | /bin/grep -v total** command is different from the previous output, the event definition will be satisfied, and the “**echo somebody changed a file in /etc >/tmp/etc_change_log**” command will get executed.

Loading pmanrmd Configuration Information into the SDR

pmanrmdloadSDR reads the configuration file and loads the information in the stanzas into the SDR. **pmanrmdloadSDR** runs on the control workstation.

The syntax for **pmanrmdloadSDR** is :

```
pmanrmdloadSDR configuration_file
```

After you load the configuration information into the SDR, refresh the affected **pmanrmd** daemons.

Chapter 27. Managing SP System Events in a Network Environment

IBM Parallel System Support Programs for AIX includes a Problem Management subsystem (refer to Chapter 26, “Using the Problem Management Subsystem” on page 361) that provides access to Event Management function without the necessity of writing C programs that use the Event Management APIs. The Problem Management subsystem gives you the ability to subscribe to Event Management events and to specify actions for those events. Issuing an SNMP trap in response to an event (which can come either from Event Management or the AIX Error Log) is one of the actions you can specify.

The ability to issue an SNMP trap in response to an event allows you to report problem events occurring in your SP system to a network manager existing on a remote node (a network manager application is not supplied with the SP). The Problem Management subsystem provides an SP SNMP proxy agent, **sp_configd**, (sometimes referred to as a *subagent daemon*) that runs on the control workstation and every SP processor node. The SP proxy agent provides the following functions:

1. A Management Information Base (MIB)
2. SNMP **GET** and **GET NEXT** command support that allows data in the MIB to be accessed by a network manager application
3. Creation and transmittal of SNMP traps to an installation-defined network manager application when the following events occur on the node on which the SP proxy agent is running:
 - A *cold start* trap is issued when the agent is activated
 - An enterprise-specific trap is issued when an entry with an “alert=true” attribute is written in the AIX Error Log
 - An enterprise-specific trap is issued when an user-specified event is detected within Event Management services

This chapter provides overview and task information for making the SP part of a network management system. In such a system, for example, the SP presents error event information, in the form of SNMP traps, to a network management application, such as NetView for AIX, which displays and logs the trap, and may notify a problem management application, such as Trouble Ticket for AIX, of problem events occurring on the SP.

In order for the SP to be part of such a network management system, network managers must be notified when selected AIX Error Log entries are written and Event Management events occur. SP-specific configuration data is provided so that management applications can determine which nodes compose the SP system. You decide what Error Log and Event Management events will trigger manager notification.

This chapter first discusses general concepts related to Network Management and then discusses information specific to enabling the reporting of SP system events in a network environment. For further discussion of Simple Network Management Protocol and the Management Information Base, refer to *IBM AIX Communications Programming Concepts*, SC23-2610.

Understanding Network Management

The Simple Network Management Protocol (SNMP) describes how management data is packaged and transported through the network in order for SNMP agents and SNMP network managers to communicate. The Management Information Base (MIB) defines the structure of the data being transported.

Understanding SNMP

SNMP uses a client/server approach to management, defining two roles: Manager (client) and Agent (server).

The Manager (NetView for AIX is an example) oversees the overall network activity. Critical events in the form of SNMP traps are sent to this Manager to alert network operators of problems in the network. The Manager also acts as the central point for storing and displaying statistics maintained by the Agent (server). These statistics are maintained by the Agent in the form of MIB variables.

The Agent is responsible for reporting on and maintaining the data pertaining to a managed resource, such as a device or, in the case of the SP Agent, a set of configuration information identifying the nodes composing the SP system and tables of resource attributes within the SP which can be monitored. Agents can run on several different types of managed nodes.

Understanding the MIB

Every Agent supports a Management Information Base (MIB), a set of variables that represent the physical and logical resources of the managed systems or Agent. The MIB is not a database, in the sense of a monolithic collection of data, but rather it represents dynamic information. The values of the variables are maintained by different system functions such as the kernel, device drivers, or subsystems. The Agent obtains the variable values from these system functions.

The Manager can read variable values via these SNMP requests:

- | | |
|-----------------|--|
| GET | Requests the SNMP Agent to retrieve the value of the specified variable and return it to the requester (the Manager). |
| GET NEXT | Requests the SNMP Agent to retrieve the value of the next variable, after the one specified in the request, and return it to the Manager. This is especially useful for retrieving tabular information, and multiple variables can be requested in it. |

To summarize, the SNMP Manager determines the status of Agents by using the GET request to poll the Agents on a regular basis. However, if an Agent discovers an exceptional event, it will alert the Manager immediately, without waiting for a GET request, by sending a trap. The trap does not necessarily describe the problem in any detail; it simply informs the Manager that there is a change of state within the resources managed by the Agent, and the Manager may poll the Agent via an SNMP GET or GET NEXT request to determine more detail about the problem.

Enabling Network Management for the SP System

For an SP to be part of a network management system, network managers must be notified of the AIX Error Log or Event Management events that you would like to centrally manage. A section of the SP proxy agent MIB (contained in the `/usr/lpp/spp/config/snmp_proxy/ibmSPMIB.my` file) contains SP-specific SNMP configuration data, enabling Managers to view the SP as a system composed of processor nodes and a control workstation. In other words, there must be a way to create SNMP traps for SP error or problem events and there must be a MIB that contains configuration information specific to the SP system.

SNMP Traps for the SP from AIX Error Log Events

The `snmp_trap_gen` error notification method notifies the `sp_configd` daemon (or SP proxy agent) to create SNMP traps when selected types of errors are recorded in the AIX Error Log. (Note that the `trappend` method from NetView for AIX also creates SNMP traps from AIX Error Log events. `trappend` formats such events to be specifically compatible with SNA alerts. Only one of these methods should be used to avoid duplicate traps being generated for a single error.) The method and daemon run on each processor node and control workstation composing the SP system. In order for a trap to be created, using the `snmp_trap_gen` method, error log entries on each node and control workstation must contain an “alert=true” attribute. The following steps describe the flow of events for creating an SNMP trap for an AIX Error Log event:

1. An application or subsystem writes to the AIX Error Log facility.
2. If the Error Log entry contains an “alert=true” attribute, the `snmp_trap_gen` error notification method runs.
3. The `snmp_trap_gen` method uses the sequence number from the Error Log entry as input to the `errpt -a` command to obtain full information about the event.
4. `snmp_trap_gen` places the event information in a FIFO file, `/var/tmp/errlog_entry`
5. `sp_configd` reads the FIFO file, parses the information into objects within the `ibmSPErrlogVars` group of the `ibmSP` MIB and creates a trap from the objects whose instantiations contain non-null values.
6. `sp_configd` sends the trap to the AIX agent, `snmpd`, which in turn sends the trap to network managers specified in the `/etc/snmpd.conf` file existing on the node.

The following information is provided in SNMP traps from AIX Error Log events generated using the `snmp_trap_gen` method and the `sp_configd` daemon.

- The `enterprise` field contains the OID (object identifier) of the `sp_configd` subagent (this is the OID assigned to the `ibmSP` MIB)
- The `specific-trap` field contains the error ID from the Error Log entry (this is the convention expected by NetView for AIX when configuring events)
- When the corresponding information exists in the Error Log entry, the `variable bindings` field contains the following object values paired with their OIDs:
 - Error label
 - Error ID
 - Error log entry time stamp

- Unique sequence number
- Machine ID parameter
- Node ID parameter
- Error class
- Error type
- Resource name
- Resource class
- Resource type
- Identifies the location code of a device
- Vital product data
- Error description
- Probable causes
- User causes
- User actions
- Install causes
- Install actions
- Failure causes
- Failure actions
- Detail data

Other fields in the trap are as specified by SNMP protocol.

Note: the values for objects with a syntax of 'DisplayString' are always in the form of English ASCII character strings which are not subject to translation.

Designating AIX Error Log Entries as Alertable

In order for **sp_configd** to pass any entries in the system error log as traps to the **snmpd** daemon for transmission to Network Managers, the Error Record Templates for the entries must have the specific “alert” field set to the value **true**. (Make sure you are authenticated as the root user.) This may be accomplished as follows:

1. On the system in which the **sp_configd** daemon is running, use the **errpt -t** command to list the templates for entries in the AIX Error Log.
2. Select the IDs of the entries for which you wish to have traps issued when they occur.
3. Create a file containing the following two lines for each entry to be designated alertable:


```
= ID:
alert = true
```
4. From the command line, enter:


```
errupdate file
```
5. To verify that the alert status has been changed to “true” (or 1), enter:


```
errpt -tF alert=1 | grep ID
```

 listing the error template. You should see the specified ID listed.

SNMP Traps for the SP from Event Management Events

The following steps describe the flow of events for creating an SNMP trap for an Event Management event:

Designating that an SNMP Trap is Issued for an Event Management Event

1. Use the **pmandef** command to subscribe to an Event Management event and specify that an SNMP trap be issued for that event. For example:

```
pmandef -s Filesystem_Monitor /  
-e 'IBM.PSSP.aixos.FS%totused:NodeNum=10;VG=myvg;LV=mylv:X>95' /  
-t 1234 -n 0
```

(In this example, whenever the file system associated with the **mylv** logical volume and **myvg** volume group on node 10 becomes more than 95% full, an SNMP trap will be generated on the control workstation.)

2. **pmand** (the Problem Management Event Management client daemon) subscribes to the event as specified in the **pmandef** command. The event must occur on the local node on which **pmand** is running.
3. **pmand** writes the contents of the Event Manager subsystem-supplied event response and the user-specified event configuration information into a FIFO file
4. **sp_configd** reads the data and creates an SNMP trap from it
5. **sp_configd** sends the trap to the SNMP managers specified in the **/etc/snmpd.conf** file on the node.
6. You specify a particular trap ID in the configuration information for an Event Manager event.

The following information is provided in SNMP traps from Event Management events generated by the **sp_configd** daemon.

The Contents of SNMP Traps Containing Event Management Events

- The **enterprise** field contains the OID of the **ibmSP** MIB
- The **specific-trap** field contains the trap ID field from the **pmand** configuration for this event. This allows configuration of NetView for AIX actions based on the type of event (which is done based on the specific trap ID)
- The **variable bindings** field contains the following object values paired with their OIDs:
 - The event ID
 - Event flags
 - A time stamp indicating the time the Event Manager generated the event
 - The number of the node and the system partition address where the event occurred
 - The resource variable name
 - MIB table and instance information allowing further information about the variable involved with the event to be obtained from the **ibmSP** MIB.
 - The value of the resource variable that triggered the event
 - The predicate for which the event was triggered

Other fields in the trap are as specified by SNMP protocol.

The MIB for the SP

The **ibmSP** MIB is provided to define SP-specific information. The **ibmSP** MIB consists of an **ibmSPConfig** group, the **ibmSPErrlogVars** group, and the **ibmSPEMVariables** group. The **ibmSP** MIB is instantiated on each SP processor node and the control workstation.

The **ibmSP** MIB is defined in the source file **ibmSPMIB.my** in the **/usr/lpp/ssp/config/snmp_proxy** directory. For a sample of the **ibmSP** MIB file, see “ibmSPMIB.my” on page 452.

The **ibmSPConfig** group defines objects containing SP system configuration information. The group consists of objects containing the following information:

- The node number of the node on which the queried subagent is running
- The IP address of the system partition in which the queried node resides
- The host name of the primary control workstation.
- The operational state of the primary control workstation (up or down)
- The host name of the backup control workstation (will be **NULL** if none exists)
- The operational state of the backup control workstation (up or down)
- The version number of the IBM Parallel System Support Programs for AIX running on the control workstation
- A table of SPNodeEntry's. Each row in the table is indexed by the IP address of a system partition combined with a node number. Each row contains the following information about a node:
 - The IP address of the system partition in which it resides
 - The node number
 - The number of the frame containing the node
 - The lowest slot number in the frame containing the node
 - The number of slots in the frame occupied by the node
 - The reliable host name assigned to the node (this is the host name associated with the SP Ethernet)
 - The initial host name assigned to the node (this is the host name assigned to the node during the customization based on data obtained from the SP Perspectives GUI)
 - The name of the system partition in which the node resides
 - The version number of the IBM Parallel System Support Programs for AIX running on the node

The **ibmSPConfig** group, when instantiated on the control workstation, will identify the default system partition IP address as the IP address of the system partition containing the reporting node and **0** as the node number of the reporting node.

The **ibmSPErrlogVars** group is instantiated on each SP processor node and the control workstation. The group consists of a sequence of objects containing information about the latest Error Log write that caused an SNMP trap to be issued on the reporting node.

The **ibmSPEMVariables** group is instantiated on each SP processor node and the control workstation. The group consists of a sequence of objects containing information about the last Event Management event that caused an SNMP trap to

be issued on the reporting node and tables of objects representing Event Management variables. The group contains up to three tables:

- A table of objects (**ibmSPEMNodeDepVarsTable**) allowing access to the definitions of node-dependent resource variables (those defined with an Event Manager *Locator* attribute). This table contains object instantiations for only those variables which have a Locator value equalling the node number of the node on which the SP proxy agent is running.
- A table of objects (**ibmSPEMNodeIndepVarsTable**) allowing access to the definitions of node-independent resource variables (those defined without an Event Manager *Locator* attribute)
- A table of objects (**ibmSPEMVarValuesTable**) allowing access to the current values assigned to Event Management resource variables

The following information is provided through object instantiations for each of the Event Management resource variables in the **ibmSPEMNodeDepVarsTable** and **ibmSPEMNodeIndepVarsTable** tables:

- Strings containing the following Event Management variable information:
 - Name of the resource variable
 - Resource variable description
 - Variable type (counter, quantity, or state)
 - Data type (long, float, structured byte string)
 - Structure byte string format (only for a structured byte string data type)
 - Initial variable value
 - An index into the **ibmSPEMVarValuesTable**, which, when used in combination with an instantiation vector, provides access to an entry containing the current value of a variable instance
 - Variable's resource class
 - Instantiation vector definition
 - Description of each element in the instantiation vector
 - Name used to read and write the variable in the PTX shared (may be **NULL**) memory
 - Default predicate for event notification (may be **NULL**)
 - A description of the event
 - Locator value specified for the variable
 - Resource order group

The **ibmSPEMVarValuesTable** provides object instantiations for the current values of variables from both the **ibmSPEMNodeDepVarsTable** and the **ibmSPEMNodeIndepVarsTable** tables.

The portion of the **ibmSPEMVariables** group instantiated by the **sp_configd** subagent depends on the type of SP node the subagent is running on. The **ibmSPEMNodeDepVarsTable** and **ibmSPEMVarValuesTable** tables are instantiated on all the processor nodes and the control workstation. The **ibmSPEMNodeIndepVarsTable** table is instantiated on only the control workstation.

Viewing the ibmSP MIB

The AIX **snmpinfo** command may be used in place of a manager client, such as NetView for AIX, to view the contents of the **ibmSP** MIB. Use of this command requires **root** authentication. The following are examples of using **snmpinfo**:

1. To dump the contents of the **ibmSP** MIB located on the host *host_name* in verbose mode, enter the command:

```
snmpinfo -m dump -v -h host_name ibmSP
```
2. To dump the contents of the **ibmSPConfig** group, enter the command:

```
snmpinfo -m dump -v -h host_name ibmSPconfig
```
3. To get values for the MIB variable **SPhostnodenumber.0** located on the local host, enter:

```
snmpinfo -m get ibmSPhostnodenumber.0
```
4. To get the value for the MIB variable following the **ibmSPhostnodenumber** variable, enter:

```
snmpinfo -m next ibmSPhostnodenumber.0
```

Disabling Network Management for the SP System

If the SP is not part of a network management system, you can delete the **sp_configd** subsystem by issuing the following command on each SP processor node and on the SP control workstation:

```
/usr/lpp/ssp/bin/sp_configdctrl -c
```

This will delete the **sp_configd** subsystem from the SRC, and remove **sp_configd** entries from the following files:

- **/etc/inittab**
- **/etc/snmpd.conf**
- **/etc/snmpd.peers**

And it will remove the **ibmSP** MIB definitions appended to:

- **/etc/mib.defs**

Configuring the SP Proxy Agent

The SP proxy agents (**sp_configd**) residing on each managed SP node and control workstation are configured to be started automatically whenever AIX is started. The configuration changes necessary to accomplish this are performed during installation on the nodes and the control workstation.

If you have a management station that “listens” for traps, place the information for the trap destination in the **/etc/snmpd.conf** file on each SP node and control workstation. Instructions for doing this are in the **/etc/snmpd.conf** file. The following steps are performed automatically by the **syspar_ctrl** script on each node and the control workstation when the PSSP software is installed:

1. These steps presume that the standard AIX Agent is already configured to be managed by the desired Managers. If this is not the case, see the information contained in the **/etc/snmpd.conf** file for required updates to the file. This update must be made on every SP node and the control workstation.

2. The `/etc/snmpd.conf` file is modified to add the SP subagent (`sp_configd`) to the set of SMUX peers with a "smux" record
3. The SP MIB file `/usr/lpp/ssp/config/snmp_proxy/ibmSPMIB.defs` is appended to the `/etc/mibs.def` file. The `/usr/lpp/ssp/config/snmp_proxy/ibmSPMIB.defs` file is compiled from `ibmSPMIB.my`
4. Information for the new `sp_configd` subagent is added to the `/etc/snmpd.peers` file. The information is the OID of the Agent and the community name (password).
5. The `snmpd` daemon is refreshed.
6. The daemon is started under SRC control.
 - a. An `src_entry` for `sp_configd` is added to the AIX Object Data Manager (ODM) so that the daemon can be controlled using the `startsrc` command. The `src_entry` invokes the `startsrc` command with the following parameters:


```
-s sp_configd -a "-t 600" -p /usr/lpp/ssp/bin/spconfigd
-u 0 -i /dev/null -o /dev/null -e /dev/null -S -n 15 -F 15
```
 - b. An entry is added to the AIX `inittab` file to start the daemon by issuing the command:


```
/usr/bin/startsrc -s sp_configd
```
 - c. The `sp_configd` daemon supports a `-t` switch that determines the amount of time (in seconds) for which data is to be cached by the daemon and re-fetched when a new request for the data is received. The default value is 60 seconds. The default can be changed in the `src_entry` or the `startsrc` command contained in the `inittab` file, or by entering `startsrc` at the command line.
 - d. The daemon can be stopped by issuing:


```
/usr/bin/stopsrc -s sp_configd
```

Configuring NetView for AIX for the SP System

If you are using NetView for AIX as the network manager for your SP system, you must configure it to recognize SP trap and configuration information.

Configuring the SP MIB to be Used by NetView for AIX

1. Copy the file `/usr/lpp/ssp/config/snmp_proxy/ibmSPMIB.my` from an SP node into `/usr/OV/snmp_mibs` as `ibmSP.mib` on the node running NetView for AIX.
2. If you have dependent nodes attached to your SP system, copy the file `/usr/lpp/ssp/config/spmgrd/ibmSPDepNode.my` from the control workstation into `/usr/OV/snmp_mibs` as `ibmSPDepNode.mib` on the node running NetView for AIX.
3. Load the appropriate MIBs using the NetView for AIX menu option **Load/Unload MIBs**.

Configuring SP Events to be Used by NetView for AIX

Traps generated containing events from SP nodes and the control workstation are displayed automatically at the NetView for AIX **Control Desk** if the node on which the event originated is a managed node. If the node is not managed, the event is recorded in the event history file and is displayed only when you query the event history file. The traps contain object values indicating:

- The trap ID
- The variable name
- The event predicate definition that caused the event
- The value of the variable at the time the event was generated
- Information necessary to use the NetView for AIX browse facility to query the node for the current value of the variable or for more information about the characteristics of the variable and its value.

You may tailor the information in the traps by using the **Trap Customization** selection under the **Event/Configuration** AIX menu option.

Chapter 28. Managing Error Logs

The IBM RS/6000 SP (SP) use both the BSD syslog and AIX Error Logging facilities, as well as a number of function-specific log files to record error events on each node. Refer to the *PSSP: Diagnosis Guide* for an overview of error logging and a description of SP specific log files.

This chapter describes error log management functions provided in the **ssp.sysman** install option of the IBM Parallel System Support Programs for AIX (PSSP). Error log management consists of:

- SMIT panel interfaces
- A set of external commands
 - **splm** for general log viewing, archiving, and service collection
 - **psyslrpt** for generating reports of BSD syslog log files
 - **psyslclr** for trimming BSD syslog log files
 - **penotify** for creating, removing, or displaying Error Notification Objects
- Sysctl-based server functions

This chapter discusses how to perform the following tasks:

- Manage the AIX Error Log facility
- Manage the BSD syslog facility
- Archive error logs
- View error logs
- Collect system data for IBM Service

To access the primary SMIT menu for managing error logs:

TYPE `smit`

- The System Management menu appears.

SELECT RS/6000 SP System Management

- The RS/6000 SP System Management menu appears.

SELECT RS/6000 SP Log Management

- The RS/6000 SP Log Management menu appears.

The fastpath invocation for the Log Management menu is:

```
smit splogmgt
```

Installing and Configuring Error Log Management

Log management functions are built upon the Sysctl facility, which uses the SP authentication services. Therefore, to generate parallel AIX Error Log and BSD syslog reports and view any logs, requires that the user issue the **kinit** command to be identified to the SP authentication services. All other log management commands additionally require that the user be defined as a principal in the **/etc/logmgt.acl** file. All users defined in the file **/etc/logmgt.acl** must also be

placed in the authentication (PSSP or AFS) database as a principal. (Refer to “Adding Principals and Assigning Initial Passwords” on page 158 for more information.) Note that the majority of log management represents administrative tasks normally requiring root authority and that a user defined in the **logmgt.acl** file will execute commands as the root user.

The following is an example of an **/etc/logmgt.acl** file:

```
#acl#
# This sample acl file for log management commands contains
# a commented line for a principal
#_PRINCIPAL root.admin@HPSSL.KGN.IBM.COM
#_ for trimming SPdaemon.log by cleanup.logs.ws
_PRINCIPAL rcmd.k7s
```

The log management server functions executed by Sysctl are located in **/usr/lpp/ssp/sysctl/bin/logmgt.cmds**. During system installation, an include statement for this file is added to the default Sysctl configuration file **/etc/syctl.conf**. If you use an alternate Sysctl configuration file, you must update the file with a statement to include the **logmgt.cmds** file. In addition, you must restart the **sysctld** daemon to pick up this change. Refer to Chapter 13, “Sysctl” on page 177 for a description of the Sysctl function, ACL and AUTH callback authorizations, and using an alternate Sysctl configuration file.

Managing the AIX Error Log Facility

For detailed information on the AIX Error Log facility, refer to:

- *IBM RS/6000 Problem Solving Guide*
- *IBM General Concepts and Procedures for RS/6000*

You can perform the following tasks using either SMIT menus or commands:

- Generate AIX Error Log reports
- Trim AIX Error Log reports
- Configure AIX Error Logs
- Manage error notification objects
- Manage error templates

When you execute commands related to AIX Error Log management you can specify whether the command should be executed on all nodes in the current system partition or you can specify node names or the name of a file containing a list of node names. The default is the local node.

From SMIT

To access the AIX Error Log SMIT menu, enter:

```
smit sperrlog
```

Generating AIX Error Log Reports

You can generate reports on entries in the AIX Error Log on a number of nodes. The report can be displayed or written to a file on the local node.

From SMIT

The fastpath invocation for the Generate an Error Report menu is:

```
smit perrpt
```

Trimming AIX Error Logs

You can trim records from error logs on a set of nodes.

From SMIT

The fastpath invocation for the Clean the Error Log menu is:

```
smit perrclear
```

Configuring the AIX Error Log

You can display the configuration parameters of the AIX Error Log to the local node.

From SMIT

The fastpath invocation for the Show Characteristics of the Error Log menu is:

```
smit perrdemon_shw
```

You can alter one or more of the configuration parameters for the AIX Error Log on a set of nodes. Because of the additional entries generated by SP system software, you should set the AIX Error Log file size to be a minimum of 4MB.

From SMIT

The fastpath invocation for the Change Characteristics of the Error Log menu is:

```
smit perrdemon_chg
```

Managing Error Notification Objects

Error Notification Objects are ODM objects held in the class `errnotify` that are used by the AIX Error Notification Facility to invoke methods upon occurrence of an error event. Fields in the `errnotify` class match to fields in an Error Template for selection. If an error is logged matching the selection criteria defined in a notification object, the method associated with that object is invoked. For more information on using the AIX Error Notification Facility, refer to *IBM General Concepts and Procedures for RS/6000*.

You can add, remove, and show notification objects in parallel on the SP system.

From SMIT

- The fastpath invocation for the Add a Notification Object menu is:

```
smit padd_en
```
- The fastpath invocation for the Remove a Notification Object menu is:

```
smit prem_en
```
- The fastpath invocation for the Show a Notification Object menu is:

```
smit pshw_en
```

From the Command Line

To add, remove, or show error notification objects in parallel on the SP system, enter:

```
penotify -f show
```

For complete details on the command, refer to the *PSSP: Command and Technical Reference*.

Following is a description of the actions taken by the notification method `EN_pend` located under the `/spdata/sys1/err_methods` directory. This method can be installed and used to invoke pre- and post- action scripts and mail a report of the logged error. This script provides a suggested structure for notification methods and can be reused with different pre- or post- action scripts as described in the sections.

EN_pend Method Flow

1. `EN_pend` looks under the directory it resides in for a file with the same name and a `.envs` suffix. If found, it sources the file to pick up environment variables `EN_RUNDEFAULT` and `EN_MAILLOC`. An `EN_pend.envs` script is installed under the same directory.
2. `EN_pend.envs` sets the `EN_RUNDEFAULT` environment variable. It also sets the `EN_MAILLOC` to root at the control workstation, if possible, or to root at the local node.
3. `EN_pend` checks for a pre-action script under the same directory and name with a `.pre` suffix and executes it if found.
4. `EN_pend` mails an expanded report of the error using the sequence of the error passed by the notification facility to the `EN_MAILLOC`, if `EN_RUNDEFAULT` and `EN_MAILLOC` variables are set.
5. `EN_pend` checks for a post-action script under the same directory and name with a `.post` suffix and executes it if found.

Installing a Notification Object

To add the `EN_pend` method to all nodes in the current partition to send a report whenever an error of type PENDING (loss of availability of a device is imminent) occurs, enter:

```
penotify -a -n "PEND_err" -P -t "PEND" -m \  
'/spdata/sys1/err_methods/EN_pend $1'
```

```
penotify -a -n "pend_err" -P -t "pend" -m \  
'/spdata/sys1/err_methods/EN_pend $1'
```

```
penotify -a -n "Pend_err" -P -t "Pend" -m \  
'/spdata/sys1/err_methods/EN_pend $1'
```

The `-P` flag will cause the object to persist after the system is restarted. Three objects are added with variations on PENDING because upper case is not always adhered to by all AIX LPPs and vendor functions. The `$1` argument causes the Error Notification Facility to pass the error sequence number to the notify method.

The **EN_pend** and **EN_pend.envs** scripts can be used to invoke different pre- and post- action scripts for different error events by creating links to them. **EN_pend** looks for **.envs**, **.pre** and **.post** scripts under the directory it is called from, and by the same basename. For example, to use **EN_pend** for reporting hdisk0 errors on nodes h0, h1, h2 and h3 and perform pre- and post- action:

1. Create **.pre** and **.post** action scripts on one of the nodes, for example, **EN_hdisk0.pre**, **EN_hdisk0.post** under the **/spdata/sys1/err_methods** directory on node h0.

2. Copy the pre- and post- scripts to nodes h1, h2 and h3 using **pcp**:

```
pcp -w h1,h2,h3 EN_hdisk0.pre /spdata/sys1/err_methods/  
pcp -w h1,h2,h3 EN_hdisk0.post /spdata/sys1/err_methods/
```

3. Create links to the **EN_pend** and **EN_pend.envs** scripts:

```
dsh -w h0,h1,h2,h3 ln -s /spdata/sys1/err_methods/EN_pend \  
/spdata/sys1/err_methods/EN_hdisk0
```

```
dsh -w h0,h1,h2,h3 ln -s /spdata/sys1/err_methods/EN_pend.envs \  
/spdata/sys1/err_methods/ EN_hdisk0.envs
```

4. Add the notification object:

```
penotify -w h0,h1,h2,h3 -f add -P -n "hdisk0_err" \  
-m '/spdata/sys1/err_methods/EN_hdisk0 $1' -N  
"hdisk0"
```

To display the notification object created, enter:

```
penotify -w h0,h1,h2,h3 -f show -n "hdisk0_err"
```

To remove this notification object, enter:

```
penotify -w h0,h1,h2,h3 -f remove -n "hdisk0_err"
```

Note: Notification methods need to be accessible to each node that the notify object is added to. We suggest that the notification scripts be kept local on each node in case of network failure. File collections should be used to maintain updates to notification methods.

Managing Error Templates

You can create a new error template in the Error Template repository for logging errors.

From SMIT

The fastpath invocation for the Add an Error Template menu is:

```
smit padd_et
```

You can remove a template from the Error Template Repository.

From SMIT

The fastpath invocation for the Remove an Error Template menu is:

```
smit prem_et
```

You can display entries from the Error Template Repository to the local host.

From SMIT

The fastpath invocation for the Show an Error Template menu is:

```
smit pshw_et
```

Managing the BSD syslog Facility

SP error logging utilizes the BSD syslog facility for recording error events.

From SMIT

To access the Syslog SMIT menu, enter:

```
smit spsyslog
```

Generating Reports on BSD syslog Log Files

The **syslogd** daemon, which logs the errors, is configured with a file designating filters for the incoming messages to determine their destination. The default configuration file is **/etc/syslog.conf**. BSD syslog errors are classified by the facility that is issuing the error, and by the error's priority value. Refer to **syslogd** in the *IBM AIX Commands Reference*, SBOF-1851, for a list of facility and priority values. Entries in the configuration file determine the destination for each error message based on these values. Destinations can be a file, user ID, or the **syslogd** daemon on another machine. We suggest that error messages be logged locally rather than forwarded to a remote **syslogd** because of the increased network traffic. File collections can be used to maintain consistent configuration of the syslog facility.

The format of a syslog log file is:

```
MMM DD HH:MM:SS node_name resource[pid]: msg
```

Where:

MMM DD HH:MM:SS Timestamp (month day hour:minute:second)

node_name The name of the node that the error occurred on

resource The name of the failing system

pid The optionally logged process ID of the failing resource.

msg A free form error message.

Errors logged by SP components will contain in the message section the following additional information pertaining to the logging resource:

LPP LPP name

Fn filename

SID SID_level_of_the_file

L# Line number or function

From SMIT

The fastpath invocation for the Generate a Syslog Report menu is:

```
smit spsyslrpt
```

From the Command Line

To report to the local node all records logged by **ftp** for all syslog log files on nodes in the current system partition, enter:

```
psyslrpt -a -r ftp
```

To report all records that logged to files selected for the daemon and user facilities starting on March 3 and to report the records to the local node from nodes host1 and host2, enter:

```
psyslrpt -w host1,host2 -f user,daemon -s 03030000
```

For complete details on the **psyslrpt** command, refer to the *PSSP: Command and Technical Reference*.

Trimming BSD syslog Log Files

The size of syslog log files is not configurable and will continue to grow until manually trimmed.

From SMIT

The fastpath invocation for the Trim Syslog Log Files menu is:

```
smit spsyslclr
```

From the Command Line

To trim all records older than 30 days from the log file **/var/adm/msg**s on the local node, enter:

```
psyslclr -y 30 -l /var/adm/msg
```

To trim all records from all log files found in the alternate syslog configuration file **/var/adm/syslog.conf**, enter:

```
psyslclr -g /var/adm/syslog.conf -y 0
```

Note: The **syslogd** daemon is stopped during the trimming process, then restarted with either the default configuration file or an alternate file, if used (**-g** option). For complete details on the **psyslclr** command, refer to the *PSSP: Command and Technical Reference*.

psyslclr can be added as a **crontab** entry for performing scheduled syslog trimming. On the control workstation, **psyslclr** is used to trim daemon facility messages older than six days. This is done in **/usr/lpp/ssp/bin/cleanup.logs.ws**, which is run from the control workstation's **crontabs** file. (Refer to Chapter 11, "Maintaining the crontabs File" on page 127 for more information.)

Viewing Error Logs

You can view error log data.

From SMIT

To access the General Log Viewing SMIT menu, enter:

```
smit splogview
```

From the Command Line

To cause the contents of each file entry and output of each command entry on the target nodes in the log table **amd.tab** to be displayed to stdout of the node the **splm** command is executed on, enter:

```
splm -a view -t /spdata/sys1/logtables/amd.tab
```

For complete details on the **splm** command, refer to the *PSSP: Command and Technical Reference*.

Archiving Error Logs

Error log archives can be created, removed, or gathered to a central location where they can be optionally written to a tape device or mailed to another location.

From SMIT

To access the Archive Logs SMIT menu, enter:

```
smit sparchive
```

Creating Archives

From SMIT

The fastpath invocation for the Create Archives menu is:

```
smit spcreate_archive
```

From the Command Line

To create an archive on each target node under directory **/var/archives/arch_weekly.tab**, enter:

```
splm -a archive -t /spdata/sys1/logtables/weekly.tab -c -d /var/archives
```

Removing Archives

From SMIT

The fastpath invocation for the Remove Archives menu is:

```
smit spremove_archive
```

From the Command Line

To remove all files and directories under and including **/var/archives/arch_weekly.tab**, enter:

```
splm -a archive -t /spdata/sys1/logtables/weekly.tab -r -d /var/archives
```

Gathering Archives

From SMIT

The fastpath invocation for the Gather Archives menu is:

```
smit spgather_archive
```

From the Command Line

To gather the compressed tar files created on each target node under directory **/var/archives/arch_weekly.tab** to the directory **/var/logrepos** on the local node.

```
splm -a gather -k archive -t /spdata/sys1/logtables/weekly.tab \  
-d /var/archives -l /var/logrepos
```

For complete details on the **splm** command, refer to the *PSSP: Command and Technical Reference*.

Collecting System Data for IBM Service

Service collections can be created, removed and gathered in the same manner as the archive functions with the addition of interacting with the AIX command **snap** to gather additional system information. The default top level directory for service collections is **/tmp** and this path will end with **svrc_log_table_name**.

From SMIT

To access the Collect Logs for Service SMIT menu, enter:

```
smit spcollect
```

Creating Service Collections

From SMIT

The fastpath invocation for the Create Service Collections menu is:

```
smit spcreate_collect
```

From the Command Line

In the following example service collections are created using table **ssp.tab**. **snap -gfk** will be called to gather general system information, file system information and kernel information. Additional log and system data designated in the **ssp.tab** table will be collected under the other directory created by **snap**. The **nodename.tar.Z** compressed tar file will be created.

```
splm -a service -t /spdata/sys1/logtables/ssp.tab -p 'gfk' -c
```

Removing Service Collections

From SMIT

The fastpath invocation for the Remove Service Collections menu is:

```
smit spremove_collect
```

From the Command Line

In the following example, the service collections that were created using table **ssp.tab** are removed:

```
splm -a service -t /spdata/sys1/logtables/ssp.tab -r
```

Gathering Service Collections

From SMIT

The fastpath invocation for the Gather Service Collections menu is:

```
smit spgather_collect
```

From the Command Line

In the following example the compressed tar files from the nodes in table **ssp.tab** are gathered to the local node and written to tape device **/dev/rmt0**. The **-s** flag specifies to stagger the process by gathering the file from one node, writing it to tape, then removing it before moving onto the next node. This provides a method for gathering all the compressed tar files to a tape or disk device, or mailing them, without requiring disk space on the local node for all the service tar files.

```
splm -a gather -k service -t /spdata/sys1/logtables/ssp.tab -d /tmp -s -o /dev/rmt0
```

For complete details on the **splm** command, see the book *PSSP: Command and Technical Reference*.

For information on the **snap** command, see the book *IBM AIX Commands Reference*.

The Communications Low-Level Application Programming Interface

Chapter 29. Understanding and Using the Communications Low-Level Application Programming Interface (LAPI)

This chapter introduces you to the concepts and use of the Communications Low-level Application Programming Interface (LAPI).

Introducing the LAPI

The LAPI is a non-standard application programming interface designed to provide optimal communication performance on the SP Switch. It is based on an *active message* programming mechanism that provides a one-sided communications model (that is, one process initiates an operation and the completion of that operation does not require any other process to take a complementary action). The LAPI library provides the functions **PUT**, **GET**, and a general *active message* function that allows programmers to supply extensions by means of additions to the notification handlers. The LAPI is designed for use by libraries and power programmers for whom performance is more important than code portability.

Why Use the LAPI?

The LAPI provides the following advantages:

- Performance – The LAPI provides basic function for optimal performance. It is designed to especially provide low latency on short messages.
- Flexibility – The LAPI's one-sided communications model provides flexibility because the completion of an operation by one process does not require any other process to take a complementary action. Also, the LAPI provides a more primitive interface (than either MPI or IP) to the SP Switch, giving the programmer the choice of how much additional communications protocol needs to be added.
- Extensibility – The LAPI supports programmer-defined handlers that are invoked when a message arrives. Programmers can customize the LAPI to their specific environments.

General characteristics provided by LAPI are:

- Reliability – The LAPI provides guaranteed delivery of messages. Errors not directly related to the application are not propagated back to the application.
- Flow control
- Support for large messages
- Non-blocking calls
- Interrupt and polling modes
- Efficient exploitation of switch function
- By default, ordering is not guaranteed

General LAPI Functions

LAPI functions are divided into three parts:

1. A basic *active message* infrastructure that allows programmers to install a set of handlers that are invoked and executed in the address space of a target process on behalf of the process originating the active message. This generic interface allows programmers to customize the LAPI function to their unique environment.
2. A set of defined functions that is complete enough to satisfy the requirements of most programmers. These defined functions make the LAPI more usable and at the same time lend themselves to efficient implementation because their syntax and semantics are known.
3. A set of control functions for the initialization and eventual orderly shutdown of the LAPI layer.

Understanding the LAPI

To help you achieve a fuller understanding of the LAPI, this section presents further details on the active message infrastructure and the defined set of functions. In addition, concepts important to understanding the LAPI are explained.

The Active Message Infrastructure

The underlying infrastructure that was selected for the LAPI is referred to as the active message. It has the following characteristics:

- The active message includes the address of a user-specified handler. When the active message arrives at the target process, the specified handler is invoked and executes in the address space of the target process.
- The active message optionally may also bring with it the user header and data from the originating process.
- Operations are unilateral in the sense that the target process does not have to take explicit action for the active message to complete.
- Storage buffers for arriving data need to be provided by the invoked handler.

Writing Handlers

The ability for programmers to write their own handlers provides a generalized, yet efficient, mechanism for customizing the interface to one's specific requirements. The user is responsible for protecting shared structures and buffers where necessary by using the locking structures available in the AIX p-threads library.

The LAPI supports messages that can be larger than the size supported by the underlying SP Switch subsystem. Therefore, the data sent with the active message may arrive at the target in multiple packets and, further, these packets can arrive out of order. This situation places some requirements on how the handler is written.

When the active message brings with it data from the originating process, the architecture requires that the handler be written as two separate routines:

1. A *header handler* function. This function is specified in the active message call. It is called when the message first arrives at the target process and provides the LAPI dispatcher (the part of the LAPI that deals with the arrival of messages and invocation of handlers) with the address of where to copy the

arriving data, the address of the optional *completion handler*, and a pointer to the parameter that is to be passed to the completion handler.

2. A *completion handler* that is called after the whole message has been received.

An Example of LAPI Active Message Function

In this example, a programmer writes a handler for the LAPI active message interface. See the book *PSSP: Command and Technical Reference* for more information on the **LAPI_Amsend** subroutine.

1. The desired function (accumulate) is to add a vector (S) to another (D) on the target node and put the results in the vector at the target:

$$D[0..N-1] = D[0..N-1] + S[0..N-1]$$

where, S[N] is a vector of length N in the address space of the origin process (*origin_process*) D[N] is a vector of length N in the address space of the target process (*target_process*)

2. The generic active message call is defined as **LAPI_Amsend** (*hndl, tgt, hdr_hdl, uhdr, uhdr_len, udata, udata_len, tgt_cntr, org_cntr, cmpl_cntr*)
3. Before making the active message call, you must obtain the address of the target counter (*target_cntr_addr*) and the address of the header handler to be executed on the target process (*accumulate_addr*). The address of the header handler is obtained by the **LAPI_Address** function.
4. Initialize the *uhdr* based on the header expected by *accumulate*. For example, the structure of *uhdr* could be:

```
typedef struct {
    void *target_addr;
    uint length;
} put_add_hdr_t;
```

```
put_add_hdr_t uhdr;
```

```
uhdr.target_addr = D;
uhdr.length = N;
```

5. Make the specific call

```
LAPI_Amsend (hndl, target_process, accumulate_addr,
&uhdr, sizeof(put_add_hdr_t), &S[0],
N*sizeof(S[0]), target_cntr_addr, &origin_cntr, &completion_cntr)
```

6. When this message is received at the target (assuming that the entire origin data is contained within a packet), the *accumulate* handler you specified is invoked by the dispatcher. The structure of the header handler is:

```
void *header_handler (lapi_handle_t hndl, void *uhdr,
    uint uhdr_len, uint msg_len, completion_handler_t,
    *completion_handler, void *user_info)
```

The structure of the completion handler is:

```
void completion_handler (lapi_handle_t hndl,
    void *user_info)
```

7. If any state information about the message is required by the completion handler, the information required must be saved in a user buffer by the header handler. The header handler passes the address of this buffer to the dispatcher

through the parameter *user_info*. The dispatcher uses this pointer as a parameter (*user_info*) for the completion handler.

8. For this example operations performed at the target process are:

- Within the dispatcher:

- a. The LAPI header is read.
- b. *uhdr* and *uhdr_len* are extracted from the LAPI header.
- c. The header handler is invoked.

```
buf = (*accumulate_addr)(hdl, uhdr, uhdr_len, msg_len,
                        &completion_handler, &user_info);
```

- d. *udata* is copied into *buf*.
- e. The completion handler is invoked.

```
(*completion_handler)(&hdl, user_info);
```

Note: If the message was not contained within a packet, the LAPI layer will save the necessary information and will invoke the completion handler after all the *udata* has arrived and copied into *buf*

- User-defined functions:

- Header handler:

```
accumulate(hdl, uhdr, uhdr_len, msg_len, completion_handler, user_info)
{
    buf = addr where incoming data should be buffered
    save (target_addr=D, length=N, buf) in user_info
    completion_handler = complete_accumulate
    return buf
}
```

- Completion handler:

```
complete_accumulate (hdl, user_info)
{
    retrieve required data (namely D,N and buf) from user_info;
    for (i=0; i<N; i++) D[ii]=D[i]
        + buf [i];
    return
}
```

The accumulate handler is the header handler and is called by the LAPI layer when the message first arrives at the target process. The header handler saves the information required by *complete_accumulate* (*target_addr*, *length*, and *buf*) in *user_info* and passes back a pointer to the *complete_accumulate* handler as *user_info*. Additionally, the header handler returns address of a buffer *buf*.

Large active messages are generally transferred as multiple packets. In this case the LAPI layer stores the incoming data in the packets as they arrive into *buf*. When all the data has been received, it calls the *complete_accumulate* function which uses *user_info* to access the two vectors, adds them and stores them at the desired location. After the return from the *complete_accumulate* routine, the LAPI layer increments *tgt_ctr*. The *origin_cntr* is incremented when it is safe to return the origin buffer back to the user.

The *cmpl_cntr* is incremented after the completion handler has completed execution. The *cmpl_cntr*, therefore, is a reflection, at the origin, of the *tgt_cntr*.

The Defined Set of Functions

Fundamentally, the defined set of functions for the LAPI provides a Remote Memory Copy (RMC) interface. The primary characteristics of the defined set of functions provided by LAPI are:

- The basic data transfer operations are memory to memory copy operations that transfer data from one virtual address space to another virtual address space.
- The operations are unilateral. That is, one process initiates an operation and the completion of the operation does not require any other process to take some complementary action. (This is unlike a send and receive operation, where a send requires a complementary receive with matching parameters to be posted for completion.)
- The operations support both “pull” and “push”. The **LAPI_Get** operation copies data from the address space of the target process into the address space of the origin process. The **LAPI_Put** operation copies data into the address space of the target process from the address space of the origin process.
- The initiating process specifies the virtual address of both the source and destination of the data (unlike a send and receive process where each side specifies the address in its own address space). To avoid the limitation of requiring that the address maps on the different processes be identical, the LAPI provides the **LAPI_Address_init** mechanism by which the different communicating processes can exchange information regarding the address map of shared data objects.
- Because data transfer operations are unilateral and no synchronization between the two processes is implied, additional primitives are provided for explicit process synchronization when it is necessary for program correctness.
- Functions are provided to detect completion and to enforce ordering.

Important LAPI Concepts

To use the LAPI, it is important to understand the following concepts:

- Origin and target
- Blocking and non-blocking calls
- Completion of communication operation
- Message ordering and atomicity
- Error handling
- Progress

Origin and Target

Origin denotes the task (or process or processor) that initiates a LAPI operation (**PUT**, **GET**, or *active message*). *Target* denotes the other task whose address space is accessed. Although multiple tasks may run on a single node, it is convenient to think of each task as running on a different node. Therefore the origin task may also be referred to as the *origin node* and the target task as the *target node*. The origin and target can be the same for any of the calls, but if the origin and target data areas overlap, the results are undefined.

Blocking and Non-Blocking Calls

A blocking procedure is one that returns only after the operation is complete. There are no restrictions on the reuse of user resources.

A non-blocking procedure is one that may return before the operation is complete and before the user is allowed to reuse all the resources specified in the call. A non-blocking operation is considered to be complete only after a completion testing function, such as **LAPI_Waitcntr** or **Lapi_Getcntr**, indicates that the operation is complete.

Completion of Communication Operation

A communication operation is considered to be complete, with respect to the buffer, when the buffer is reusable.

A **PUT** is complete with respect to the *origin* buffer when the data has been copied out of the buffer at the origin and may be overwritten. A **GET** is complete with respect to the *origin* buffer when that origin buffer holds the new data that was obtained by **GET**.

A **PUT** is complete with respect to the *target* buffer when the new data is available at the target buffer. A **GET** is complete with respect to the *target* buffer when the data has been copied out of the buffer at target and the target task may overwrite that buffer.

Communication Behaviors: Two communication behaviors support two different definitions of “completion”:

- In **standard** behavior, a communication operation is defined as complete at the *origin* task when it is complete with respect to the *origin* buffer; it is complete at the *target* task when it is complete with respect to the *target* buffer.
- In **synchronous** behavior, a communication operation is defined as complete at the *origin* task when it is complete with respect to both the *origin* buffer and *target* buffer. It is complete at the *target* task when it is complete with respect to the *target* buffer.

The LAPI defines both standard and synchronous behaviors for **PUT** operations. The LAPI defines only synchronous behavior for **GET** operations.

Message Ordering and Atomicity

Two LAPI operations that have the same *origin* task are considered to be ordered with respect to the *origin* if one of the operations starts after the other has completed at the *origin* task. Similarly, two LAPI operations that have the same *target* task are considered to be ordered with respect to the *target* if one of the operations starts after the other has completed at the *target* task. If two operations are not ordered, they are considered concurrent. The LAPI provides no guarantees of ordering for concurrent communication operations. The LAPI does provide mechanisms which an application can use to guarantee order.

As an example, consider the case where a node issues two standard behavior **PUT** operations to the same target node, where the targets overlap. These two operations may complete in any order, including the possibility of the first **PUT** overlapping the second, in time. The contents of the overlapping region will be undefined, even after both **PUTs** complete. Using synchronous behavior for both **PUT** operations, (waiting for the first to complete before starting the second) will

ensure that the overlapping region contains the result of the second after both **PUTs** have completed.

Error Handling

If an error occurs during a communications operation, the error may be signaled at the origin of operation, or the target or both. Some errors may be caught before the communication operation begins, and these will be signaled at the origin. However, some errors will not occur until the communication is in progress (a segmentation violation at the target, for example); these may be signaled at either or both ends of the communication.

Progress

All LAPI operations are unilateral by default and can complete successfully or fail, independent of the actions of other tasks. Specifically, a LAPI operation to a particular target should complete even if the target is stuck in an infinite loop (that is, when the target process is in interrupt mode).

Using the LAPI

In general, LAPI functions:

- Are non-blocking calls
- Provide both polling and interrupt mode
- Signal completion by incrementing counters at each end
- Provide both C and Fortran bindings

Complementary functions provide for checking completion of operations and for enforcing relative ordering if required. Additionally, functions allow processes to exchange addresses that will be used in LAPI operations.

Specific LAPI Functions

The LAPI provides the following specific functions.

Active Message

The active message function (**LAPI_Amsend**) is a non-blocking call that causes the specified active message handler to be invoked and executed in the address space of the target process. Completion of the operation is signaled if counters are specified. Both standard and synchronous behaviors are supported. The **LAPI_Amsend** function provides two counters (*org_cntr* and *cmpl_cntr*) which can be used to provide the two behaviors. The *org_cntr* is incremented when the origin buffer can be reused (standard). The *cmpl_cntr* is incremented after the completion handler has completed execution (synchronous).

Data Transfer

Data transfer functions are non-blocking calls that cause data to be copied from a specified region in the origin address space to the specified region in the target address space (in the case of a **LAPI_Put** operation) or from a specified region in the target address space to a specified region in the origin address space (in the case of a **LAPI_Get** operation). Completion of the operation is signaled if counters are specified. Both standard and synchronous operations are supported for **PUT**. Only synchronous operation is possible in the case of **GET**. Standard **PUT** is provided by incrementing the *org_cntr* when the origin buffer can be reused.

Synchronous **PUT** is provided by incrementing the *cmpl_cntr* after the data has been written into the target buffer. The **LAPI_Getcntr** (or **LAPI_Waitcntr**) function should be used in conjunction with the *org_cntr* and *cmpl_cntr* counters to guarantee the respective standard and synchronous behavior of the LAPI **PUT** operations.

Synchronizing

The **LAPI_Rmw** function is used to synchronize two independent operations such as two processes sharing a common data structure. The operation is performed at the target process and is atomic. The operation takes a variable from the origin and performs one of four selected operations on a variable from the target and replaces the target variable with the results of the operation. The original value of the target variable is returned to the origin. **LAPI_Rmw** provides four different read/modify/write operations:

- **SWAP**
- **COMPARE_AND_SWAP**
- **FETCH_AND_ADD**
- **FETCH_AND_OR**

Completion is signaled at the origin if the counter is specified.

Completion Checking

The following counter functions provide the means for a process to manage the completion state of the LAPI operations.

- **LAPI_Waitcntr** - Wait on a counter to reach a specified value and return when the counter is equal to or greater than that value (blocking)
- **LAPI_Getcntr** - Get the current value of a specified counter (non-blocking)
- **LAPI_Setcntr** - Set the counter to a specified value

These functions also provide an efficient means to order the flow of LAPI operations or the use of certain user managed resources (for example, buffers). For example, a series of **PUTs** to a single target and buffer requires that the contents of the buffer at the target remains in step with the order of execution of the **PUTs** at the origin. Using the *cmpl_cntr* counter in the **LAPI_Put** function in conjunction with the **LAPI_Waitcntr** function provides the necessary ordering.

Ordering

LAPI-Fence and **LAPI_Gfence** operations provide a means to enforce the order of execution of LAPI functions. LAPI functions initiated prior to these fencing operations are guaranteed to complete before LAPI functions initiated after the fencing functions. **LAPI_Fence** is a local operation which is used to guarantee that all LAPI operations initiated by the local process and the same process thread are complete. **LAPI_Gfence** is a collective operation involving all processes in the parallel program. **LAPI_Gfence** provides a barrier operation for the parallel program. Both **LAPI_Fence** and **LAPI_Gfence** operations are a data fence that guarantee that the data movement is complete. These are not an operation fence which would need to include active message completion handlers completing at the target.

Progress

The **LAPI_Probe** function is used in polling mode to transfer control to the communication subsystem in order to make progress on arriving messages.

Address Manipulation

The **LAPI_Address_init** collective operation allows processes to exchange operand addresses of interest. Such function is required if the processes do not have identical address maps. The **LAPI_Address** is used by Fortran programs when an address needs to be stored in an array. In Fortran there is no concept of “address” (as there is in the C language) and this function gives that ability to Fortran.

LAPI Setup

LAPI_Init and **LAPI_Term** operations are used to initialize and terminate the communication structures required to effect LAPI communications for a specific instance of LAPI. **LAPI_Init** returns a unique handle which is used to associate a communication channel with the LAPI instance. This handle is subsequently passed as a parameter to each of the other LAPI functions. The **LAPI_Term** function is used to terminate a specific instance of LAPI.

Error Handling and Messages

The **LAPI_Init** function provides a means for the user of LAPI to register an error handler. This error handler is specified as part of the *lapi_info* structure parameter which is passed to the **LAPI_Init** function. The **LAPI_Msg_String** function provides the means to translate a LAPI call return code value (integer) into a message string.

LAPI Environment

The **LAPI_Qenv** function is used to query the state of the LAPI communications subsystem. The **LAPI_Senv** function allows the programmer to specify the value of some of the LAPI communications subsystem's environment variables. An important value that can be specified is the interrupt state. The interrupt state is set by specifying **INTERRUPT_SET** as **on** (for interrupt mode) or **off** (for polling mode). The default setting for **INTERRUPT_SET** is **on**.

The LAPI Execution Model

The goal of LAPI is to provide a threads-safe environment and support an execution model that allows for maximum execution concurrency within the LAPI library.

Using the setup function (**LAPI_Init**), a user process establishes a LAPI context. Within a LAPI context, the LAPI library is threads-safe, and multiple threads may make LAPI calls within the same context. The different calls may execute concurrently with each other and with the user threads. However, in reality execution concurrency among these calls is limited by the locking required with LAPI to maintain integrity of its internal data structures and the need to share a single underlying communication channel.

As with any multi-threaded application, coherence of user data is the responsibility of the user. Specifically, if two or more LAPI calls from different threads can execute concurrently and if they specify overlapping user buffer areas, then the result is undefined. It is the responsibility of the user to coordinate the required synchronization between threads that operate on overlapping buffers.

The user application thread, as well as the completion handlers, cannot hold mutual exclusion resources before making LAPI calls; if they do, it is possible to run into deadlock situations.

Because user-defined handlers may be called concurrently from multiple threads, it is the user's responsibility to make them threads-safe.

The LAPI does not currently support multiple user communication threads. User communication thread is defined as a user created thread which makes calls to the LAPI library. Multiple user computational threads are supported. From the user application thread a second thread is created by **LAPI_Init** to process notification events and invoke the LAPI handlers as illustrated in Figure 42.

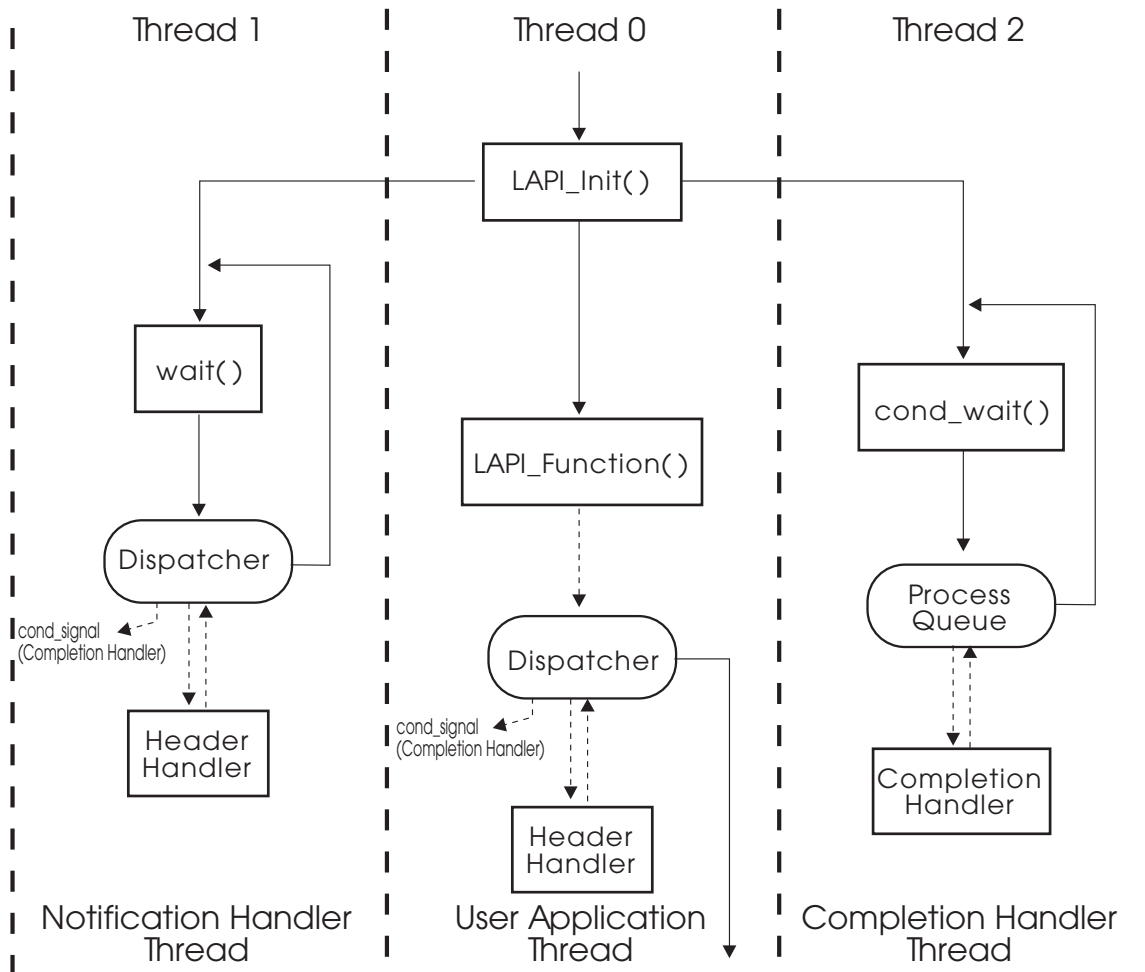


Figure 42. A LAPI Thread Model

The application thread, notification thread and the completion handler thread are shown in Figure 42.

Threads 0 and 1 (the application thread and the notification thread) attempt to invoke the LAPI dispatcher whenever possible; in this way, progress on incoming and outgoing messages can be made while minimizing additional overhead. Most LAPI calls (though not all) made by the application thread also result in the LAPI dispatcher being automatically run. The notification thread waits in the Kernel for the occurrence of a notification event. When an event occurs, the Kernel wakes up

the waiting thread. As shown in Figure 42, after the notification thread returns from waiting in the Kernel, it invokes the LAPI dispatcher.

The LAPI Dispatcher is the central control point that orchestrates the invocation of functions and threads necessary to process outstanding incoming and outgoing LAPI messages.

The LAPI Dispatcher can run from the application's user's thread, from the notification thread or from the completion handler thread. Locking is used to ensure that only one instance of the dispatcher runs at a time to maintain integrity. On incoming messages, the LAPI dispatcher manages the reassembly of data from different packets (which may arrive out-of-order) into the specified buffer, and then invokes the completion handler if necessary.

Thread 2 is created by **LAPI_Init** to execute completion handlers associated with active messages. Completion handlers are written by users and may make LAPI function calls which in turn will invoke the LAPI Dispatcher. The completion handler thread processes work from the completion handler queue. When the queue is empty the thread waits using a **pthread_cond_wait()**. If an active message (**LAPI_Amsend**) includes a completion handler, the dispatcher queues a request on the completion queue after the whole message has arrived and has been reassembled in the specified buffer; the dispatcher then sends a **pthread_cond_signal** to the completion handler thread. If this thread was in a wait state it will begin processing the completion handler queue, otherwise, if it was not waiting, the thread signal is ignored.

LAPI handlers are not guaranteed to execute one at a time. Note that LAPI calls may execute concurrently within the origin or target or both. The same restrictions stated previously about not holding on to mutual exclusion resources when making LAPI calls still applies.

This discussion of a threads-safe environment and maximum execution concurrency within the LAPI library applies to both the polling and interrupt modes. In polling mode any calls to the communication library attempt to make progress on the context specified in the call. Further, the function **LAPI_Probe** is provided to allow applications to explicitly check for and handle incoming messages.

The execution model of the handlers consists of the following events:

Event	Action
Message Arrival	Copies the message from the network into the appropriate data access memory space.
Interrupt / Poll	Causes an interrupt if required, based on the mode.
Dispatcher Start	Invokes the dispatcher.
New Message Packet	Checks the LAPI header and determines (by checking the receive state message reassembly table) if the packet is part of a pending message or if it is a new message. Calls the

header-handler function for the first packet of a new message.

Return from Header-Handler

If the message is contained in more than one packet, the LAPI Dispatcher will log that there is a pending message, save the completion handler address, and save the user's buffer address to be used during the message reassembly of pending message packets.

Pending Message Packet

Copies the message to the appropriate portion of the user buffer specified through the header-handler. If the packet completes the message, the dispatcher queues the completion handler; otherwise the dispatcher returns to check for message arrivals.

Completion Handler

When the completion handler is executed, (after the return from the completion handler) updates the appropriate target counter before continuing.

Allocating Buffers

1. The user allocates as many buffers per origin as wanted.
 - Rule: At origin, if the target buffer allocation is exhausted, wait for the counter of previous requests.
 - Example: Use $\geq P$ buffers (one per origin). At origin with one request pending. Wait on counter before issuing next request.
2. If the header handler blocks, no further progress is made, including messages pending (that is, the communications adapter is stalled).

Caution: The user is allowed to make LAPI calls from within the completion handler. However, the users should exercise caution when writing completion handlers which do long computation and then issue LAPI calls. The long computation may cause completion handler queues to fill up if multiple active messages are sent before these handlers complete. This may cause fetch deadlocks. In such cases, users should fork a separate thread from within the completion handler and have the forked thread make LAPI calls to eliminate the possibility of a fetch deadlock occurring.

Executing LAPI Programs

The LAPI communications application programming interface must be used in conjunction with the IBM Parallel Environment for AIX (PE) product. Specifically the Parallel Operating Environment (POE) component of PE is used to compile and run LAPI parallel programs. POE also provides support for parallel programs to use the Message Passing Interface (MPI) both separately and in combination with the LAPI. The publication *IBM Parallel Environment for AIX: Operation and Use* provides detailed information about using the POE.

Compiling LAPI Programs

As with a serial application, you must compile a parallel C, C++, or Fortran program before you can run it. Instead of using the `cc_r`, `xlc_r`, or `xlf` commands, however, you use the commands `mpcc_r`, `mpCC_r`, or `mpxlf_r`, which support programs using the threaded LAPI library. The `mpcc_r`, `mpCC_r`, and `mpxlf_r` commands not only compile your program, but also link in the POE Partition Manager and PSSP Communication Subsystem (CSS) interfaces. When you later run the program, the CSS libraries will be dynamically linked with the executable program. Subroutines in these libraries enable POE's home node (the node from which the parallel program is invoked) to communicate with the parallel tasks, and tasks with each other.

The following table shows what to enter to compile a LAPI program.

To compile a C program	<code>mpcc_r program.c -o program</code>
To compile a C++ program	<code>mpCC_r program.C -o program</code>
To compile a Fortran program	<code>mpxlf_r program.f -o program</code>

Creating a Static Executable:

Note: IBM suggests you avoid creating statically bound executables. If service is ever applied that affects any of the CSS libraries, you will need to recompile your application to create a new executable that will work with the new libraries. This could lead to a lot of work and may expose you to potential problems, which would be avoided if dynamic libraries are used.

The following is a sample makefile:

```
COMPILER = mpcc_r
CFLAGS   = -g

FCOMPILER = mpxlf_r
FFLAGS    = -g

STATIC = -bnso -bI:/usr/lib/syscalls.exp -bI:/usr/lib/threads.exp \
        -bI:/usr/lpp/ssp/css/libus/fs_ext.exp -lpe_r -lhal_r

all: Get Put Am Rmw Getf Putf Rmwf Amf Get_static

Get: Get.c
    $(COMPILER) $(CFLAGS) -v -o $@ $@.c

Getf: Getf.f
    $(FCOMPILER) $(FFLAGS) -v -o $@ $@.f

Put: Put.c
    $(COMPILER) $(CFLAGS) -v -o $@ $@.c

Putf: Putf.f
    $(FCOMPILER) $(FFLAGS) -v -o $@ $@.f

Am: Am.c
    $(COMPILER) $(CFLAGS) -v -o $@ $@.c

Amf: Amf.f
```

```

$(FCOMPILER) $(FFLAGS) -v -o $@ $@.f

Rmw: Rmw.c
$(COMPILER) $(CFLAGS) -v -o $@ $@.c

Rmwf: Rmwf.f
$(FCOMPILER) $(FFLAGS) -v -o $@ $@.f

Get_static: Get.c
$(COMPILER) $(CFLAGS) $(STATIC) -v -o $@
Get.c

clean:
rm -rf Get Put Am Rmw Getf Putf Rmwf Amf Get_static

```

Running LAPI Programs

Before running your program, you need to set up your execution environment. There are a number of POE environment variables discussed throughout *IBM Parallel Environment for AIX: Operation and Use* and summarized in the “POE Environment Variables and Command-Line Flags” appendix of that book. LAPI programs must set the environment variable **MP_EUILIB** (or the command-line flag **-eulib**) to **us** (**us** is the User Space communication subsystem). The default setting for **MP_EUILIB** is **ip** (**ip** is the IP communication subsystem). The LAPI supports only User Space communication using the SP Switch. The LAPI does not support using the IP communication subsystem.

All other procedures for running a LAPI program using POE are common with the MPI programs and are described in *IBM Parallel Environment for AIX: Operation and Use*.

Specific LAPI Hints

The LAPI library uses the SIGBUS signal handler. If an application using LAPI also uses SIGBUS, the application's SIGBUS handler should be registered before the call is made to the **LAPI_Init** function.

The **LAPI_Senv** function should be used to enable error checking (**ERROR_CHK** should be set on) during application development and disabled (**ERROR_CHK** should be set off) for normal application use.

When the order of execution between any two LAPI functions within one task of a parallel program needs to be guaranteed, using **LAPI_Waitcptr** between the two LAPI functions will usually be more efficient than **LAPI_Fence**. **LAPI_Fence** requires that all LAPI operations initiated on the current thread before the **LAPI_Fence** be completed before any LAPI operation after the fence is allowed to start. **LAPI_Waitcptr** can be used to indicate the completion of a single LAPI function which had been initiated on the current thread before the **LAPI_Waitcptr**.

The scope of **LAPI_Fence** is per thread. For example, a **LAPI_Fence** which is issued from the completion handler thread will only guarantee that no LAPI operations initiated after the fence (on the completion handler thread) will start until all LAPI operations initiated before the fence have completed. In this case there are no guarantees about the order of LAPI operations initiated from the main application thread.

LAPI_Waitcntr can be used to indicate the completion of a single LAPI function which may have been initiated from an alternate thread (completion handler) within the same task. Therefore the possibility exists to use **LAPI_Waitcntr** to wait for the completion of another LAPI function which is initiated after the call to **LAPI_Waitcntr**.

LAPI_Waitcntr can be used to guarantee order of execution of **LAPI_Amsend** operations which are initiated from a single origin task. When **LAPI_Amsend** operations use the *cmpl_cntr* counter, this counter is incremented after the completion counter (or header handler if a completion handler is not specified) has executed at the target task. **LAPI_Fence** and **LAPI_Gfence** do not provide an indication that **LAPI_Amsend** operations have completed execution at the target.

LAPI_Waitcntr is a blocking call. If a user prefers to avoid this blocking operation a program loop comprised of the sequence

```
LAPI_Getcntr  
a check of the value returned from Get  
LAPI_Probe
```

will provide an equivalent logical operation and provide the user with added flexibility.

LAPI_Init must be called before any thread (the main thread or the completion handler thread) can make a LAPI call. In addition to this, **LAPI_Address_init** or **LAPI_Gfence** should be the second LAPI call. These two functions provide a barrier which guarantees that all other LAPI tasks have initialized their LAPI subsystems and are ready to receive requests from remote tasks. Failure to provide these barrier functions may result in dropped switch packets, low performance at start-up and unnecessary switch congestion. The instance of the LAPI subsystem should be quiesced before **LAPI_Term** is called to terminate the LAPI instance. This can be done by calling **LAPI_Gfence** before **LAPI_Term**.

When one task of a parallel job (using POE) determines a condition which requires that all tasks of the parallel job be terminated, the task needs to send a process terminating signal (for example, **SIGTERM** or **SIGQUIT**) to itself. The POE daemon (**PMD**) which spawned that task will:

1. Catch **SIGCHILD**
2. Send a message to the POE "home node"
3. The POE "home node" broadcasts a message to all the **PMDs** to terminate the job
4. Each of the **PMDs** sends **SIGTERM** to their child process and waits for **SIGCHILD**
5. Each task termination is reported back to the "home node"
6. The complete POE job ends

General LAPI Hints

Programs using the LAPI library are compiled with the POE commands **mpcc_r**, **mpCC_r** and **mpxlf_r**. The user's program may (but is not required to) create threads. The POE commands also link a POE version of the standard re-entrant C library, **libc_r.a**, which allows synchronization of the tasks at exit. All the PE and CSS libraries are dynamic shared libraries, enhancing the portability of the user's application between operating system levels and machine types. **mpcc_r** invokes the re-entrant C compiler with the compiler option to include the threaded

(re-entrant) libraries and to generate thread-aware code. **mpcc_r** links in the threaded LAPI library, including the threaded versions of the POE utility libraries.

A threaded program has more than one independent instruction stream, but all threads share the same address space, file, and environment variables.

Core Dumps: If a task produces a core file, it is written to an appropriately named subdirectory of the user's current directory. The partial dump produced by default does not contain the stack and status information for all threads; thus it is of limited usefulness in trying to diagnose hang conditions. It is possible to request AIX to produce a full core file, but such files are generally larger than permitted by AIX user limits. The communication subsystem alone generates more than 64MB of core information. Thus, if possible, use the attach capability of **dbx**, **xldb** or **pdbx** to examine the task while it is still running.

Hangs: Coordinating the threads in a task requires careful locking and signaling. Program deadlocks waiting on locks that haven't been released are common, in addition to the deadlock possibilities offered by improper use of the LAPI calls.

Use with Non-Thread-Safe Libraries: A threaded LAPI program must meet the same criteria as any other threaded program. It must avoid using non-thread safe functions in more than one thread (for example, **strtok**). In addition, it must use only thread-safe libraries if library functions are called on more than one thread. In AIX Version 4.2.1, the **libm.a** and **libbsd.a** libraries are NOT thread-safe and their use in a program should be examined carefully.

Linking with Libraries Built with libc.a: Compiling a threaded LAPI program will cause the **libc_r.a** library to be used to resolve all the calls to the standard C library. If your program links with a library that has been built using the standard C library, it is still usable (assuming that it provides the necessary logical thread safety) under the following conditions:

- The library has been built using a reference to the C library shared object (this is the default, unless the **-bnso** flag was used to build it)
- The run-time library path resolves the file reference **libc.a** to the POE version of **libc_r.a**.

To explain further, the run-time library path for an executable is composed of the **AIX LIBPATH** environment variable concatenated with the library path string contained in the **a.out** file (and put there when the program was linked). All the shared libraries used by the executable must be found in the directories occurring in this string. Normally the **AIX LIBPATH** environment variable is empty, and the **cc** (**xlc**) compiler creates a string in which **/usr/lib** is the first entry. Thus, the executable looks for **/usr/lib/libc.a**. However, the **cc_r** (**xlc_r**) compiler creates a string starting with **/usr/lib/threads:/usr/lib**. Thus, the executable will look for **libc.a** in **/usr/lib/threads** first. And, in fact, there is exactly one member of **/usr/lib/threads: /usr/lib/threads/libc.a**, which is, however, a symbolic link to **/usr/lib/libc_r.a**! Thus, the executable looking for **libc.a** actually loads **libc_r.a**. Calls to symbols in **libc.a** are resolved by the second entry, **/usr/lib/libc_r.a**. Thus, all calls to the standard C library are resolved by the same library, providing a consistent internal state.

mpcc_r does the same sort of thing. **mpcc_r** puts **/usr/lpp/ppe.poe/lib/threads:/usr/lpp/ppe.poe/lib** in the executable's libpath string

and expects `/usr/lpp/ppe.poe/lib/threads/libc.a` to be a symbolic link to `/usr/lpp/ppe.poe/lib/libc_r.a`

The **LIBPATH** environment variable can be set by the user directly. The current version of POE (Version 2.3) sets the default LIBPATH as follows:

LIBPATH=\$MP_EUILIBPATH/\$MP_EUILIB, and expects that the `mpcc_r` command has caused `/usr/lpp/ppe.poe/lib` to be included in the libpath search string.

Use of Segment Registers (-bmaxdata Restrictions): The User Space LAPI library uses two segment registers of the 10 that are unassigned in the user's AIX process space. Thus, the user may use a maximum of 8 segments (**-bmaxdata=0x80000000**) for extended heap for large data arrays. The MPI library uses 3 segment registers. A program calling both the LAPI and MPI interfaces has a maximum of 5 segments available for shared memory or extended heap.

IBM suggests that you don't write threaded message passing programs until you are quite familiar with writing and debugging threaded single-task programs.

AIX requires that threaded programs that include `<pthread.h>` put this include first, before includes for `<stdio.h>` or other system includes. This is because `<pthread.h>` defines some conditional compile variables that modify the code generation of subsequent includes, particularly `<stdio.h>`. Note that `<pthread.h>` is not required unless the file uses thread-related calls or data.

POE catches asynchronous signals that normally terminate a program (**SIGINT**, **SIGQUIT**, **SIGTERM**, **SIGDANGER**) and terminates the entire parallel job. In addition, POE catches synchronous signals (**SIGSEGV**, **SIGBUS**) that are specific to a thread, terminates the entire parallel job, and causes a core file to be written at the point of interrupt on that thread. Since it is possible for several threads to generate synchronous signals simultaneously, the POE signal handler blocks subsequent occurrences.

The stacksize for the main thread is set as a characteristic of the user (in `/etc/security/limits` which is maintained by SMIT). The user can set the stacksize of any additional threads created; the default is 96K. Thread stacks are allocated in the process heap.

The main program should **exit**, not just **pthread_exit**. **pthread_exit** may not terminate all threads. **exit** will result in all threads being terminated.

The AIX XLF FORTRAN compiler does not produce thread-safe code. However, a program having one FORTRAN thread (and possibly additional C threads) can use the threaded LAPI library via the `mpxlf_r` command. This command links in the thread-tolerant FORTRAN library (`libxlf90_t.a`). This requires XLF FORTRAN Version 4.

In AIX Version 4.2.1, only the `libc_r.a` and `libsaa.a` are thread-safe. Applications using other libraries must ensure that they are used in a thread-safe way (usually by serializing entry to the function). This applies to all threaded programs under AIX, and is not specific to the threaded LAPI library.

If the user's application is also using the MPI protocol, the threaded MPI library (`libmpi_r.a`) must be used. Use of the MPI signal library (`libmpi.a`) along with the LAPI library is not supported.

The AIX compilers support a flag **-qarch** that allows the user to target code generation to a particular processor architecture. While this option can provide performance enhancements on specific platforms, it inhibits portability, particularly between the Power and PowerPC machines. The LAPI library is not targeted to a particular architecture, and is the same on PowerPC and Power nodes.

Once the LAPI library is initialized, if the process forks, only the forking thread exists in the child process. The child process doing LAPI communications is not supported.

IBM Parallel Environment for AIX: Hitchhiker's Guide provides additional information which has applicability to running LAPI parallel jobs using POE.

LAPI Programming Examples

Active Message (C)

This C program is an example of the use of the LAPI active message call:

```

/*Am.c
**Example Program illustrating use of the LAPI Active Message Call
*/

#include <lapi.h>

#define A_MAX    2
#define I_MAX    10

typedef struct {
    compl_hdlr_t *compl_hdlr; /* header for active message handler */
    int uinfo;                /* pointer to completion handler */
} usrhdr_t;                  /* uinfo passed to -- */
                             /* the completion handler */

volatile int cont=0;

/*
** Function:      The completion handler for the active call.
**               This is invoked at the target after all the data
**               of the active message send (LAPI_Amsend) call have
**               reached the target.
** Parameters:   hndl -> pointer to the handle for the LAPI context
**               param -> pointer to the user param (specified
**                       by user in the header handler function)
**
*/

void
do_get(lapi_handle_t *hndl, void *param)                                do_get
{
    int loop, rc;
    int *buf;

    buf = (int *) param;
    printf("In Completion Handler: Result of AM call\n");

    /* Print Updated buffer */
    for (loop=0; loop < I_MAX; loop++) {
        printf("val[%d] = %d\n", loop, buf[loop]);
    }

    cont= 1;
}

```

```

/*
** Function:   User's active message header handler. This is invoked
**            at the target when the active message first arrives
**            at the target
** Parameters: hndl   ->   pointer to the handle for the LAPI
**                    context
**            uhdr    ->   pointer to the user header
**            uhdrlen ->   pointer to the length of the user
**                    header
**            msglen  ->   pointer to the length of the message
**            compl_hdlr -> pointer to the completion handler
**                    function pointer. This is to be set
**                    by the user in this function. (CAN
**                    be NULL)
**            saved_info -> pointer to the user_info. This is set
**                    by the user in this function. This
**                    parameter is then passed to the
**                    completion handler when the completion
**                    handler is invoked.
*/

void *
hdr_hdlr(lapi_handle_t *hndl, void *uhdr, uint *uhdrlen, uint *msglen,
         compl_hdlr_t **compl_hdlr, void **saved_info)
{
    void *buf;
    usrhdr_t *vhdr;

    printf("In Header Handler\n");
    vhdr = (usrhdr_t *) uhdr;
    *compl_hdlr = (compl_hdlr_t *) vhdr->compl_hdlr;
    *saved_info = (void *) vhdr->uinfo;
    buf = (void *) vhdr->uinfo;

    return (buf);
}

int
main(int argc, char **argv)
{
    lapi_handle_t t_hndl; /* LAPI context handle - */
                          /* returned */
    lapi_info_t t_info; /* LAPI info structure */
    int task_id, /* My task id */
        num_tasks; /* Number of tasks in my */
                  /* job */
    lapi_cntr_t l_cntr; /* Origin counter */
    lapi_cntr_t t_cntr; /* Target counter */
    lapi_cntr_t c_cntr; /* Completion counter */
    int t_buf[I_MAX]; /* Buffer to manipulate */
    void *global_addr[A_MAX]; /* Array to store */
                              /* t_buf addr from all the */
                              /* tasks. The size of this */
                              /* array needs to each */
                              /* number of tasks */
    void *tgt_addr[A_MAX]; /* Array to store target */
                           /* counter addr from all */
                           /* the tasks. */
    void *hdlr_addr[A_MAX]; /* Array to store header */
                           /* handlers */
    void *compl_hdlr_addr[A_MAX]; /* Address of */
                                   /* completion handler */
    usrhdr_t t_uhdr; /* Store Header Handler */
}

```

```

                                                    /* information */
void          *uhdr, *udata;
int           uhdrln, udatalen;
int           loop, rc, tgt, val, cur_val;
char          err_msg_buf[LAPI_MAX_ERR_STRING];

t_info.err_hdlr = NULL; /* Not registering error handler */
                    /* function */

...main

if ((rc = LAPI_Init(&t_hdl, &t_info)) != LAPI_SUCCESS) {
    LAPI_Msg_string(rc, err_msg_buf);
    printf("Error Message: %s, rc = %d\n", err_msg_buf, rc);
    exit (rc);
}

rc = LAPI_Qenv(t_hdl, TASK_ID, &task_id); /* Get task id */
                    /* within job */
rc = LAPI_Qenv(t_hdl, NUM_TASKS, &num_tasks); /* Get no. of tasks */
                    /* in job */

if (num_tasks != 2) {
    printf("Error Message: Program should run on 2 nodes\n");
    exit(1);
}

/* Turn off parameter checking - default is on */
rc = LAPI_Senv(t_hdl, ERROR_CHK, 0);

/* Initialize counters to be zero at the start */
rc = LAPI_Setcctr(t_hdl, &l_cctr, 0);
rc = LAPI_Setcctr(t_hdl, &t_cctr, 0);
rc = LAPI_Setcctr(t_hdl, &c_cctr, 0);

/*
** Exchange buffer address, tgt_cctr address and hndr_hdlr address
** and completion handler address of every task. Collective calls
*/
rc = LAPI_Address_init(t_hdl, t_buf, global_addr);
rc = LAPI_Address_init(t_hdl, &t_cctr, tgt_addr);
rc = LAPI_Address_init(t_hdl, (void *)&hndr_hdlr, hndlr_addr);
rc = LAPI_Address_init(t_hdl, (void *)&do_get, compl_hdlr_addr);

if (task_id == 0) { /* Task id is 0 , Origin */
    tgt = task_id + 1;
    for (loop=0; loop < I_MAX; loop++) { /* Update buffer */
        t_buf[loop] = task_id - loop;
    }
    rc = LAPI_Gfence(t_hdl); /* Global fence to sync before */
                    /* starting */

    /* Fill in uhdr and udata buffers for AM call */
    t_uhdr.compl_hdlr = (compl_hdlr_t *) compl_hdlr_addr[1];
    t_uhdr.uinfo      = (int)global_addr[tgt];
    uhdr              = (void *)&t_uhdr;
    uhdrln            = sizeof(usrhdr_t);
    udata             = (void *) t_buf;
    udatalen         = I_MAX*sizeof(int);

    rc = LAPI_Amsend(t_hdl, tgt, hndlr_addr[tgt], uhdr, uhdrln,
        (void *) udata, udatalen, tgt_addr[tgt], &l_cctr, &c_cctr);

    /* Wait for local AM completion */
    rc = LAPI_Waitcctr(t_hdl, &l_cctr, 1, &cur_val);

```

```

/* Can now change local buffer */
for (loop=0; loop < I_MAX; loop++) { /* Update buffer */
    t_buf[loop] = loop * task_id;
}

/* Wait for target AM completion */
rc = LAPI_Waitcctr(t_hdl, &c_ctr, 1, &cur_val);
printf("Node %d, done issuing AM to node %d\n", task_id, tgt);

rc = LAPI_Gfence(t_hdl);
rc = LAPI_Get(t_hdl, tgt, I_MAX*sizeof(int), global_addr[tgt],
              (void *)t_buf, tgt_addr[tgt], &l_ctr);
/* Wait for local Get completion */
rc = LAPI_Waitcctr(t_hdl, &l_ctr, 1, NULL);

printf("Node %d, done issuing Get from node %d\n", task_id, tgt);
printf("Result of Get after the Am from node %d: \n", tgt);
for (loop=0; loop < I_MAX; loop++) { /* Update buffer */
    printf("Val[%d] = %d\n", loop, t_buf[loop]);
}
} else { /* Task id is 1 , Target */
    tgt = task_id - 1;
    for (loop=0; loop < I_MAX; loop++) { /* Zero out buffer */
        t_buf[loop] = 0;
    }
    rc = LAPI_Gfence(t_hdl); /* Global fence to sync before */
                          /* starting */

    /* Process AM */
    rc = LAPI_Getcctr(t_hdl, &t_ctr, &val);
    while (val < 1) {
        sleep(1); /* Do some work */
        rc = LAPI_Probe(t_hdl); /* Poll the adapter once */
        rc = LAPI_Getcctr(t_hdl, &t_ctr, &val);
    }
    /* To reset the t_ctr value */
    rc = LAPI_Waitcctr(t_hdl, &t_ctr, 1, &cur_val);
    printf("Node %d, done doing work and processing AM\n", task_id);
    while (!cont) {
        sleep(1); /* Do some work */
    }
    rc = LAPI_Gfence(t_hdl);
    /* Process Get */
    rc = LAPI_Getcctr(t_hdl, &t_ctr, &val);
    while (val < 1) {
        sleep(1); /* Do some work */
        rc = LAPI_Probe(t_hdl); /* Poll the adapter once */
        rc = LAPI_Getcctr(t_hdl, &t_ctr, &val);
    }
    /* To reset the t_ctr value */
    rc = LAPI_Waitcctr(t_hdl, &t_ctr, 1, &cur_val);
    printf("Node %d, done doing work and processing Get\n", task_id);
}
rc = LAPI_Gfence(t_hdl); /* Global fence to sync before */
                          /* terminating job */
rc = LAPI_Term(t_hdl);
}

```

Active Message (Fortran)

This Fortran program is an example of the use of the LAPI active message call:

```
/* Amf.f */
INCLUDE 'lapif.h'

INTEGER VOLATILE CONT
COMMON /DATA/ CONT
INTEGER T_HNDL, T_INFO(10)
INTEGER TASKID, NUMTASKS
INTEGER T_BUF(10)
INTEGER L_CNTR, T_CNTR, C_CNTR
INTEGER GLOBAL_ADDR(2)
INTEGER TGT_ADDR(2), HNDLR_ADDR(2)
INTEGER LOOP, IERROR, TGT, TGT2, VAL, CUR_VAL, LENGTH
INTEGER T_ADDR
INTEGER T_UHDR(2)
INTEGER UHDR, UDATA
INTEGER UHDRLEN, UDATALEN
EXTERNAL DO_GET, HDR_HNDLR

INTEGER ERR_MSG_BUF(40)

c Not registering error handler function
CALL LAPI_ADDRESS(MY_ERR_HNDLR, T_ADDR, IERROR)
T_INFO(7) = T_ADDR                                T_INFO

CALL LAPI_INIT(T_HNDL, T_INFO, IERROR)
IF (IERROR .NE. LAPI_SUCCESS) THEN                IF
    VAL = IERROR
    CALL LAPI_MSG_STRING(VAL, ERR_MSG_BUF, IERROR)
    WRITE(6,*) 'Error Message ', IERROR
    STOP 1
ENDIF

c GET task number and number of tasks in job
CALL LAPI_QENV(T_HNDL, TASK_ID, TASKID, IERROR)
CALL LAPI_QENV(T_HNDL, NUM_TASKS, NUMTASKS, IERROR)

CALL LAPI_QENV(T_HNDL, TIMEOUT, TIME_OUT, IERROR)
CALL LAPI_QENV(T_HNDL, INTERRUPT_SET, INTR_SET, IERROR)

IF (TIME_OUT .gt. 30) THEN                          IF
    VAL = 15
    CALL LAPI_SENV(T_HNDL, TIMEOUT, VAL, IERROR)
ENDIF

IF (INTR_SET .eq. 1) THEN                             IF
c Turn off interrupts
    VAL = 0
    CALL LAPI_SENV(T_HNDL, INTERRUPT_SET, VAL, IERROR)
ENDIF

c Turn off parameter checking - default is on
    VAL=0
    CALL LAPI_SENV(T_HNDL, ERROR_CHK, VAL, IERROR)

c Initialize counters to be zero at the start
CALL LAPI_SETCNTR(T_HNDL, L_CNTR, VAL, IERROR)
CALL LAPI_SETCNTR(T_HNDL, T_CNTR, VAL, IERROR)
CALL LAPI_SETCNTR(T_HNDL, C_CNTR, VAL, IERROR)

WRITE(6,*) "Node ", TASKID, " Running AM fortran test." WRITE
```

```

IF (NUMTASKS .eq. 2) THEN
c      Run only if number of tasks equal 2
c      Exchange buffer address to every task - Collective call
CALL LAPI_ADDRESS(T_BUF, T_ADDR, IERROR)
CALL LAPI_ADDRESS_INIT(T_HNDL,T_ADDR,GLOBAL_ADDR,IERROR)
CALL LAPI_ADDRESS(T_CNTR, T_ADDR, IERROR)
CALL LAPI_ADDRESS_INIT(T_HNDL,T_ADDR,TGT_ADDR,IERROR)
CALL LAPI_ADDRESS(HDR_HNDLR, T_ADDR, IERROR)
CALL LAPI_ADDRESS_INIT(T_HNDL,T_ADDR,HNDLR_ADDR,IERROR)

c      Task id is 0 , Origin
IF (TASKID .eq. 0) THEN
    TGT = TASKID + 1

c      Buffer in Fortran start at 1 and not 0
TGT2 = TGT + 1
LENGTH = 10*4

DO LOOP = 1, 10
    Update buffer
    T_BUF(LOOP) = TASKID - LOOP;
ENDDO

c      Global FENCE to sync before starting
CALL LAPI_GFENCE(T_HNDL, IERROR)

c      Fill in uhdr and udata buffers for AM call
CALL LAPI_ADDRESS(DO_GET, T_UHDR(1), IERROR)
T_UHDR(2) = GLOBAL_ADDR(TGT2)
UHDRLEN = 2 * 4

c      Issue AM from origin to target
CALL LAPI_AMSEND(T_HNDL, TGT, HNDLR_ADDR(TGT2),
1          T_UHDR, UHDRLEN, T_BUF, LENGTH,
2          TGT_ADDR(TGT2), L_CNTR, C_CNTR, IERROR)

c      CALL LAPI_PUT(T_HNDL,TGT,LENGTH,GLOBAL_ADDR(TGT2),
c      1          T_BUF, TGT_ADDR(TGT2),
c      2          L_CNTR, C_CNTR, IERROR)

VAL = 1
CALL LAPI_WAITCNTR(T_HNDL, L_CNTR, VAL,
1          LAPI_ADDR_NULL, IERROR)

c      Local buffer can be reused now
DO LOOP = 1, 10
    T_BUF(LOOP) = TGT2;
ENDDO

VAL = 1
CALL LAPI_WAITCNTR(T_HNDL, C_CNTR, VAL,
1          LAPI_ADDR_NULL, IERROR)

c      CALL LAPI_GFENCE(T_HNDL, IERROR)
c      Issue GET from origin to target
CALL LAPI_GET(T_HNDL,TGT,LENGTH,GLOBAL_ADDR(TGT2),
1          T_BUF, TGT_ADDR(TGT2),
2          L_CNTR, IERROR)

VAL = 1
CALL LAPI_WAITCNTR(T_HNDL, L_CNTR, VAL,
1          LAPI_ADDR_NULL, IERROR)

WRITE(6,*) "Node ",TASKID,
1          "done issuing GET from node ", TGT

```

IF

IF

T_BUF

T_UHDR

T_BUF

...T_BUF

WRITE

```

WRITE(6,*) "Result of GET from node ", TGT                                WRITE

DO LOOP = 1, 10
  WRITE(6,*) "T_BUF(", LOOP, ") = ", T_BUF(LOOP)                        WRITE
ENDDO

c Task id is 1 , Target
ELSEIF (TASKID .eq. 1) THEN                                           ELSEIF
  TGT = TASKID - 1
  DO LOOP = 1, 10
    c Zero out buffer
    T_BUF(LOOP) = 0                                                    T_BUF
  ENDDO

  VAL = 0
  c Global FENCE to sync before starting
  CALL LAPI_GFENCE(T_HNDL, IERROR)
  CALL LAPI_GETCNTR(T_HNDL, T_CNTR, VAL, IERROR)
  DO WHILE (VAL .LT. 1)
    c Can Do some work
    CALL LAPI_PROBE(T_HNDL, IERROR)
    CALL LAPI_GETCNTR(T_HNDL, T_CNTR, VAL, IERROR)
  ENDDO
  WRITE(6,*) "Result of AM from ", TGT, ":"                               WRITE
  DO LOOP = 1, 10
    WRITE(6,*) "T_BUF(", LOOP, ") = ", T_BUF(LOOP)                    WRITE
  ENDDO

  DO WHILE (CONT .NE. 1)
    c Can Do some work
    CALL LAPI_PROBE(T_HNDL, IERROR)
  ENDDO

  CALL LAPI_GFENCE(T_HNDL, IERROR)
  c To clear the T_CNTR VALue
  VAL = 1
  CALL LAPI_WAITCNTR(T_HNDL, T_CNTR, VAL, CUR_VAL, IERROR)
  WRITE(6,*) "Node ", TASKID,                                         WRITE
1 "done doing work and processing AM"
  VAL = 0
  CALL LAPI_GETCNTR(T_HNDL, T_CNTR, VAL, IERROR)
  DO WHILE (VAL .LT. 1)
    c Can Do some work
    CALL LAPI_PROBE(T_HNDL, IERROR)
    CALL LAPI_GETCNTR(T_HNDL, T_CNTR, VAL, IERROR)
  ENDDO

  c To clear the T_CNTR VALue
  VAL = 1
  CALL LAPI_WAITCNTR(T_HNDL, T_CNTR, VAL, CUR_VAL, IERROR)
  WRITE(6,*) "Node ", TASKID,                                         WRITE
1 "done doing work and processing GET"
  ENDIF

ENDIF                                                                    ...WRITE

c Global FENCE to sync before terminating job
CALL LAPI_GFENCE(T_HNDL, IERROR)

CALL LAPI_TERM(T_HNDL, IERROR)
END

SUBROUTINE MY_ERR_HNDLR (HNDL, ERROR_CODE, ERR_TYPE,
1 TASKID, SRC)

```



```

INCLUDE "lapif.h"
INTEGER HNDL, ERROR_CODE, ERR_TYPE, TASKID, SRC
INTEGER BUF(40)

WRITE(6,*) "In my error handler, HNDL=",HNDL,
1      " ERROR_CODE=",ERROR_CODE," ERR_TYPE=",ERR_TYPE,
2      " TASKID=",TASKID," SRC=",SRC
CALL LAPI_MSG_STRING(ERROR_CODE, BUF, IERROR)
WRITE(6,*) "In my error handler, error code = ", ERROR_CODE
IF (ERROR_CODE .ne. LAPI_ERR_TIMEOUT) THEN
c      Cause program to exit
      STOP 2
ENDIF

RETURN
END

INTEGER FUNCTION HDR_HNDLR(HNDL, UHDR, UHDRLEN, MSGLEN,
1      COMPL_HNDLR, SAVED_INFO)
INCLUDE "lapif.h"
INTEGER HNDL
INTEGER UHDR(*)
INTEGER UHDRLEN
INTEGER MSGLEN, COMPL_HNDLR, SAVED_INFO
INTEGER T_ADDR

WRITE(6,*) "In Header Handler"
WRITE(6,*) "In Header Handler: UHDRLEN = ", UHDRLEN
COMPL_HNDLR = UHDR(1)
SAVED_INFO = UHDR(2)
HDR_HNDLR = UHDR(2)

RETURN
END

SUBROUTINE DO_GET (HNDL, PARAM)

INCLUDE "lapif.h"
INTEGER HNDL, PARAM(10)
INTEGER LOOP
INTEGER VOLATILE CONT
COMMON /DATA/ CONT

WRITE(6,*) "In Completion Handler: Result of AM call"
c Print Updated buffer
DO LOOP = 1, 10
      WRITE(6,*) "val[" ,LOOP, "] = ",PARAM(LOOP)
ENDDO
CONT = 1

RETURN
END

```

WRITE

WRITE

...WRITE

Get (C)

This C program is an example of the use of **LAPI_Get**:

```
/* Get.c
** Example Program illustrating the use of LAPI_Get
*/

#include <lapi.h>

#define A_MAX    2
#define I_MAX    10

int
main(int argc, char **argv)                                main
{
    lapi_handle_t t_hdl;                                     /* LAPI context handle - */
                                                            /* returned */
    lapi_info_t   t_info;                                   /* LAPI info structure */
    int task_id,                                          /* My task id */
        num_tasks;                                       /* Number of tasks in my job */
    int          t_buf[I_MAX];                             /* Buffer to manipulate */
    lapi_cntr_t  l_cntr;                                   /* Origin counter */
    lapi_cntr_t  t_cntr;                                   /* Target counter */
    void         *global_addr[A_MAX];                     /* Array to store t_buf addr */
                                                            /* from all the tasks. The */
                                                            /* size of this array needs */
                                                            /* to each number of tasks */
    void         *tgt_addr[A_MAX];                         /* Array to store target */
                                                            /* counter addr from all the */
                                                            /* tasks. */

    int          loop, rc, tgt, val, cur_val;
    char         err_msg_buf[LAPI_MAX_ERR_STRING];

    t_info.err_hdlr = NULL; /* Not registering error handler */
                            /* function */
    if ((rc = LAPI_Init(&t_hdl, &t_info)) != LAPI_SUCCESS) {
        LAPI_Msg_string(rc, err_msg_buf);
        printf("Error Message: %s, rc = %d\n", err_msg_buf, rc);
        exit (rc);
    }
    rc = LAPI_Qenv(t_hdl, TASK_ID, &task_id);             /* Get task id */
                                                            /* within job */
    rc = LAPI_Qenv(t_hdl, NUM_TASKS, &num_tasks);         /* Get no. of tasks */
                                                            /* in job */

    if (num_tasks != 2) {
        printf("Error Message: run with MP_PROCS set to 2\n");
        exit(1);
    }

    /* Turn off parameter checking - default is on */
    rc = LAPI_Senv(t_hdl, ERROR_CHK, 0);

    /* Initialize counters to be zero at the start */
    rc = LAPI_Setcntr(t_hdl, &l_cntr, 0);
    rc = LAPI_Setcntr(t_hdl, &t_cntr, 0);

    /* Exchange buffer address and target_counter address of every task */
    rc = LAPI_Address_init(t_hdl,t_buf,global_addr);      /* Collective */
                                                            /* call */

    rc = LAPI_Address_init(t_hdl,&t_cntr,tgt_addr);        /* Collective */
                                                            /* call */                                ...main
}
```

```

if (task_id == 0) { /* Task id is 0 , Origin */
    tgt = task_id + 1;
    for (loop=0; loop < I_MAX; loop++) { /* Zero out buffer */
        t_buf[loop] = 0;
    }
    rc = LAPI_Gfence(t_hdl); /* Global fence to sync before */
                               /* starting */
    rc = LAPI_Get(t_hdl, tgt, I_MAX*sizeof(int), global_addr[tgt],
                 (void *)t_buf, tgt_addr [tgt], &l_cntr);
    rc = LAPI_Waitcctr(t_hdl, &l_cntr, 1, NULL);

    printf("Node %d, done issuing Get from node %d\n", task_id, tgt) ;
    printf("Result of Get from node %d:\n", tgt);
    for (loop=0; loop < I_MAX; loop++) { /* Update buffer */
        printf("Val[%d] = %d\n", loop, t_buf[loop]);
    }
} else { /* Task id is 1 , Target */
    tgt = task_id - 1;
    for (loop=0; loop < I_MAX; loop++) { /* Update buffer */
        t_buf[loop] = loop + task_id;
    }
    rc = LAPI_Gfence(t_hdl); /* Global fence to sync before */
                               /* starting */
    rc = LAPI_Getcctr(t_hdl, &t_cntr, &val);
    while (val != 1) {
        sleep(1); /* Do some work */
        rc = LAPI_Probe(t_hdl); /* Poll the adapter once */
        rc = LAPI_Getcctr(t_hdl, &t_cntr, &val);
    }
    /* To clear the t_cntr value */
    rc = LAPI_Waitcctr(t_hdl, &t_cntr, val, &cur_val);
    printf("Node %d, done doing work and processing Get\n", task_id) ;
}
rc = LAPI_Gfence(t_hdl); /* Global fence to sync before */
                               /* terminating job */
rc = LAPI_Term(t_hdl);
}

```

Get (Fortran)

This Fortran program is an example of the use of **LAPI_GET**:

```

/* Getf.f */
INCLUDE 'lapif.h'

INTEGER T_HNDL, T_INFO(10)
INTEGER TASKID, NUMTASKS
INTEGER T_BUF(10)
INTEGER L_CNTR, T_CNTR
INTEGER GLOBAL_ADDR(2)
INTEGER TGT_ADDR(2)
INTEGER LOOP, IERROR, TGT, TGT2, VAL, CUR_VAL, LENGTH
INTEGER T_ADDR1, T_ADDR2, T_ADDR3, T_ADDR4
INTEGER ERR_MSG_BUF(40)

c    Not registering error handler function
CALL LAPI_ADDRESS(LAPI_ADDR_NULL, T_ADDR, IERROR)
T_INFO(7) = T_ADDR                                T_INFO

CALL LAPI_INIT(T_HNDL, T_INFO, IERROR)
IF (IERROR .NE. LAPI_SUCCESS) THEN                IF
    VAL = IERROR
    CALL LAPI_MSG_STRING(VAL, ERR_MSG_BUF, IERROR)
    WRITE(6,*) 'Error Message ', IERROR
    STOP 1

```

```

ENDIF

c   GET task number and number of tasks in job
CALL LAPI_QENV(T_HNDL, TASK_ID, TASKID, IERROR)
CALL LAPI_QENV(T_HNDL, NUM_TASKS, NUMTASKS, IERROR)

c   Turn off parameter checking - default is on
VAL=0
CALL LAPI_SENV(T_HNDL, ERROR_CHK, VAL, IERROR)

c   Initialize counters to be zero at the start
CALL LAPI_SETCNTR(T_HNDL, L_CNTR, VAL, IERROR)
CALL LAPI_SETCNTR(T_HNDL, T_CNTR, VAL, IERROR)

IF (NUMTASKS .eq. 2) THEN IF
c   Run only if number of tasks equal 2
c   Exchange buffer address to every task - Collective call
CALL LAPI_ADDRESS(T_BUF, T_ADDR, IERROR)
CALL LAPI_ADDRESS_INIT(T_HNDL,T_ADDR,GLOBAL_ADDR,IERROR)
CALL LAPI_ADDRESS(T_CNTR, T_ADDR, IERROR)
CALL LAPI_ADDRESS_INIT(T_HNDL,T_ADDR,TGT_ADDR,IERROR)

c   Task id is 0 , Origin IF
IF (TASKID .eq. 0) THEN
  TGT = TASKID + 1

c   Buffer in Fortran start at 1 and not 0
  TGT2 = TGT + 1
  LENGTH = 10*4

  DO LOOP = 1, 10
c   Zero out buffer
    T_BUF(LOOP) = 0; T_BUF
  ENDDO ...T_BUF

c   Global FENCE to sync before starting
  CALL LAPI_GFENCE(T_HNDL, IERROR)

c   Issue GET from origin to target
  CALL LAPI_GET(T_HNDL,TGT,LENGTH,GLOBAL_ADDR(TGT2),
1         T_BUF, TGT_ADDR(TGT2),
2         L_CNTR, IERROR)
  VAL = 1
  CALL LAPI_WAITCNTR(T_HNDL, L_CNTR, VAL,
1         LAPI_ADDR_NULL, IERROR)

  WRITE(6,*) "Node ",TASKID, WRITE
1  "done issuing GET from node ", TGT WRITE
  WRITE(6,*) "Result of GET from node ", TGT

  DO LOOP = 1, 10
    WRITE(6,*) "T_BUF(",LOOP,") = ", T_BUF(LOOP) WRITE
  ENDDO

c   Task id is 1 , Target ELSEIF
ELSEIF (TASKID .eq. 1) THEN
  TGT = TASKID - 1
  DO LOOP = 1, 10
c   Update buffer
    T_BUF(LOOP) = LOOP + TASKID T_BUF
    WRITE(6,*) "T_BUF(",LOOP,") = ", T_BUF(LOOP) WRITE
  ENDDO

```

```

        VAL = 0
c      Global FENCE to sync before starting
        CALL LAPI_GFENCE(T_HNDL, IERROR)
        CALL LAPI_GETCNTR(T_HNDL, T_CNTR, VAL, IERROR)
        DO WHILE (VAL .LT. 1)
c          Can Do some work
            CALL LAPI_PROBE(T_HNDL, IERROR)
            CALL LAPI_GETCNTR(T_HNDL, T_CNTR, VAL, IERROR)
        ENDDO

c      To clear the T_CNTR VALue
        VAL = 1
        CALL LAPI_WAITCNTR(T_HNDL, T_CNTR, VAL, CUR_VAL, IERROR)
        WRITE(6,*) "Node ", TASKID,
1         "done doing work and processing Put"
        ENDF

        ENDF

c      Global FENCE to sync before terminating job
        CALL LAPI_GFENCE(T_HNDL, IERROR)

        CALL LAPI_TERM(T_HNDL, IERROR)
        END

```

WRITE

Put (C)

This C program is an example of the use of **LAPI_Put** and **LAPI_Get**:

```

/* Put.c
** Example program illustrating the use of LAPI_Put and LAPI_Get
*/

#include <lapi.h>

#define A_MAX    2
#define I_MAX    10

int
main(int argc, char **argv)
{
    lapi_handle_t t_hdl;           /* LAPI context handle - */
                                   /* returned */
    lapi_info_t   t_info;         /* LAPI info structure */
    int          task_id;         /* My task id */
    int          num_tasks;       /* Number of tasks in my job */
    int          t_buf[I_MAX];    /* Buffer to manipulate */
    lapi_cntr_t  l_cntr;         /* Origin counter */
    lapi_cntr_t  t_cntr;         /* Target counter */
    lapi_cntr_t  c_cntr;         /* Completion counter */
    void         *global_addr[A_MAX]; /* Array to store t_buf */
                                   /* addr from all the tasks */
                                   /* The size of this array */
                                   /* needs to each number of tasks */
    void         *tgt_addr[A_MAX]; /* Array to store target */
                                   /* counter ad from all the */
                                   /* tasks. */

    int          loop, rc, tgt, val, cur_val;
    char         err_msg_buf[LAPI_MAX_ERR_STRING];

    t_info.err_hdlr = NULL; /* Not registering error handler function */
    if ((rc = LAPI_Init(&t_hdl, &t_info)) != LAPI_SUCCESS) {
        LAPI_Msg_string(rc, err_msg_buf);
        printf("Error Message: %s, rc = %d\n", err_msg_buf, rc);
        exit (rc);
    }
}

```

main

```

rc = LAPI_Qenv(t_hdl, TASK_ID, &task_id); /* Get task number */
/* within job */
rc = LAPI_Qenv(t_hdl, NUM_TASKS, &num_tasks); /* Get number of */
/* tasks in job */

if (num_tasks != 2) }
    printf("Error Message: Run with MP_PROCS set to 2\n");
    exit(1);
}

/* Turn off parameter checking - default is on */
rc = LAPI_Senv(t_hdl, ERROR_CHK, 0);

/* Initialize counters to be zero at the start */
rc = LAPI_Setcctr(t_hdl, &l_cctr, 0);
rc = LAPI_Setcctr(t_hdl, &t_cctr, 0);
rc = LAPI_Setcctr(t_hdl, &c_cctr, 0);

/* Exchange buffer address and target counter address of every task */
rc = LAPI_Address_init(t_hdl,t_buf,global_addr); /* Collective call */
/* ...main
rc = LAPI_Address_init(t_hdl,&t_cctr,tgt_addr); /* Collective call */

if (task_id == 0) { /* Task id is 0 , Origin */
    tgt = task_id + 1;
    for (loop=0; loop < I_MAX; loop++) { /* Update buffer */
        t_buf[loop] = task_id - loop;
    }
    rc = LAPI_Gfence(t_hdl); /* Global fence to sync before starting */
    rc = LAPI_Put(t_hdl,tgt,I_MAX*sizeof(int), global_addr[tgt],
        (void *)t_buf,tgt_addr[tgt],&l_cctr,&c_cctr);
    /* Wait for local Put completion */
    rc = LAPI_Waitcctr(t_hdl, &l_cctr, 1, NULL);

    /* Can now change local buffer */
    for (loop=0; loop < I_MAX; loop++) { /* Update buffer */
        t_buf[loop] = loop * task_id;
    }

    /* Wait for target Put completion at task 1 as well */
    rc = LAPI_Waitcctr(t_hdl, &c_cctr, 1, NULL);
    printf("Node %d, done issuing Put to node %d\n", task_id, tgt);

    rc = LAPI_Get(t_hdl,tgt,I_MAX*sizeof(int), global_addr[tgt],
        (void *)t_buf,tgt_addr[tgt],&l_cctr);
    /* Wait for local Get completion */
    rc = LAPI_Waitcctr(t_hdl, &l_cctr, 1, NULL);
    printf("Node %d, done issuing Get from node %d\n", task_id, tgt);
    printf("Result of Get after the Put from node %d:\n", tgt);
    for (loop=0; loop < I_MAX; loop++) { /* Update buffer */
        printf("Val[%d] = %d\n", loop, t_buf[loop]);
    }
} else { /* Task id is 1 , Target */
    tgt = task_id - 1;
    for (loop=0; loop < I_MAX; loop++) { /* Zero out buffer */
        t_buf[loop] = 0;
    }
    rc = LAPI_Gfence(t_hdl); /* Global fence to sync before starting */
    /* Process Put */
    rc = LAPI_Getcctr(t_hdl, &t_cctr, &val);
    while (val < 1) {
        sleep(1); /* Do some work */
        rc = LAPI_Probe(t_hdl); /* Poll the adapter once */
        rc = LAPI_Getcctr(t_hdl, &t_cctr, &val);
    }
}

```

```

}
/* To clear the t_cntr value */
rc = LAPI_Waitcntr(t_hndl, &t_cntr, 1, &cur_val);
printf("Node %d, done doing work and processing Put\n", task_id);
printf("Result of Put from %d:\n", tgt);
for (loop=0; loop < I_MAX; loop++) { /* Update buffer */
    printf("Val[%d] = %d\n", loop, t_buf[loop]);
}
/* Process Get */
rc = LAPI_Getcntr(t_hndl, &t_cntr, &val);
while (val < 1) {
    sleep(1); /* Do some work */
    rc = LAPI_Probe(t_hndl); /* Poll the adapter once */
    rc = LAPI_Getcntr(t_hndl, &t_cntr, &val);
}

...main

/* To clear the t_cntr value */
rc = LAPI_Waitcntr(t_hndl, &t_cntr, 1, &cur_val);
printf("Node %d, done doing work and processing Get\n", task_id);
}
rc = LAPI_Gfence(t_hndl); /* Global fence to sync before terminating job */
rc = LAPI_Term(t_hndl);
}

```

Put (Fortran)

This Fortran program is an example of the use of **LAPI_PUT** and **LAPI_GET**:

```

/* Putf.f */
INCLUDE 'lapif.h'

INTEGER T_HNDL, T_INFO(10)
INTEGER TASKID, NUMTASKS
INTEGER T_BUF(10)
INTEGER L_CNTR, T_CNTR, C_CNTR
INTEGER GLOBAL_ADDR(2)
INTEGER TGT_ADDR(2)
INTEGER LOOP, IERROR, TGT, TGT2, VAL, CUR_VAL, LENGTH
INTEGER T_ADDR
INTEGER ERR_MSG_BUF(40)

c Not registering error handler function
CALL LAPI_ADDRESS(LAPI_ADDR_NULL, T_ADDR, IERROR)
T_INFO(7) = T_ADDR                                T_INFO

CALL LAPI_INIT(T_HNDL, T_INFO, IERROR)
IF (IERROR .NE. LAPI_SUCCESS) THEN                IF
    VAL = IERROR
    CALL LAPI_MSG_STRING(VAL, ERR_MSG_BUF, IERROR)
    WRITE(6,*)'Error Message ',IERROR
    STOP 1
ENDIF

c GET task number and number of tasks in job
CALL LAPI_QENV(T_HNDL, TASK_ID, TASKID, IERROR)
CALL LAPI_QENV(T_HNDL, NUM_TASKS, NUMTASKS, IERROR)

c Turn off parameter checking - default is on
VAL=0
CALL LAPI_SENV(T_HNDL, ERROR_CHK, VAL, IERROR)

c Initialize counters to be zero at the start
CALL LAPI_SETCNTR(T_HNDL, L_CNTR, VAL, IERROR)
CALL LAPI_SETCNTR(T_HNDL, T_CNTR, VAL, IERROR)

```

```

CALL LAPI_SETCNTR(T_HNDL, C_CNTR, VAL, IERROR)

IF (NUMTASKS .eq. 2) THEN IF
c   Run only if number of tasks equal 2
c   Exchange buffer address to every task - Collective call
CALL LAPI_ADDRESS(T_BUF, T_ADDR, IERROR)
CALL LAPI_ADDRESS_INIT(T_HNDL,T_ADDR,GLOBAL_ADDR,IERROR)
CALL LAPI_ADDRESS(T_CNTR, T_ADDR, IERROR)
CALL LAPI_ADDRESS_INIT(T_HNDL,T_ADDR,TGT_ADDR,IERROR)

c   Task id is 0 , Origin
IF (TASKID .eq. 0) THEN IF
    TGT = TASKID + 1

c   Buffer in Fortran start at 1 and not 0
    TGT2 = TGT + 1
    LENGTH = 10*4

    DO LOOP = 1, 10
c      Update buffer
      T_BUF(LOOP) = TASKID - LOOP; T_BUF
    ENDDO

c   Global FENCE to sync before starting ...T_BUF
CALL LAPI_GFENCE(T_HNDL, IERROR)

c   Issue PUT from origin to target
CALL LAPI_PUT(T_HNDL,TGT,LENGTH,GLOBAL_ADDR(TGT2),
1         T_BUF, TGT_ADDR(TGT2),
2         L_CNTR, C_CNTR, IERROR)
    VAL = 1
1   CALL LAPI_WAITCNTR(T_HNDL, L_CNTR, VAL,
        LAPI_ADDR_NULL, IERROR)

c   Local buffer can be reused now
    DO LOOP = 1, 10
      T_BUF(LOOP) = TGT2; T_BUF
    ENDDO

    VAL = 1
1   CALL LAPI_WAITCNTR(T_HNDL, C_CNTR, VAL,
        LAPI_ADDR_NULL, IERROR)

c   Issue GET from origin to target
CALL LAPI_GET(T_HNDL,TGT,LENGTH,GLOBAL_ADDR(TGT2),
1         T_BUF, TGT_ADDR(TGT2),
2         L_CNTR, IERROR)
    VAL = 1
1   CALL LAPI_WAITCNTR(T_HNDL, L_CNTR, VAL,
        LAPI_ADDR_NULL, IERROR)

c   WRITE(6,*) "Node ",TASKID, WRITE
1     "done issuing GET from node ", TGT WRITE
    WRITE(6,*) "Result of GET from node ", TGT

    DO LOOP = 1, 10
      WRITE(6,*) "T_BUF(",LOOP,") = ", T_BUF(LOOP) WRITE
    ENDDO

c   Task id is 1 , Target
ELSEIF (TASKID .eq. 1) THEN ELSEIF
    TGT = TASKID - 1
    DO LOOP = 1, 10
c      Zero out buffer

```



```

        T_BUF(LOOP) = 0
    ENDDO
                                        T_BUF

    VAL = 0
c    Global FENCE to sync before starting
    CALL LAPI_GFENCE(T_HNDL, IERROR)
    CALL LAPI_GETCNTR(T_HNDL, T_CNTR, VAL, IERROR)
    DO WHILE (VAL .LT. 1)
c        Can Do some work
        CALL LAPI_PROBE(T_HNDL, IERROR)
        CALL LAPI_GETCNTR(T_HNDL, T_CNTR, VAL, IERROR)
    ENDDO
    WRITE(6,*) "Result of Put from ",TGT, ":"
                                        WRITE
    DO LOOP = 1, 10
        WRITE(6,*) "T_BUF(",LOOP,") = ", T_BUF(LOOP)
                                        WRITE
    ENDDO

c    To clear the T_CNTR VALue
                                        ...WRITE
    VAL = 1
    CALL LAPI_WAITCNTR(T_HNDL, T_CNTR, VAL, CUR_VAL, IERROR)
    WRITE(6,*) "Node ", TASKID,
                                        WRITE
1        "done doing work and processing PUT"
        VAL = 0
        CALL LAPI_GETCNTR(T_HNDL, T_CNTR, VAL, IERROR)
        DO WHILE (VAL .LT. 1)
c            Can Do some work
            CALL LAPI_PROBE(T_HNDL, IERROR)
            CALL LAPI_GETCNTR(T_HNDL, T_CNTR, VAL, IERROR)
        ENDDO

c    To clear the T_CNTR VALue
                                        WRITE
    VAL = 1
    CALL LAPI_WAITCNTR(T_HNDL, T_CNTR, VAL, CUR_VAL, IERROR)
    WRITE(6,*) "Node ", TASKID,
                                        WRITE
1        "done doing work and processing GET"
    ENDIF

    ENDIF

c    Global FENCE to sync before terminating job
    CALL LAPI_GFENCE(T_HNDL, IERROR)

    CALL LAPI_TERM(T_HNDL, IERROR)
    END

```

Rmw (C)

This C program is an example of **LAPI_Rmw** (read/modify/write function):

```

/* Rmw.c */
** Example Program showing use of read modify write function (LAPI_Rmw).
*/

#include <lapi.h>
#include <signal.h>
#include <unistd.h>

#define A_MAX    2

/*
** User error handler function.
*/

void my_err_hdlr (lapi_handle_t *hdl, int *error_code,
                 lapi_err_t *err_type, int *task_id, int *src)
{

```

```

char buf[LAPI_MAX_ERR_STRING];

printf("In my error handler, hndl=%d, error_code=%d, err_type=%d, "
      "task_id=%d, src=%d\n", *hndl,*error_code,*err_type,*task_id,*src);
LAPI_Msg_string(*error_code,buf);
printf("In my error handler, error code = %d, error reason = %s\n",
      *error_code, buf);
if (*error_code != LAPI_ERR_TIMEOUT)
    kill(getpid(),SIGTERM); /* Cause program to exit */
}

int
main(int argc, char **argv)
{
    lapi_handle_t t_hndl; /* LAPI context handle - returned */
    lapi_info_t t_info; /* LAPI info structure */
    int task_id; /* My task id */
    int num_tasks; /* Number of tasks in my job */
    int t_buf; /* Buffer to manipulate */
    int t2_buf; /* Temporary Buffer */
    lapi_cntr_t l_cntr; /* Origin counter */
    lapi_cntr_t t_cntr; /* Target counter */
    void *global_addr[A_MAX]; /* Array to store t_buf addr from */
    /* all the tasks. The size of */
    /* this array needs to each */
    /* number of tasks */
    void *tgt_addr[A_MAX]; /* Array to store target counter */
    /* addr from all the tasks. */
    int loop, rc, tgt, val, cur_val, prev_tgt_val;
    char err_msg_buf[LAPI_MAX_ERR_STRING];
    int time_out; /* Get current timeout value */
    int intr_set; /* Get current interrupt setting */

    t_info.err_hndl = my_err_hndl; /* register an error handler */
    /* function */

    if ((rc = LAPI_Init(&t_hndl, &t_info)) != LAPI_SUCCESS) {
        LAPI_Msg_string(rc, err_msg_buf);
        printf("Error Message: %s, rc = %d\n", err_msg_buf, rc);
        exit(rc);
    }

    rc = LAPI_Qenv(t_hndl, TASK_ID, &task_id);
    rc = LAPI_Qenv(t_hndl, NUM_TASKS, &num_tasks);

    if (num_tasks != 2) {
        printf("Error Message: Run with MP_PROCS set to 2\n");
        LAPI_Term(t_hndl);
        exit(1);
    }

    rc = LAPI_Qenv(t_hndl, TIMEOUT, &time_out); /* Value in seconds */
    rc = LAPI_Qenv(t_hndl, INTERRUPT_SET, &intr_set);

    if (time_out > 30) {
        rc = LAPI_Senv(t_hndl, TIMEOUT, 15); /* Should be > */
        /* MIN_TIMEOUT */
    }

    if (intr_set == 1) {
        rc = LAPI_Senv(t_hndl, INTERRUPT_SET, 0); /* Turn off */
        /* interrupts */
    }

    /* Turn off parameter checking - default is on */
}

```

```

rc = LAPI_Senv(t_hndl, ERROR_CHK, 0);

/* Initialize counters to be zero at the start */
rc = LAPI_Setcntr(t_hndl, &l_cntr, 0);
rc = LAPI_Setcntr(t_hndl, &t_cntr, 0);

/* Exchange buffer address to every task */
rc = LAPI_Address_init(t_hndl, &t_buf, global_addr);          /* Collective */
                                                    /* call */
rc = LAPI_Address_init(t_hndl, &t_cntr, tgt_addr);          /* Collective */
                                                    /* call */

if (task_id == 0) { /* Task id is 0 , Origin */
    tgt = task_id + 1;
    t_buf = 1; /* Initial value to add at target */

    rc = LAPI_Gfence(t_hndl); /* Global fence to sync before */
                                /* starting */
    rc = LAPI_Rmw(t_hndl, FETCH_AND_ADD, tgt, global_addr[tgt],
                  &t_buf, &prev_tgt_val, &l_cntr);
    /* Wait for local Rmw completion */
    rc = LAPI_Waitcntr(t_hndl, &l_cntr, 1, NULL);

    printf("Node %d, done issuing Rmw to node %d\n", task_id, tgt);

    rc = LAPI_Get(t_hndl,tgt,sizeof(int),global_addr [tgt],
                  (void *)&t2_buf,tgt_addr [tgt],&l_cntr);
    /* Wait for local Get completion */
    rc = LAPI_Waitcntr(t_hndl, &l_cntr, 1, NULL);
    printf("Node %d, done issuing Get from node %d\n", task_id, tgt);
    printf("Result of Get after the Rmw from node %d:\n", tgt);
    printf("Correct value should be %d = %d\n",
           t_buf + prev_tgt_val, t2_buf);
} else { /* Task id is 1 , Target */
    tgt = task_id - 1;
                                                    ...main

    t_buf = 5; /* Set initial buffer value */
    rc = LAPI_Gfence(t_hndl); /* Global fence to sync before */
                                /* starting */

    /* Process Get */
    rc=LAPI_Getcntr(t_hndl, &t_cntr, &val);
    while (val < 1) {
        sleep(1); /* Do some work */
        rc = LAPI_Probe(t_hndl); /* Poll the adapter once */
        rc = LAPI_Getcntr(t_hndl, &t_cntr, &val);
    }
    /* To clear the t_cntr value */
    rc = LAPI_Waitcntr(t_hndl, &t_cntr, 1, &cur_val);
    printf("Node %d, done doing work and processing Get\n", task_id);
}
rc = LAPI_Gfence(t_hndl); /* Global fence to sync before */
                                /* terminating job */
rc = LAPI_Term(t_hndl);
}

```

Rmw (Fortran)

This Fortran program is an example of **LAPI_RMW** (read/write/modify function):

```

/* Rmwf.f */
INCLUDE 'lapif.h'

INTEGER T_HNDL, T_INFO(10)
INTEGER TASKID, NUMTASKS
INTEGER T_BUF, T2_BUF, PREV_TGT_VAL

```

```

INTEGER L_CNTR, T_CNTR
INTEGER GLOBAL_ADDR(2)
INTEGER TGT_ADDR(2)
INTEGER LOOP, IERROR, TGT, TGT2, VAL, CUR_VAL, LENGTH
INTEGER T_ADDR, TIME_OUT, INTR_SET
INTEGER ERR_MSG_BUF(40)

c Not registering error handler function
CALL LAPI_ADDRESS(MY_ERR_HNDLR, T_ADDR, IERROR)
T_INFO(7) = T_ADDR                                T_INFO

CALL LAPI_INIT(T_HNDL, T_INFO, IERROR)
IF (IERROR .NE. LAPI_SUCCESS) THEN                IF
    VAL = IERROR
    CALL LAPI_MSG_STRING(VAL, ERR_MSG_BUF, IERROR)
    WRITE(6,*)'Error Message ',IERROR
    STOP 1
ENDIF

c GET task number and number of tasks in job
CALL LAPI_QENV(T_HNDL, TASK_ID, TASKID, IERROR)
CALL LAPI_QENV(T_HNDL, NUM_TASKS, NUMTASKS, IERROR)

CALL LAPI_QENV(T_HNDL, TIMEOUT, TIME_OUT, IERROR)
CALL LAPI_QENV(T_HNDL, INTERRUPT_SET, INTR_SET, IERROR)

IF (TIME_OUT .gt. 30) THEN                          IF
    VAL = 15
    CALL LAPI_SENV(T_HNDL, TIMEOUT, VAL, IERROR)
ENDIF

IF (INTR_SET .eq. 1) THEN                            IF
c Turn off interrupts
    VAL = 0
    CALL LAPI_SENV(T_HNDL, INTERRUPT_SET, VAL, IERROR)
ENDIF

c Turn off parameter checking - default is on
VAL=0
CALL LAPI_SENV(T_HNDL, ERROR_CHK, VAL, IERROR)

c Initialize counters to be zero at the start
CALL LAPI_SETCNTR(T_HNDL, L_CNTR, VAL, IERROR)
CALL LAPI_SETCNTR(T_HNDL, T_CNTR, VAL, IERROR)
CALL LAPI_SETCNTR(T_HNDL, C_CNTR, VAL, IERROR)

IF (NUMTASKS .eq. 2) THEN                            IF
c Run only if number of tasks equal 2
c Exchange buffer address to every task - Collective call
CALL LAPI_ADDRESS(T_BUF, T_ADDR, IERROR)
CALL LAPI_ADDRESS_INIT(T_HNDL,T_ADDR,GLOBAL_ADDR,IERROR)
CALL LAPI_ADDRESS(T_CNTR, T_ADDR, IERROR)
CALL LAPI_ADDRESS_INIT(T_HNDL,T_ADDR,TGT_ADDR,IERROR)
...IF

c Task id is 0 , Origin
IF (TASKID .eq. 0) THEN                              IF
    TGT = TASKID + 1

c Buffer in Fortran start at 1 and not 0
TGT2 = TGT + 1

c Get INTEGER size buffer
LENGTH = 4

c Initial value to add at target

```

```

T_BUF = 1

c      Global FENCE to sync before starting
      CALL LAPI_GFENCE(T_HNDL, IERROR)

c      Issue RMW from origin to target
      CALL LAPI_RMW(T_HNDL, FETCH_AND_ADD, TGT,
1         GLOBAL_ADDR(TGT2), T_BUF,
2         PREV_TGT_VAL, L_CNTR, IERROR)
      VAL = 1
      CALL LAPI_WAITCNTR(T_HNDL, L_CNTR, VAL,
1         CUR_VAL, IERROR)
      WRITE(6,*) "Node ", TASKID, WRITE
1         "done issuing RMW from node ", TGT

c      Issue GET from origin to target
      CALL LAPI_GET(T_HNDL, TGT, LENGTH, GLOBAL_ADDR(TGT2),
1         T2_BUF, TGT_ADDR(TGT2),
2         L_CNTR, IERROR)
      VAL = 1
      CALL LAPI_WAITCNTR(T_HNDL, L_CNTR, VAL,
1         LAPI_ADDR_NULL, IERROR)

      WRITE(6,*) "Node ", TASKID, WRITE
1         "done issuing GET from node ", TGT
      WRITE(6,*) "Result of GET from node ", TGT WRITE

      VAL = T_BUF + PREV_TGT_VAL
      WRITE(6,*) "Correct value should be ", VAL, WRITE
1         " = ", T2_BUF

c      Task id is 1 , Target
      ELSEIF (TASKID .eq. 1) THEN ELSEIF
      TGT = TASKID - 1
c      Set initial buffer value
      T_BUF = 5

      VAL = 0
c      Global FENCE to sync before starting
      CALL LAPI_GFENCE(T_HNDL, IERROR)
      CALL LAPI_GETCNTR(T_HNDL, T_CNTR, VAL, IERROR)
      DO WHILE (VAL .LT. 1)
c          Can Do some work
          CALL LAPI_PROBE(T_HNDL, IERROR)
          CALL LAPI_GETCNTR(T_HNDL, T_CNTR, VAL, IERROR)
      ENDDO

...ELSEIF

c      To clear the T_CNTR VALue
      VAL = 1
      CALL LAPI_WAITCNTR(T_HNDL, T_CNTR, VAL, CUR_VAL, IERROR)
      WRITE(6,*) "Node ", TASKID, WRITE
1         "done doing work and processing GET"
      ENDIF

      ENDIF

c      Global FENCE to sync before terminating job
      CALL LAPI_GFENCE(T_HNDL, IERROR)

      CALL LAPI_TERM(T_HNDL, IERROR)
      END

```

```

SUBROUTINE MY_ERR_HNDLR (HNDL, ERROR_CODE, ERR_TYPE,
1      TASKID, SRC)

INCLUDE "lapif.h"
INTEGER BUF(40)

WRITE(6,*) "In my error handler, HNDL=",HNDL,
1      " ERROR_CODE=",ERROR_CODE," ERR_TYPE=",ERR_TYPE,
2      " TASKID=",TASKID," SRC=",SRC
CALL LAPI_MSG_STRING(ERROR_CODE, BUF, IERROR)
WRITE(6,*) "In my error handler, error code = ", ERROR_CODE
IF (ERROR_CODE .ne. LAPI_ERR_TIMEOUT) THEN
c      Cause program to exit
      STOP 2
ENDIF

RETURN
END

```

WRITE

WRITE
IF

Appendix A. Managing Root Volume Groups

This is a discussion of the SP system implementation of root volume group mirroring and of using alternate root volume groups. It gives a brief description of common terminology and mirroring support in AIX. It does not provide a complete description of AIX mirroring or volume group support. For that information, reference the appropriate AIX publications.

You can mirror root volume groups so that you have a backup or two of the same version of AIX always ready and waiting to take over in the event of a disk failure. You can also set up alternate root volume groups. That means you can configure a different root volume group, one that contains another version of AIX on another disk or set of disks. Then on different occasions you can boot a node from one root volume group or another. For instance you might want one set of disks for AIX 4.3.2 and another for AIX 4.2.1 so that you can switch from operating with one to the other, perhaps to test a new installation before you use it for production work.

Terminology

Physical Volume	A physical volume is an individual hardware disk drive; it might be internal, SCSI-attached or an external SSA drive.
Volume Group	A volume group is a collection of physical volumes. A physical volume can be in one and only one volume group.
Root Volume Group	A root volume group is a special volume group that contains the AIX base operating system.
Logical Volume	A logical volume is disk space that can be used for file systems, paging, or dump. A logical volume can only be in one volume group, but can span physical volumes.
Logical Partition	A logical partition is a convenient measure of space and is a block of space in a logical volume. Logical partitions can be 1 meg in size up to 128 meg in size, in powers of two.
Physical Partition	A physical partition is, like a logical partition, a unit of measure. Logical partitions and physical partitions are in a one to one ratio in an unmirrored environment. If mirroring is being used, there will be 2 or 3 physical partitions for each logical partition, depending on the number of copies specified by the user.
Mirroring	The ability to make a copy of a logical volume or volume group to provide redundancy. AIX allows one (the original), two (the original plus a mirror image), or three (the original plus two mirror images) copies of a logical volume or volume group. The copies are made below the application layer, transparent to the typical user.
Quorum	A quorum is a method used to determine whether or not there are enough physical volumes remaining in a volume group for that volume group to remain online (varied on). Quorum can be turned on or off by the user. If quorum is on, at least 51 percent of the physical volume <i>votes</i> must be

present or the volume group will be varied offline to maintain data integrity. If quorum is off, only one vote of all the physical volumes is needed to keep the volume group online.

Note: In general, quorum should be turned off in a mirrored environment and turned on in an unmirrored environment. Quorum is turned on by default during node installation. You can set a mirroring option to turn quorum off when mirroring is initiated and you can change the option to have it turned on when you suspend mirroring.

Mirroring Support in AIX

Redundancy is an important element of modern computer system availability. AIX provides redundant copies of the operating system through disk mirroring. Disk mirroring provides for one or two identical copies of AIX, in addition to the original. A mirrored root volume group means that there are multiple copies of the operating system image available to a workstation or node. Mirrored system images are distributed so that a node can remain in operation even after one of the mirrored units fails. If a physical volume which contains the original copy of AIX should fail, a mirrored copy takes over. This take-over is transparent to the user and to any user applications that happen to be running. Through mirroring of the operating system environment, AIX eliminates the possibility that a single physical volume can be a single point of failure for the SP system. PSSP 3.1 provides full support for mirroring on all PSSP 3.1 SP nodes.

Root Volume Group Support in PSSP

PSSP 3.1 supports AIX mirroring and alternate root volume groups on SP nodes by providing commands to:

- Maintain a database of information about the root volume group
- Initiate or suspend mirroring on the nodes
- Maintain the bootlist on the nodes
- Define alternate root volume groups

PSSP 3.1 uses the **Volume_Group** class of the SDR to store mirroring information in volume group objects. Each volume group object contains information that describes a root volume group on a node. When you install PSSP on the control workstation and initialize the SDR, a single volume group object is created with the name of **rootvg** for each node. You can use the **spmkgobj** command to create additional volume group objects to represent alternate root volume groups for a node. For more information on the **Volume_Group** class, refer to “Class = Volume_Group” on page 513.

An attribute of the **Node** object, the **selected_vg** attribute, designates one of the volume group objects as the currently selected volume group. When the SDR is initialized, all the **selected_vg** attributes are set to **rootvg**, the default root volume group object.

PSSP 3.1 provides the following commands for managing root volume groups:

- Commands to manipulate root volume group information:

spmkgobj	Create a new Volume_Group object.
spchvgobj	Change Volume_Group information.
sprmvobj	Remove a Volume_Group object that is not the current volume group.
spbootins	Set boot/install configuration data in the SDR.

Note: Many attributes that are now associated with the Volume_Group object have been moved from the **spbootins** command to the **spchvgobj** command.

- Commands to initiate or suspend mirroring or to switch between alternate root volume groups::

spmirrorvg	Initiate mirroring.
spunmirrorvg	Suspend mirroring.
spbootlist	Set the bootlist based on specifications in the Node and Volume_Group objects.

Each of these commands accepts a list of nodes as input. See the book *PSSP: Command and Technical Reference* for details of how to use these commands.

While these commands comprise the base support for mirroring and alternate root volume groups in PSSP, you can accomplish some tasks using SP Perspectives, or SMIT. They are interfaces between you and the commands, each being more or less appropriate for you depending on your level of experience. Example tasks in this appendix are shown using the base line commands.

Using Mirrored Root Volume Groups

You can set mirroring options and initiate mirroring on nodes that have already been installed, or you can set mirroring options and have mirroring initiated during installation of a node. You can also suspend mirroring. The mirroring options that you can set using the **spchvgobj** command for a given root volume group are: how many copies you want (2 or 3), which disks to use, and whether quorum is to be on or off.

In general, quorum should be turned off in a mirrored environment and turned on in an unmirrored environment. Quorum is turned on by default during node installation unless you specify **-q false** with the **spchvgobj** command before installation when you first set the mirroring options. After installation, the **-q** flag has no affect except when you initiate or suspend mirroring. To turn quorum off on a node that is already installed, specify **-q false** with the **spchvgobj** command when you set the mirroring options before you initiate mirroring. If you want to turn quorum on again, specify **-q true** with the **spchvgobj** command before you suspend mirroring.

To initiate mirroring of a root volume group on a node that is already installed, use the **spmirrorvg** command after you set mirroring options. To suspend mirroring, use the **spunmirrorvg** command. These commands also issue the **bosboot** and **bootlist** commands on the nodes to create the necessary boot records and to manage the list of bootable devices. If the state of quorum changes on a node as a result of initiating or suspending mirroring, a warning message appears prompting you to reboot the node for the new state of quorum to take effect. See the book

| *PSSP: Command and Technical Reference* for details of how to use these
| commands.

| Examples of some typical tasks follow.

| **Installing a Node without Mirroring**

| This sample task installs a node without mirroring and sets the **lppsource_name**
| attribute for the node.

| To install node 3 without mirroring and set `lppsource_name` to `aix432`, do the
| following:

- | 1. Change the volume group **rootvg** to use the lpp source name **aix432** on node
| 3.

```
| spchvgobj -v aix432 -r rootvg -l 3
```

- | 2. Change the node 3 bootp response to *install*.

```
| spbootins -r install -l 3 -s yes
```

- | 3. Network boot the node.

```
| nodecond 1 3
```

| **Installing a Node with Mirroring**

| This is a sample task to install a node with mirroring established from the start.

| To install node 5 with two disks for mirroring with the original copy on `hdisk0` and a
| mirror image on `hdisk1`, do the following:

- | 1. Change the **rootvg** volume group information for node 5 to indicate that you
| want two copies using physical disks `hdisk0` and `hdisk1`, and you want quorum
| turned off (quorum is turned on by default during installation).

```
| spchvgobj -h hdisk0,hdisk1 -c 2 -r rootvg -q false -l 5
```

- | 2. Change the node 5 bootp response to *install*.

```
| spbootins -r install -l 5 -s yes
```

- | 3. Network boot the node.

```
| nodecond 1 5
```

| **Initiating Mirroring on a Node Already Installed without It**

| This sample task extends the root volume group on a node and starts mirroring.
| The node has already been installed but without mirroring.

| To create a mirror of `hdisk0` on `hdisk1` for node 3, do the following:

- | 1. Change the **rootvg** volume group on node 3 to have two copies using physical
| disks `hdisk0` and `hdisk1`, and indicate that you want quorum turned off.

```
| spchvgobj -h hdisk0,hdisk1 -c 2 -r rootvg -q false -l 3
```

- | 2. Start mirroring on node 3.

```
| spmirrorvg -l 3
```

- | 3. Reboot node 3 if the state of quorum changes, as indicated by the warning
| message.

```
cshutdown -r -N 3
```

Restrictions to Mirroring

The following is a list of mirroring restrictions:

- The nodes to be mirrored and the control workstation must be running AIX 4.3.2 or later and PSSP 3.1 or later.
- Mirroring is not performed for dump logical devices.
- AIX supports only double or triple mirroring (the original and one or two more copies of the original).
- AIX does not support mirroring of striped logical volumes.
- Mirroring support does not extend to the control workstation from these commands.
- PSSP supports turning quorum on or off only when mirroring is initiated or suspended.

Using Alternate Root Volume Groups

PSSP 3.1 provides support for alternate root volume groups. This means that you can install several different versions of AIX on different disks that are connected to the same node. That node can then easily be switched from operating with one version to another by rebooting from the disk that has the root volume group of your choice.

Alternate root volume groups can be useful, such as for switching between versions of AIX that have different levels of security, or for quickly backing out from testing new software levels to running production work.

To create alternate root volume groups use the **spmkgobj** command and to remove them use the **sprmvobj** command. Use the **spbootins** command to specify which root volume group is to be used in the next install. Use the **spbootlist** command to set the node's bootlist before rebooting. See the book *PSSP: Command and Technical Reference* for details of how to use these commands.

Examples of some typical tasks follow.

Installing a New Root Volume Group

To create a new volume group called **newvg** on hdisk1 with an lpp source name of aix432 on nodes 3 and 5, do the following:

1. Create a new volume group:

```
spmkgobj -r newvg -h hdisk1 -v aix432 -l 3,5
```

2. Change the selected volume group to **newvg**, set the bootp response to *install*, and run the `setup_server` command:

```
spbootins -c newvg -r install -l 3,5
```

Note: The `spbootins` command runs the `setup_server` command by default.

3. Network boot the nodes:

```
nodecond 1 3 &  
nodecond 1 5
```

The nodes are installed with the information specified in the **newvg** volume group and will reboot from the specified disks.

Note: The bootp response is automatically set to *disk*.

Rebooting a Node from an Alternate Root Volume Group

To switch from operating with the **newvg** volume group on hdisk1 to operating with the **oldvg** volume group on hdisk0, do the following:

1. Change the current volume group as recorded in the SDR:

```
spbootins -c oldvg -l 3,5
```

2. Set the bootlists:

```
spbootlist -l 3,5
```

3. Reboot the nodes:

```
cshutdwn -r -N 3 5
```

Removing an Alternate Root Volume Group

A volume group that is not the current root volume group can be removed. To remove the **newvg** volume group now that it is not the current volume group, issue the following command:

```
sprmvobj -r newvg -l 3,5
```

Appendix B. SP Daemons

The following daemons are specific to the SP servers and processor nodes:

fault_service_Worm_RTG_SP Fault-handling daemon for the SP Switch

haemd Event Manager daemon; matches information about the state of system resources with information about resource conditions that are of interest to client programs to create events; runs on all nodes of a system partition; one instance per system partition runs on the control workstation.

hagsd Group Services daemon; provides services for coordinating and monitoring changes to the state of a client program that is running on a set of nodes in an SP system; runs on all nodes of a system partition; one instance per system partition runs on the control workstation.

hagsglsmd A Group Services subsystem daemon that provides global synchronization services; runs on all nodes of a system partition; one instance per system partition runs on the control workstation.

hardmon Interfaces to the serial port on each SP frame; does tasks such as polling hardware for changes in hardware status, sending state changes to interested parties, sending commands to change power state and key lock position, and providing an interface to the S1 serial port of each processing node

hatsd Topology Services daemon; provides information to other PSSP subsystems about the state of the nodes and adapters on the SP system; runs on all nodes of a system partition; one instance per system partition runs on the control workstation.

hrd Host responds daemon; runs on the control workstation (See Appendix E, "The System Data Repository" on page 477 for details.)

kadmind Kerberos V4 database administration server.

kerberos Kerberos authentication server (doesn't run on nodes)

kpropd Kerberos backup communication daemon (doesn't run on nodes)

pmand The Event Management client daemon that dispatches actions based on subscribed-to events

pmanrmd Provides resource variables for problem management

s70d The hardware monitoring daemon used by Hardmon to interface with the RS/6000 Enterprise Server Model S70 and S70 Advanced hardware. The s70d daemon emulates an SP frame and node supervisor. It performs such tasks as: polling the hardware for changes in status, sending state changes back to Hardmon for subsequent dispersal to interested parties, accepting commands from Hardmon to change the power state, and providing an interface to the server's serial port. This daemon is stated by Hardmon. There is no command line interface to this daemon.

sdrd System Data Repository server

sp_configd	Creates and sends SNMP traps when selected types of errors are recorded
splogd	Logs SP hardware state changes, reports SP hardware errors, and provides a state change alert
spmgrd	An SNMP Manager daemon that runs on the control workstation and interacts with SNMP Agents for extension nodes.
supfilesrv	Provides services for file collections
switchtbl	Job Switch Resource Table Services daemon
sysctld	Sysctl daemon; runs on the control workstation, all nodes, and other RS/6000s
xntpd	Network time protocol daemon, keeps clocks synchronized across the network

These AIX daemons might also run on the SP servers and processor nodes:

automount	Automount daemon; automatically mounts a directory on request based on map entries
cron	Runs shell commands at specified intervals
errdemon	Standard AIX error daemon for recording error log information
inetd	Provides Internet server management for a network
named	Provides server functions for the Domain Name Protocol
nfsd, biod, statd, lockd	Standard NFS daemons
portmap	Provides RPC services
qdaemon	Schedules print jobs from the enq command
rlogind	Provides rlogin services
rpe.mountd	Responds to client NFS mount requests
rpestatd, rpe.lockd	Provide System V style of file and record locking with NFS
sendmail	Routes mail for local or network delivery
snmpd	Provides the Agent functions of the simple network management protocol
srcmstr	System resource controller master daemon
syslogd	Logs system messages
telnetd	Provides telnet services
uprintfd	Constructs and writes kernel messages
writesrv	Allows the write command to send and receive messages
xmservd	Performance Toolbox
ypserv	Runs on NIS servers and slaves to provide NIS services
ypbind	Runs on all NIS clients to communicate with respective server

Appendix C. Sample Files

afscclient.cust

This is a sample of the **afscclient.cust** file.

```
#####
#
# Module: <afscclient.cust>
#
#-----
#
# Description: This script is a sample which is designed to be used
# as a guide for users who may want to install AFS on SP nodes.
# This script is based on a premise that the Control Workstation (CW)
# has already been installed and activated as an AFS client.
#
# This sample script is to be executed by "root" user on the SP node.
# This may be used with the /tftpboot/script.cust with modification
# based on the customer requirements and configuration.
#
#####
PATH=/bin:/usr/bin:/etc:/usr/sbin:/usr/bin/X11:/var/sysman:/var/sysman/etc:
/etc/amd:/usr/lpp/ssp/bin:/usr/lpp/ssp/install/bin:/usr/afsws/etc:/usr/vice/etc:

SERVER='cat /etc/ssp/server_hostname | cut -d" " -f3'

#####
# Create the required AFS directory on /usr filesystem. If the /usr is to
# be used as client/server make sure the directories are installed on the
# SP node server first, and then made available to the SP node clients.
# Make sure you have adequate space (about 5000K) is required to load the
# AFS executables and files on the /usr filesystem.
# We will create the directories on /usr, and then copy the required files
# from the CW to the /usr/vice/etc directory on the SP node.
# For our sample configuration we will reference the CW as "$SERVER"
/usr/sbin/chfs -a size='+10000' /usr

/bin/mkdir -p /usr/vice/etc/C
/bin/mkdir /usr/vice/etc/dkload
# Activate the ticket granting ticket so the remote copy will work successfully
#to bring AFS files from the CW to the SP node.
/usr/lpp/ssp/rcmd/bin/rcmdtgt

# Copy the files from CW /usr/vice/etc directory to the SP node /usr/vice/etc
# directory on the SP node.

/usr/lpp/ssp/rcmd/bin/rcp $SERVER:/usr/vice/etc/CellServDB /usr/vice/etc/CellServDB
/usr/lpp/ssp/rcmd/bin/rcp $SERVER:/usr/vice/etc/ThisCell /usr/vice/etc/ThisCell
/usr/lpp/ssp/rcmd/bin/rcp $SERVER:/usr/vice/etc/afsd /usr/vice/etc/afsd
/usr/lpp/ssp/rcmd/bin/rcp $SERVER:/usr/vice/etc/cacheinfo /usr/vice/etc/cacheinfo
/usr/lpp/ssp/rcmd/bin/rcp $SERVER:/usr/vice/etc/C/* /usr/vice/etc/C/
/usr/lpp/ssp/rcmd/bin/rcp $SERVER:/usr/vice/etc/dkload/* /usr/vice/etc/dkload/
/usr/lpp/ssp/rcmd/bin/rcp $SERVER:/usr/vice/etc/rc.afs /usr/vice/etc/rc.afs

#####
# This next step will setup the cache that will be used on the SP nodes
# These values can be modified, based on the amount of users that will be
# using afs services. We will use the same values as placed on the CW for our
# configuration being used in the sample script. The values to be modified are
```

```

# the filesystem size for /usr/vice/cache, and /usr/vice/etc/cacheinfo size.
# This configuration will setup for a medium afs configuration of 4-8 users.
# We will use a cacheinfo of 80000K, and filesystem of 90000K.
# We will create the /usr/vice/cache directory and then create a JFS that will
# be mounted

/bin/mkdir /usr/vice/cache

/usr/sbin/crfs -v jfs -g rootvg -a size=180000 -m '/usr/vice/cache' -A 'yes' -p 'rw'
/bin/sleep 5
/usr/sbin/mount /usr/vice/cache

#####
# This next step will setup the the afs as one of the supported filesystems
# in the /etc/vfs file. You can either "edit" and place the information
# afs 4 none none
# in the file, or append the information using the "echo" command.

/bin/echo "afs 4 none none" >> /etc/vfs

#####
# This next step will setup the the afs directory, and then activate the afs
# daemon (afsd) that connect to the AFS server. This will be executed by
# starting the /etc/rc.afs script.
#
# The /etc/rc.afs will setup and execute the AFS libraries to be used as
# kernel extension in the SP nodes. You may want to use different execution
# variables based on you configuration and its use with nfs services
# We expect the files to be available in the /usr/vice/etc/dkload directory
#
# /usr/vice/etc/dkload/cfgexport -a export.ext
# /usr/vice/etc/dkload/cfgafs -a afs.ext
#
#####

/bin/mkdir /afs
/usr/vice/etc/rc.afs

/bin/sleep 60
#####
# This next step will setup a soft link of the /afs/<cellname>/@sys/usr/afsws
# directory to /usr/afsws on the SP node. The exact pathnames will need
# to reflect the afs configuration cell name, and the system directory.
# This will be based on where the afsws executables are located.
#
# /bin/ln -s /afs/<cellname>/@sys/usr/afsws /usr/afsws
# My cell execution will be the following:
/bin/ln -s /afs/ppd.pok.ibm.com/rs_aix41/usr/afsws /usr/afsws

#####
# This next step will setup the itab entry for afs in the /etc/inittab file.
#
/usr/sbin/mkitab "rcafs:2:wait:/usr/vice/etc/rc.afs>/dev/console 2>&1 /
# Start AFS daemon "

#####
# This next step will add the /usr/afsws/etc /usr/afsws/bin and the
# /usr/vice/etc into your current path environment. You may want to
# update the path in .profile, or ksh.rc or other login setup.
PATH=$PATH:/usr/afsws/etc:/usr/afsws/bin:/usr/vice/etc:

```

block_usr_sample

This script creates a file of user names in a column format called **/tmp/usr.input**. The **usr.input** file can be used with the **spacs_cntrl** command. Once you are comfortable using Login Control, you can uncomment the **spacs_cntrl** command in this script. This file is in **/usr/lpp/ssp/samples**.

```
#!/usr/lpp/ssp/perl/bin/perl
#####
# Description:
#
# This sample script will build a file of users from the /etc/passwd
# file to input to spacs_cntrl.
#
# The following items should be checked and possibly changed to conform
# to the policy at your site.
#
# 1) uid threshold to start adding users to the file. Should be large
# enough to prevent system users such as root, adm, bin, lpd, etc.
# from being added to the file for spacs_cntrl. The arbitrary value
# used is 125.
#
# 2) flags for spacs_cntrl. The logging flag is issued to
# allow you to follow the actions of spacs_cntrl. You may wish to
# remove this flag once you are familiar with running spacs_cntrl.
# The -s flag suppresses error messages which are logged then -l is
# issued.
#####

# required files.

$usrfile="/tmp/usr.input";
$uidstart = 125;
$allokay = 0;
$syserror = 2;

# open the usr.input file

unless ( open(USRFILE,">$usrfile") ) {
    print "block_usr: Cannot open usr.input file.\n";
    exit ($syserror);
}

# write any names with a uid of 125 or above into file

while ( ($uname,$passwd,$uid) = getpwent ) {
    if ( $uid >= $uidstart ) {
        print USRFILE $uname,"\n";
    }
}

close(USRFILE);

# make sure file has entries

if ( -z $usrfile ) {
    print "block_usr: No entries in $usrfile. Not executing spacs_cntrl.\n";
    exit($syserror);
}
```

```

# Uncomment the following for this script to automatically run the spacs_cntrl command
# and block users.
# issue spacs_cntrl to block users in the file.

#system "/usr/lpp/ssp/bin/spacs_cntrl -s -l -f $usrfile block";
#rc=$?;
#if ( $rc != 0 ) {
#   $rc = ($rc >> 8);
#   print "block_usr: Possible error from spacs_cntrl. Return code = $rc\n";
#   exit($rc);
#}

exit ($alokay)

```

bootptab.info File

If it exists, the **/etc/bootptab.info** file is read by the **sphrdwrad** command for hardware Ethernet addresses to be placed in the SDR. If it does not exist, then the **sphrdwrad** command still runs but takes longer to collect the information from the hardware.

Each line should list an SP node, specified by node number or *frame,slot* followed by a blank and the hardware Ethernet address. The file should look similar to this:

```

17 02608C2E48D9
19 02608C2D6712
21 02608C2E49A4
23 02608C2E48E2

```

ibmSPDepNode.my

A copy of this MIB can be found in file **/usr/lpp/ssp/config/spmgrd/ibmSPDepNode.my**. The file contains configuration information for a dependent node adapter on a dependent node. It is used by the **spmgrd** daemon to send configuration attribute values to the SNMP Agent running on the dependent node.

```

-- Licensed Materials - Property of IBM
--
-- 5765-529
--
-- (C) Copyright IBM Corp. 1997 All Rights Reserved.
--
-- US Government Users Restricted Rights - Use, duplication or disclosure
-- restricted by GSA ADP Schedule Contract with IBM Corp.
--
-- Script Name:  depnode.my
--
-- Description:
--      definition of the depnode mib.
--
-- "@(#)75 1.2
-- src/ssp/spmgr/src/ibmSPDepNode.my,spmgr, ssp_rspr, rsprt2d9 2/7/9

DEPNODE-MIB DEFINITIONS ::= BEGIN

IMPORTS
    Counter, Gauge, TimeTicks, IPAddress, DisplayString, enterprises
        FROM RFC1155-SMI
    TRAP-TYPE

```

```

                                FROM RFC1215;

ibm          OBJECT IDENTIFIER ::= { enterprises 2 }

ibmProd      OBJECT IDENTIFIER ::= { ibm 6 }

ibmSP        OBJECT IDENTIFIER ::= { ibmProd 117 }

-----
--
--          IBM NODEDEP MIB
--
--
ibmSPDepNode OBJECT IDENTIFIER ::= { ibmSP 4 }

ibmSPDepNodeTable OBJECT-TYPE
    SYNTAX SEQUENCE OF IbmSPDepNodeEntry
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
        "This entity's table of dependent secondary nodes
         attached to IBM SP switch networks."
    ::= { ibmSPDepNode 1 }

ibmSPDepNodeEntry OBJECT-TYPE
    SYNTAX IbmSPDepNodeEntry
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
        "Configuration information for a dependent
         secondary node on SP switch network."
    INDEX { ibmSPDepNodeName }
    ::= { ibmSPDepNodeTable 1 }

IbmSPDepNodeEntry ::=
    SEQUENCE {
        ibmSPDepNodeName
            DisplayString,
        ibmSPDepNodeNumber
            INTEGER,
        ibmSPDepSwToken
            OCTET STRING,
        ibmSPDepSwARP
            INTEGER,
        ibmSPDepSwNodeNumber
            INTEGER,
        ibmSPDepIPAddr
            IpAddress,
        ibmSPDepNetMask
            IpAddress,
        ibmSPDepIPMaxLinkPkt
            INTEGER,
        ibmSPDepIPHostOffset
            INTEGER,
        ibmSPDepConfigState
            INTEGER,
        ibmSPDepSysName
            DisplayString,
        ibmSPDepNodeState
            INTEGER,
        ibmSPDepSwChipLink
            INTEGER,
        ibmSPDepNodeDelay

```

```

        INTEGER,
        ibmSPDepAdminStatus
        INTEGER
    }

ibmSPDepNodeName OBJECT-TYPE
    SYNTAX DisplayString
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "Identifier assigned to this node in its
        system's administrative environment."
    ::= { ibmSPDepNodeEntry 1 }

ibmSPDepNodeNumber OBJECT-TYPE
    SYNTAX INTEGER
    ACCESS read-write
    STATUS mandatory
    DESCRIPTION
        "Relative node number assigned to this node in the SP system.
        Note that this is distinct from switch node number."
    ::= { ibmSPDepNodeEntry 2 }

ibmSPDepSwToken OBJECT-TYPE
    SYNTAX OCTET STRING (SIZE(10))
    ACCESS read-write
    STATUS mandatory
    DESCRIPTION
        "Opaque correlator assigned in the SP system
        for this node's switch connection."
    ::= { ibmSPDepNodeEntry 3 }

ibmSPDepSwARP OBJECT-TYPE
    SYNTAX INTEGER {
        disabled(1),
        enabled(2)
    }
    ACCESS read-write
    STATUS mandatory
    DESCRIPTION
        "Indicates whether ARP is enabled for the partition
        of the switch network that contains the associated
        dependent secondary node.
        NOTE: An SNMP Agent may be responsible for switch
        nodes in several SP system partitions, and ARP
        may be enabled in none, some, or all of the partitions.

        A value of disabled(1) indicates that ARP is
        not to be used to resolve network protocol
        addresses to switch node numbers for the switch
        network. Instead, the mapping of IP addresses
        to switch node numbers is determined via the
        algorithm in the description of ibmSPDepIPHostOffset.

        A value of enabled(2) indicates that ARP is
        to be used to resolve network protocol
        addresses to switch node numbers for the switch
        network."
    ::= { ibmSPDepNodeEntry 4 }

ibmSPDepSwNodeNumber OBJECT-TYPE
    SYNTAX INTEGER (0..65535)
    ACCESS read-write
    STATUS mandatory

```

```

DESCRIPTION
    "The switch node number of the associated SP switch
    interface. In common networking terminology,
    this is the physical address for the interface
    on the SP switch network, and is also called
    the Node Device ID in the SP dependent node
    architecture."
 ::= { ibmSPDepNodeEntry 5 }

ibmSPDepIPAddr OBJECT-TYPE
SYNTAX  IpAddress
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
    "The IP address for the associated SP switch interface."
 ::= { ibmSPDepNodeEntry 6 }

ibmSPDepNetMask OBJECT-TYPE
SYNTAX  IpAddress
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
    "The subnet mask associated with the IP address of
    this entry. The value of the mask is an IP
    address with all the network bits set to 1 and all
    the host bits set to 0."
 ::= { ibmSPDepNodeEntry 7 }

ibmSPDepIPMaxLinkPkt OBJECT-TYPE
SYNTAX  INTEGER
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
    "The maximum number of bytes in the data portion of
    link-layer packets for IP datagrams on the
    SP switch network."
 ::= { ibmSPDepNodeEntry 8 }

ibmSPDepIPHostOffset OBJECT-TYPE
SYNTAX  INTEGER
ACCESS  read-write
STATUS  mandatory
DESCRIPTION
    "This value is the difference between switch node
    numbers and the host portion of the corresponding
    IP addresses when ARP is disabled on the SP switch
    network. It is the value to subtract from the
    host portion of an IP address to calculate the
    corresponding switch node number."
 ::= { ibmSPDepNodeEntry 9 }

ibmSPDepConfigState OBJECT-TYPE
SYNTAX  INTEGER {
    notConfigured(1),
    firmwareLoadFailed(2),
    driverLoadFailed(3),
    diagnosticFailed(4),
    microcodeLoadFailed(5),
    fullyConfigured(6)
}
ACCESS  read-only
STATUS  mandatory
DESCRIPTION
    "This value indicates the final configuration state

```

```

        of the corresponding SP switch interface."
 ::= { ibmSPDepNodeEntry 10 }

ibmSPDepSysName OBJECT-TYPE
    SYNTAX DisplayString
    ACCESS read-write
    STATUS mandatory
    DESCRIPTION
        "The fully-qualified domain name of the SP system
        partition that contains this dependent node. This
        name is administratively assigned in the SP system."
 ::= { ibmSPDepNodeEntry 11 }

ibmSPDepNodeState OBJECT-TYPE
    SYNTAX INTEGER {
        nodeUp(1),
        nodeDown(2)
    }
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "This value indicates the state of the dependent
        node with respect to its capability to handle
        Switch Manager protocol via its SP switch interface.

        A value of nodeUp(1) indicates that the node has
        been reinitialized or reconfigured sufficiently
        to handle the Switch Manager protocol to unfence
        the node if it were previously fenced from the
        SP switch network.

        A value of nodeDown(2) indicates that the node is
        no longer able to handle Switch Manager protocol
        messages on the SP switch network."
 ::= { ibmSPDepNodeEntry 12 }

ibmSPDepSwChipLink OBJECT-TYPE
    SYNTAX INTEGER
    ACCESS read-write
    STATUS mandatory
    DESCRIPTION
        "The send port number for packets from the
        adjacent switch chip to the dependent node.
        The dependent node needs this value to build
        the packet that it sends to the adjacent
        switch chip to trigger that chip to reply
        with a clock synchronization packet."
 ::= { ibmSPDepNodeEntry 13 }

ibmSPDepNodeDelay OBJECT-TYPE
    SYNTAX INTEGER
    ACCESS read-write
    STATUS mandatory
    DESCRIPTION
        "This value is an estimate of how many clock
        cycles elapse before a clock synchronization packet
        sent from the adjacent switch chip to the dependent node
        results in loading of the switch time-of-day register
        for the node.

        The dependent node needs this value to build
        the packet that it sends to the adjacent
        switch chip to trigger that chip to reply with
        a clock synchronization packet."

```



```

 ::= { ibmSPDepNodeEntry 14 }

ibmSPDepAdminStatus OBJECT-TYPE
    SYNTAX INTEGER {
        up(1),
        down(2),
        reconfigure(3)
    }
    ACCESS read-write
    STATUS mandatory
    DESCRIPTION
        "This value indicates the desired state of the dependent
        node with respect to its capability to handle the
        Switch Manager protocol via its SP switch interface.
        If the dependent node is not in the desired state,
        then the SNMP Agent triggers actions to change the state
        to the described state.

        A value of up(1) indicates that the node should be
        initialized and configured sufficiently to handle
        the Switch Manager protocol.

        A value of down(2) indicates that the node should be
        reset such that it is not able to receive and handle
        Switch Manager protocol messages on the SP switch
        network.

        A value of reconfigure(3) indicates that the SNMP Agent
        must send the switchInfoNeeded trap to request new
        SP switch configuration information for the dependent
        node. This causes the SNMP Manager to send the new
        configuration information for the dependent node MIB
        objects via set-request PDU(s). Once the new values
        are set by the SNMP Manager, the SNMP Agent reinitializes
        and reconfigures the dependent node sufficiently to
        handle the Switch Manager protocol."
 ::= { ibmSPDepNodeEntry 15 }

-- IBM enterprise-specific traps dependent secondary nodes on SP switch:

switchInfoNeeded TRAP-TYPE
    ENTERPRISE ibmSP
    VARIABLES { ibmSPDepNodeName }
    DESCRIPTION
        "A switchInfoNeeded trap signifies that the sending
        protocol entity needs SP switch configuration
        information for a dependent secondary node within
        the sender's configuration."
 ::= 1

switchConfigState TRAP-TYPE
    ENTERPRISE ibmSP
    VARIABLES { ibmSPDepConfigState }
    DESCRIPTION
        "A switchConfigState trap indicates the final
        configuration state of a dependent secondary node's
        SP switch interface."
 ::= 2

switchNodeUp TRAP-TYPE
    ENTERPRISE ibmSP
    VARIABLES { ibmSPDepNodeName }
    DESCRIPTION
        "A switchNodeUp trap indicates that a dependent

```

```

secondary node in the sender's configuration has
just been reinitialized or reconfigured and is able
to handle SP Switch Manager protocol messages."
 ::= 3

switchNodeDown TRAP-TYPE
  ENTERPRISE ibmSP
  VARIABLES { ibmSPDepNodeName }
  DESCRIPTION
    "A switchNodeDown trap indicates that a dependent
    secondary node in the sender's configuration
    is no longer able to handle Switch Manager protocol
    messages on the switch network."
 ::= 4

END

```

ibmSPMIB.my

A copy of this MIB can be found in file **/usr/lpp/ssp/config/snmp_proxy/ibmSPMIB.my**. The file defines SP-specific information for the creation of SNMP traps for the SP from AIX Error Log and Event Management Events. It also contains information about the nodes and the control workstation that make up the SP. The MIB is supported by the **sp_configd** daemon.

```

-- Licensed Materials - Property of IBM
--
-- 5765-529
--
-- (C) Copyright IBM Corp. 1996 All Rights Reserved.
--
-- US Government Users Restricted Rights - Use, duplication or disclosure
-- restricted by GSA ADP Schedule Contract with IBM Corp.
--
-- Script Name:  ibmSPMIB.my
--
-- Description:
-- definition of the ibmSP mib.
--

IBMSP-MIB  DEFINITIONS ::= BEGIN

IMPORTS
    Counter, Gauge, TimeTicks, IpAddress, DisplayString, enterprises
    FROM RFC1155-SMI
    TRAP-TYPE
    FROM RFC1215;

ibm      OBJECT IDENTIFIER ::= { enterprises 2 }

ibmProd  OBJECT IDENTIFIER ::= { ibm 6 }

ibmSP    OBJECT IDENTIFIER ::= { ibmProd 117 }

-----
--
--
--          IBM SP MIB

```

```

--

ibmSPConfig OBJECT IDENTIFIER ::= { ibmSP 1 }

ibmSPhostnodenumber OBJECT-TYPE
    SYNTAX INTEGER
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "A number from 0 to the total number of slots. It identifies
        the relative node number assigned to this processor."
    ::= { ibmSPConfig 1 }

ibmSPhostpartaddr OBJECT-TYPE
    SYNTAX IpAddress
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "ip address assigned to the system partition in which this host
        resides. If this host is acting as a control workstation (i.e.
        the value of ibmSPhostnodenumber.0 is 0), this will be the ip
        address of the default partition."
    ::= { ibmSPConfig 2 }

ibmSPCWScodeversion OBJECT-TYPE
    SYNTAX DisplayString
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The ssp release that is installed on the control workstation.
        Values are of the form 2.0, 2.1, etc. A NULL value means the
        release is not known."
    ::= { ibmSPConfig 3 }

ibmSPprimaryCWSname OBJECT-TYPE
    SYNTAX DisplayString
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "hostname of the primary control workstation."
    ::= { ibmSPConfig 4 }

ibmSPprimaryCWSoperstatus OBJECT-TYPE
    SYNTAX INTEGER {
        up(1),
        down(2)
    }
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "operational state of the primary control workstation."
    ::= { ibmSPConfig 5 }

ibmSPbackupCWSname OBJECT-TYPE
    SYNTAX DisplayString
    ACCESS read-only
    STATUS mandatory

```

```

DESCRIPTION
  "hostname of the backup control workstation."
  ::= { ibmSPConfig 6 }

ibmSPbackupCWSoperstatus OBJECT-TYPE
SYNTAX INTEGER {
  up(1),
  down(2)
}
ACCESS read-only
STATUS mandatory
DESCRIPTION
  "operational state of the backup control workstation."
  ::= { ibmSPConfig 7 }

ibmSPSystemTable OBJECT-TYPE
SYNTAX SEQUENCE OF IbmSPNodeEntry
ACCESS not-accessible
STATUS mandatory
DESCRIPTION
  "A list of SPNodeEntrys."
  ::= { ibmSPConfig 8 }

ibmSPNodeEntry OBJECT-TYPE
SYNTAX IbmSPNodeEntry
ACCESS not-accessible
STATUS mandatory
DESCRIPTION
  "Identifies a processor node residing in an SP frame."
INDEX { ibmSPpartitionaddr, ibmSPnodenumber }
  ::= { ibmSPSystemTable 1 }

IbmSPNodeEntry ::=
SEQUENCE {
  ibmSPpartitionaddr
    IpAddress,
  ibmSPnodenumber
    INTEGER,
  ibmSPframenumbr
    INTEGER,
  ibmSPslotnumber
    INTEGER,
  ibmSPslotsused
    INTEGER,
  ibmSPinitialhostname
    DisplayString,
  ibmSPreliablehostname
    DisplayString,
  ibmSPsysparname
    DisplayString,
  ibmSPcodeversion
    DisplayString
}

ibmSPpartitionaddr OBJECT-TYPE
SYNTAX IpAddress
ACCESS read-only
STATUS mandatory

```

DESCRIPTION
"ip address assigned to the partition in which this node resides."
::= { ibmSPNodeEntry 1 }

ibmSPnodenumber OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-only
STATUS mandatory
DESCRIPTION
"A number from 1 to the total number of slots. It identifies
the relative node number assigned to the processor."
::= { ibmSPNodeEntry 2 }

ibmSPframenumbers OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-only
STATUS mandatory
DESCRIPTION
"A number from 1 to the number of frames. It identifies
the number of the frame in which the processor node resides."
::= { ibmSPNodeEntry 3 }

ibmSPslotnumber OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-only
STATUS mandatory
DESCRIPTION
"A number from 1 to the number of slots in a frame. It
identifies the number of the first slot occupied by the
processor node within the frame."
::= { ibmSPNodeEntry 4 }

ibmSPslotsused OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-only
STATUS mandatory
DESCRIPTION
"A number from 1 to two. It identifies the number of slots
occupied by the processor."
::= { ibmSPNodeEntry 5 }

ibmSPinitialhostname OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
"The initial hostname assigned to the node during the SP customization
phase."
::= { ibmSPNodeEntry 6 }

ibmSPreliablehostname OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
"The hostname associated with the SP ethernet."
::= { ibmSPNodeEntry 7 }

```

ibmSPsysparname OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "The name of the system partition containing this processor
    node."
 ::= { ibmSPNodeEntry 8}

ibmSPcodeversion OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "The ssp release that is installed on the processor node.
    Values are of the form 2.0, 2.1, etc. A NULL value means the
    release is not known."
 ::= { ibmSPNodeEntry 9 }

    -- The format of traps containing errlog entries whose templates
    -- are defined with 'Alert=yes'.

-- ibmSPerrlogTrap TRAP-TYPE
-- ENTERPRISE ibmSP
-- VARIABLES { ibmSPellabel, ibmSPelidentifier, ibmSPeldatetime,
--   ibmSPelsequencenum, ibmSPelmachineid, ibmSPelnodeid,
--   ibmSPelclass, ibmSPeltype, ibmSPelresource,
--   ibmSPelrscclass, ibmSPelrscstype, ibmSPellocation,
--   ibmSPelvpd, ibmSPetdescription, ibmSPetprobcauses,
--   ibmSPetusercauses, ibmSPetuseraction, ibmSPetinstcauses,
--   ibmSPetinstaction, ibmSPetfailcauses, ibmSPetfailaction,
--   ibmSPetdetaildata}
-- DESCRIPTION
--   "These traps contain the contents of errlog entries formatted
--   into objects defining the contents of the errlog entry.
--   Since any single errlog entry does not contain all of the
--   fields defined in the collection of errlog templates, when a
--   object contains a null value, it will not be included in the
--   trap."
--   ::= ibmSPelidentifier.0

ibmSPerrlogVars OBJECT-TYPE
SYNTAX SEQUENCE OF IbmSPerrlogEntry
ACCESS not-accessible
STATUS mandatory
DESCRIPTION
    "A single SPNodeErrlogEntry."
 ::= { ibmSP 2 }

IbmSPerrlogEntry ::=
SEQUENCE {
    ibmSPellabel
        DisplayString,
    ibmSPelidentifier
        DisplayString,
    ibmSPeldatetime
        DisplayString,
    ibmSPelsequencenum

```

```

    DisplayString,
    ibmSPelmachineid
    DisplayString,
    ibmSPelnodeid
    DisplayString,
    ibmSPelclass
    DisplayString,
    ibmSPeltype
    DisplayString,
    ibmSPelresource
    DisplayString,
    ibmSPelrscclass
    DisplayString,
    ibmSPelrscstype
    DisplayString,
    ibmSPellocation
    DisplayString,
    ibmSPelvpd
    DisplayString,
    ibmSPetdescription
    DisplayString,
    ibmSPetprobcauses
    DisplayString,
    ibmSPetusercauses
    DisplayString,
    ibmSPetuseraction
    DisplayString,
    ibmSPetinstcauses
    DisplayString,
    ibmSPetinstaction
    DisplayString,
    ibmSPetfailcauses
    DisplayString,
    ibmSPetfailaction
    DisplayString,
    ibmSPetdetaildata
    DisplayString
}

```

ibmSPellabel OBJECT-TYPE

SYNTAX DisplayString

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The label associated with the error identifier defined by
the ibmSPelidentifier object."

::= { ibmSPerrlogVars 1 }

ibmSPelidentifier OBJECT-TYPE

SYNTAX DisplayString

ACCESS read-only

STATUS mandatory

DESCRIPTION

"A unique identifier defining the type of error entry written
to the system error log existing on the host from which the
trap originated."

::= { ibmSPerrlogVars 2 }

```

ibmSPeldatetime OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "A timestamp identifying the time that this error log entry
    was written to the system error log. It is of the form:
    day month day_of_month hour:min:sec"
 ::= { ibmSPerrlogVars 3 }

ibmSPelsequencenum OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "A decimal number which is the sequence number assigned to this
    entry. This is the value specified on the -l switch of the
    errpt command used to obtain the trap data on the host from
    which the trap originated."
 ::= { ibmSPerrlogVars 4 }

ibmSPelmachineid OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "A decimal number which is the machine ID of the host on which
    the trap originated. This is the value returned by the AIX
    'uname -m' command when issued on the host from which the trap
    originated."
 ::= { ibmSPerrlogVars 5 }

ibmSPelnodeid OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "An alpha-numeric string which is the node ID of the host on
    which the trap originated. This is the value returned by the AIX
    'uname -m' command when issued on the host from which the trap
    originated."
 ::= { ibmSPerrlogVars 6 }

ibmSPelclass OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "An alphabetic string which is the error class of this entry:
    H (hardware), S (software), O (errlogger command messages)."
 ::= { ibmSPerrlogVars 7 }

ibmSPeltype OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "An alphabetic string which is the severity of the error entry:

```



```

    'PEND' (impending loss of availability), 'PERF' (unacceptable
    performance degradation), 'PERM' (permanent), 'TEMP' (temporary),
    'UNKN' (unknown)."
```

```

::= { ibmSPErrlogVars 8 }

ibmSPelresource OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "The resource name associated with the error. For hardware
    errors this is a device name, for software errors this is
    the name of the failing executable, for operator command
    messages this is 'OPERATOR'."
::= { ibmSPErrlogVars 9 }

ibmSPelrscclass OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "The resource class associated with the error. For hardware
    errors this is a device class (or 'NONE')."
::= { ibmSPErrlogVars 10 }

ibmSPelrsctype OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "The resource type associated with the error. For hardware
    errors this is a device type (or 'NONE'), for software errors
    (when specified) this is an LPP."
::= { ibmSPErrlogVars 11 }

ibmSPellocation OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "For hardware errors, information about the location of the
    failing device (or 'NONE')."
::= { ibmSPErrlogVars 12 }

ibmSPelvpd OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "IBM or user supplied vital product data."
::= { ibmSPErrlogVars 13}

ibmSPetdescription OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "Error description."
```

```

::= { ibmSPerrlogVars 14}

ibmSPetprobcauses OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "Probable causes of the error."
::= { ibmSPerrlogVars 15 }

ibmSPetusercauses OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "User actions which may have caused the error."
::= { ibmSPerrlogVars 16 }

ibmSPetuseraction OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "Recommended actions the user may take to correct the error."
::= { ibmSPerrlogVars 17 }

ibmSPetinstcauses OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "Installation causes of the error."
::= { ibmSPerrlogVars 18 }

ibmSPetinstaction OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "User actions which may have been performed during installation
    to cause the error."
::= { ibmSPerrlogVars 19 }

ibmSPetfailcauses OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "A list of candidates which may be the source of the error."
::= { ibmSPerrlogVars 20 }

ibmSPetfailaction OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "A list of recommended actions which may be taken to correct
    the possible failures."

```

```

 ::= { ibmSPerrlogVars 21 }

ibmSPeIdetaildata OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
 "Detailed data about this particular error."
 ::= { ibmSPerrlogVars 22 }

ibmSPeMVariables OBJECT IDENTIFIER ::= { ibmSP 3 }

-- The format of traps containing errlog entries whose templates
-- are defined with 'Alert=yes'.

-- ibmSPeMEventTrap TRAP-TYPE
-- ENTERPRISE ibmSP
-- ibmSPeMEventVarValue
-- DisplayString,
-- ibmSPeMEventPredicate
-- DisplayString
-- VARIABLES { ibmSPeMEventID, ibmSPeMEventFlags, ibmSPeMEventTime,
-- ibmSPeMEventLocation, ibmSPeMEventPartitionAddress,
-- ibmSPeMEventVarsTableName, ibmSPeMEventVarsTableInstanceID,
-- ibmSPeMEventVarName, ibmSPeMEventVarValueInstanceVector,
-- ibmSPeMEventVarValuesTableInstanceID,
-- ibmSPeMEventVariableValue, ibmSPeMEventPredicate,
-- DESCRIPTION
-- "These traps contain the contents of events generated from the
-- PSSP Event Manager. The events have been formatted
-- into objects defining the contents of the event and where
-- the variable pertaining to the event and its value are located
-- in the ibmSP mib.
-- ::= ibmSPeMEventID.0

ibmSPeMEvent OBJECT-TYPE
SYNTAX SEQUENCE OF IbmSPeMEventEntry
ACCESS not-accessible
STATUS mandatory
DESCRIPTION
 "A single IbmSPeMEventEntry."
 ::= { ibmSPeMVariables 1 }

ibmSPeMEventEntry ::=
SEQUENCE {
  ibmSPeMEventID
    INTEGER,
  ibmSPeMEventFlags
    INTEGER,
  ibmSPeMEventTime
    TimeTicks,
  ibmSPeMEventLocation
    INTEGER,
  ibmSPeMEventPartitionAddress
    IpAddress,
  ibmSPeMEventVarsTableName
    DisplayString,
  ibmSPeMEventVarsTableInstanceID

```

```

    DisplayString,
    ibmSPEMEventVarName
    DisplayString,
    ibmSPEMEventVarValueInstanceVector
    DisplayString,
    ibmSPEMEventVarValuesTableInstanceID
    DisplayString,
    ibmSPEMEventVarValue
    DisplayString,
    ibmSPEMEventPredicate
    DisplayString
}

ibmSPEMEventID OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "The trap identifier assigned to the occurrence of the Event."
 ::= { ibmSPEMEvent 1}

ibmSPEMEventFlags OBJECT-TYPE
SYNTAX INTEGER {
    re-arm(1),
    false-predicate(2),
    unregister-event(4)
}
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "EM Flags."
 ::= { ibmSPEMEvent 2}

ibmSPEMEventTime OBJECT-TYPE
SYNTAX TimeTicks
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "Elapsed time between the activation of the SP proxy sub-agent
    and the occurrence of the event."
 ::= { ibmSPEMEvent 3}

ibmSPEMEventLocation OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "The number of the SP node from which the event was generated."
 ::= { ibmSPEMEvent 4}

ibmSPEMEventPartitionAddress OBJECT-TYPE
SYNTAX IpAddress
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "The ip address of the SP partition from which the event was
    generated."
 ::= { ibmSPEMEvent 5}

```

```

ibmSPEMEventVarsTableName OBJECT-TYPE
    SYNTAX DisplayString
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The name of the table in the SP MIB in which contains the variable
        definition objects."
    ::= { ibmSPEMEvent 6}

ibmSPEMEventVarsTableInstanceID OBJECT-TYPE
    SYNTAX DisplayString
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The instance identifier used to locate the row in the table named
        by the ibmSPEMEventVarsTableName object value which contains
        further information about the EM Event variable definition."
    ::= { ibmSPEMEvent 7}

ibmSPEMEventVarName OBJECT-TYPE
    SYNTAX DisplayString
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The name of the EM variable about which the event is recorded."
    ::= { ibmSPEMEvent 8}

ibmSPEMEventVarValueInstanceVector OBJECT-TYPE
    SYNTAX DisplayString
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The instantiation vector of the EM variable instance that resulted
        in the event."
    ::= { ibmSPEMEvent 9}

ibmSPEMEventVarValuesTableInstanceID OBJECT-TYPE
    SYNTAX DisplayString
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The instance identifier used to locate the row in the
        ibmSPEMVarValuesTable identifying for which instantiation of the named
        EM variable the value is reported. This is used to obtain the current
        variable value in the table to see if it has changed since the time
        the trap was issued."
    ::= { ibmSPEMEvent 10}

ibmSPEMEventVarValue OBJECT-TYPE
    SYNTAX DisplayString
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The value of the variable instance at the time of the event."
    ::= { ibmSPEMEvent 11}

ibmSPEMEventPredicate OBJECT-TYPE

```

```

SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "The predicate string which caused the event."
::= { ibmSPEMEvent 12}

```

```

ibmSPEMNodeDepVarsTable OBJECT-TYPE
SYNTAX SEQUENCE OF IbmSPEMNodeDepVarEntry
ACCESS not-accessible
STATUS mandatory
DESCRIPTION
    "A table of node-dependent EM Variable attributes. Variables
    in this table are only instantiated on the node containing the
    resource monitor. See the ibmSPEMNodeDepVarLocator object
    description."
::= { ibmSPEMVariables 2 }

```

```

ibmSPEMNodeDepVarEntry OBJECT-TYPE
SYNTAX IbmSPEMNodeDepVarEntry
ACCESS not-accessible
STATUS mandatory
DESCRIPTION
    "The attributes of an event manager variable. Each octet of
    the variable-length index string is encoded in a separate
    sub-identifier. The number of the sub-identifiers in the index
    is IMPLIED."
INDEX { ibmSPEMNodeDepVarName }
::= { ibmSPEMNodeDepVarsTable 1 }

```

```

IbmSPEMNodeDepVarEntry ::=
SEQUENCE {
    ibmSPEMNodeDepVarName
        DisplayString,
    ibmSPEMNodeDepVarDescr
        DisplayString,
    ibmSPEMNodeDepVarType
        DisplayString,
    ibmSPEMNodeDepVarDataType
        DisplayString,
    ibmSPEMNodeDepVarSBSFormat
        DisplayString,
    ibmSPEMNodeDepVarInitValue
        DisplayString,
    ibmSPEMNodeDepVarCurValueIndex
        INTEGER,
    ibmSPEMNodeDepVarClass
        DisplayString,
    ibmSPEMNodeDepVarVecEIDefn
        DisplayString,
    ibmSPEMNodeDepVarVecEIDescr
        DisplayString,
    ibmSPEMNodeDepVarPTXName
        DisplayString,
    ibmSPEMNodeDepVarDefPred
        DisplayString,
    ibmSPEMNodeDepVarEventDescr
        DisplayString,

```

```

    ibmSPEMNodeDepVarLocator
      DisplayString,
    ibmSPEMNodeDepVarOrderGroup
      DisplayString
  }

ibmSPEMNodeDepVarName OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
  "A resource variable name as defined to the Event Manager."
 ::= { ibmSPEMNodeDepVarEntry 1 }

ibmSPEMNodeDepVarDescr OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
  "A description of the variable, including its semantics."
 ::= { ibmSPEMNodeDepVarEntry 2 }

ibmSPEMNodeDepVarType OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
  "One of the strings Counter, Quantity or State."
 ::= { ibmSPEMNodeDepVarEntry 3 }

ibmSPEMNodeDepVarDataType OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
  "One of the strings long, float or SBS. SBS is only permitted
  for a variable of the type State. If the value of this
  object is SBS, then the definitions of the structured fields
  that comprise the structured byte string contained in the
  ibmSPEMNodeDepVarSBSFormat object."
 ::= { ibmSPEMNodeDepVarEntry 4 }

ibmSPEMNodeDepVarSBSFormat OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
  "If the ibmSPEMNodeDepVarDataType object with the corresponding
  instance id has a value of SBS, then this object describes the
  structured fields within the variable. Included for each structured
  field is the structured field name, followed by an equal sign,
  followed by the data type for the field. The structured fields
  are defined sequentially beginning with sequence number 0."
 ::= { ibmSPEMNodeDepVarEntry 5 }

ibmSPEMNodeDepVarInitValue OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only

```

```

STATUS mandatory
DESCRIPTION
  "The initial value of a resource variable before it is observed
  for the first time."
::= { ibmSPEMNodeDepVarEntry 6 }

ibmSPEMNodeDepVarCurValueIndex OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-only
STATUS mandatory
DESCRIPTION
  "An index into the ibmSPEMVarValuesTable to locate value
  instances for this variable."
::= { ibmSPEMNodeDepVarEntry 7 }

ibmSPEMNodeDepVarClass OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
  "The name of the class to which this resource variable belongs."
::= { ibmSPEMNodeDepVarEntry 8 }

ibmSPEMNodeDepVarVecElDefn OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
  "The name(s) of instantiation vector elements associated with the
  resource variable."
::= { ibmSPEMNodeDepVarEntry 9 }

ibmSPEMNodeDepVarVecElDescr OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
  "A description of the instantiation vector elements associated with
  the resource variable."
::= { ibmSPEMNodeDepVarEntry 10 }

ibmSPEMNodeDepVarPTXName OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
  "The name used to read and write the resource variable in the PTX
  shared memory."
::= { ibmSPEMNodeDepVarEntry 11 }

ibmSPEMNodeDepVarDefPred OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
  "The default predicate to be applied to the resource variable."
::= { ibmSPEMNodeDepVarEntry 12 }

```



```

ibmSPEMNodeDepVarEventDescr OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "A description of the event generated by the application of the
    default predicate."
 ::= { ibmSPEMNodeDepVarEntry 13 }

ibmSPEMNodeDepVarLocator OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "The name of the vector element whose value is the number of the
    node containing the variable instance."
 ::= { ibmSPEMNodeDepVarEntry 14 }

ibmSPEMNodeDepVarOrderGroup OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "The name of an Availability group. All events generated by the
    default predicates of resource variables within this order group
    name are guaranteed ordered delivery with respect to one
    another."
 ::= { ibmSPEMNodeDepVarEntry 15 }

ibmSPEMNodeIndepVarsTable OBJECT-TYPE
SYNTAX SEQUENCE OF IbmSPEMNodeIndepVarEntry
ACCESS not-accessible
STATUS mandatory
DESCRIPTION
    "A table of node-independent EM Variable attributes. No EM Locator
    element is defined for these variables so the location of the
    resource monitor is unknown within the partition. Variables
    in this table are only instantiated on the Control Work Station.
    See the ibmSPEMNodeDepVarLocator object description."
 ::= { ibmSPEMVariables 3 }

ibmSPEMNodeIndepVarEntry OBJECT-TYPE
SYNTAX IbmSPEMNodeIndepVarEntry
ACCESS not-accessible
STATUS mandatory
DESCRIPTION
    "The attributes of an event manager variable. Each octet of
    the variable-length index string is encoded in a separate
    sub-identifier. The number of the sub-identifiers in the index
    is IMPLIED."
INDEX { ibmSPEMNodeIndepPartaddr, ibmSPEMNodeIndepVarName }
 ::= { ibmSPEMNodeIndepVarsTable 1 }

IbmSPEMNodeIndepVarEntry ::=
SEQUENCE {
    ibmSPEMNodeIndepPartaddr
        IpAddress,
    ibmSPEMNodeIndepVarName

```

```

    DisplayString,
    ibmSPEMNodeIndepVarDescr
    DisplayString,
    ibmSPEMNodeIndepVarType
    DisplayString,
    ibmSPEMNodeIndepVarDataType
    DisplayString,
    ibmSPEMNodeIndepVarSBSFormat
    DisplayString,
    ibmSPEMNodeIndepVarInitValue
    DisplayString,
    ibmSPEMNodeIndepVarCurValueIndex
    INTEGER,
    ibmSPEMNodeIndepVarClass
    DisplayString,
    ibmSPEMNodeIndepVarVecElDefn
    DisplayString,
    ibmSPEMNodeIndepVarVecElDescr
    DisplayString,
    ibmSPEMNodeIndepVarPTXName
    DisplayString,
    ibmSPEMNodeIndepVarDefPred
    DisplayString,
    ibmSPEMNodeIndepVarEventDescr
    DisplayString,
    ibmSPEMNodeIndepVarOrderGroup
    DisplayString
}

ibmSPEMNodeIndepPartaddr OBJECT-TYPE
SYNTAX IpAddress
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "ip address assigned to the system partition in which the
    this resource variable resides."
 ::= { ibmSPEMNodeIndepVarEntry 1 }

ibmSPEMNodeIndepVarName OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "A resource variable name as defined to the Event Manager."
 ::= { ibmSPEMNodeIndepVarEntry 2 }

ibmSPEMNodeIndepVarDescr OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "A description of the variable, including its semantics."
 ::= { ibmSPEMNodeIndepVarEntry 3 }

ibmSPEMNodeIndepVarType OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory

```

DESCRIPTION

"One of the strings Counter, Quantity or State."
::= { ibmSPEMNodeIndepVarEntry 4 }

ibmSPEMNodeIndepVarDataType OBJECT-TYPE

SYNTAX DisplayString

ACCESS read-only

STATUS mandatory

DESCRIPTION

"One of the strings long, float or SBS. SBS is only permitted for a variable of the type State. If the value of this object is SBS, then the definitions of the structured fields that comprise the structured byte string contained in the ibmSPEMNodeIndepVarSBSFormat object."

::= { ibmSPEMNodeIndepVarEntry 5 }

ibmSPEMNodeIndepVarSBSFormat OBJECT-TYPE

SYNTAX DisplayString

ACCESS read-only

STATUS mandatory

DESCRIPTION

"If the ibmSPEMNodeIndepVarDataType object with the corresponding instance id has a value of SBS, then this object describes the structured fields within the variable. Included for each structured field is the structured field name, followed by an equal sign, followed by the data type for the field. The structured fields are defined sequentially beginning with sequence number 0."

::= { ibmSPEMNodeIndepVarEntry 6 }

ibmSPEMNodeIndepVarInitValue OBJECT-TYPE

SYNTAX DisplayString

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The initial value of a resource variable before it is observed for the first time."

::= { ibmSPEMNodeIndepVarEntry 7 }

ibmSPEMNodeIndepVarCurValueIndex OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-only

STATUS mandatory

DESCRIPTION

"An index into the ibmSPEMVarValuesTable table to locate value instances for this variable."

::= { ibmSPEMNodeIndepVarEntry 8 }

ibmSPEMNodeIndepVarClass OBJECT-TYPE

SYNTAX DisplayString

ACCESS read-only

STATUS mandatory

DESCRIPTION

"The name of the class to which this resource variable belongs."

::= { ibmSPEMNodeIndepVarEntry 9 }

ibmSPEMNodeIndepVarVecElDefn OBJECT-TYPE

SYNTAX DisplayString

ACCESS read-only

```

STATUS mandatory
DESCRIPTION
  "The name(s) of instantiation vector elements associated with the
  resource variable."
::= { ibmSPEMNodeIndepVarEntry 10 }

ibmSPEMNodeIndepVarVecEIDescr OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
  "A description of the instantiation vector elements associated with
  the resource variable."
::= { ibmSPEMNodeIndepVarEntry 11 }

ibmSPEMNodeIndepVarPTXName OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
  "The name used to read and write the resource variable in the PTX
  shared memory."
::= { ibmSPEMNodeIndepVarEntry 12 }

ibmSPEMNodeIndepVarDefPred OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
  "The default predicate to be applied to the resource variable."
::= { ibmSPEMNodeIndepVarEntry 13 }

ibmSPEMNodeIndepVarEventDescr OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
  "A description of the event generated by the application of the
  default predicate."
::= { ibmSPEMNodeIndepVarEntry 14 }

ibmSPEMNodeIndepVarOrderGroup OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
  "The name of an Availability group. All events generated by the
  default predicates of resource variables within this order group
  name are guaranteed ordered delivery with respect to one
  another."
::= { ibmSPEMNodeIndepVarEntry 15 }

ibmSPEMVarValuesTable OBJECT-TYPE
SYNTAX SEQUENCE OF IbmSPEMVarValuesEntry
ACCESS not-accessible
STATUS mandatory
DESCRIPTION
  "A table of current EM Variable values. Variables."

```

```

 ::= { ibmSPEMVariables 4 }

ibmSPEMVarValuesEntry OBJECT-TYPE
SYNTAX IbmSPEMVarValuesEntry
ACCESS not-accessible
STATUS mandatory
DESCRIPTION
    "The current value of an event manager variable instantiation.
    The value of the ibmSPEMVarValueIndex is assigned when the
    SP sub-agent (sp_configd) is initialized; it is contained in the
    ibmSPEMNodeDepVarCurValueIndex object within the ibmSPNodeDepVarsTable
    (if the variable is node-dependent) or in the
    ibmSPEMNodeIndepVarCurValueIndex object within the
    ibmSPEMNodeIndepVarTable (if the variable is node-independent).
    Each octet of the variable-length ibmSPEMVarValueInstanceVector value
    string is encoded in a separate sub-identifier, preceded by its
    length which may be 0 if its value is null. A 0 length indicates
    the function represented by the EM variable is not being monitored."
INDEX { ibmSPEMVarValueIndex, ibmSPEMVarValueInstanceVector }
 ::= { ibmSPEMVarValuesTable 1 }

IbmSPEMVarValuesEntry ::=
SEQUENCE {
    ibmSPEMVarValueIndex
        INTEGER,
    ibmSPEMVarValueInstanceVector
        DisplayString,
    ibmSPEMVarValuePartaddr
        IpAddress,
    ibmSPEMVarValueName
        DisplayString,
    ibmSPEMVarValue
        DisplayString
}

ibmSPEMVarValueIndex OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "An index used as part of the instance id to identify the
    object instance containing the current value of an EM
    resource variable instance "
 ::= { ibmSPEMVarValuesEntry 1}

ibmSPEMVarValueInstanceVector OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "The instantiation vector elements associated with the
    resource variable."
 ::= { ibmSPEMVarValuesEntry 2}

ibmSPEMVarValuePartaddr OBJECT-TYPE
SYNTAX IpAddress
ACCESS read-only
STATUS mandatory

```

```
DESCRIPTION
    "ip address assigned to the system partition in which the
    this resource variable resides."
::= { ibmSPEMVarValuesEntry 3}

ibmSPEMVarValueName OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "A resource variable name as defined to the Event Manager."
::= { ibmSPEMVarValuesEntry 4}

ibmSPEMVarValue OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "The current value of a resource variable."
::= { ibmSPEMVarValuesEntry 5}

END
```

Appendix D. Listing SP System Resource Variables

As described in Chapter 25, “The Event Management Subsystem” on page 337, the PSSP software includes a number of resource monitors that collect resource data and send it to the Event Management subsystem. You can also install resource monitors that:

- Are supplied by other IBM products or third parties.
- Have been written by programmers at your installation.

Once all of these resource monitors are running, you might want to determine what resource data is being collected in the system.

There are several ways to do this:

- You can look in the load list files that contain the Event Management configuration data for each of the resource monitors in your system.

These files contain the data that is loaded into the SDR as Event Management objects by the **haemloadcfg** command.

- You can use commands.

You can use the PSSP **SDRGetObjects** command to list Event Management objects that have been stored in the SDR. You can use the AIX **dspmsg** and **dspcat** commands to list individual messages and the contents of message catalogs that contain descriptions of resource variables.

- You can use the SP Perspectives graphical user interface.

You can use the Event Perspective portion of SP Perspectives to list information about the resource variables and instance vectors that have been defined in the system.

Note that there are no Event Management resource variables for dependent nodes.

Looking at the Event Management Load List Files

The load list file for the resource variables supplied by PSSP is the **/usr/lpp/spp/install/config/haemloadlist** file. This file is a plain text file that consists of stanzas, each of which describes an object in one of the Event Management SDR classes. For detailed information about the format of a load list file, see the man page for the **haemloadlist** file in *PSSP Command and Technical Reference*.

If you have installed other resource monitors, look at the load list files that contain their configuration data.

Getting Information about Event Management Objects in the SDR

To get information about all of the event management objects in the SDR, you can issue the **SDRGetObjects** command for each class of objects. You can also direct the output to a file, which you can then view or print. To do this, enter the following commands:

```

SDRGetObjects -G EM_Resource_Variable > rv_data
SDRGetObjects -G EM_Resource_Class > class_data
SDRGetObjects -G EM_Structured_Byte_String > sbs_data
SDRGetObjects -G EM_Instance_Vector > iv_data
SDRGetObjects -G EM_Resource_Monitor > rm_data

```

Specifying the **-G** flag ensures that you receive information about all of the resource variables in all of the system partitions in the SP system. If you are only interested in the resource variables in the current system partition, omit the **-G** flag.

Figure 43 on page 475 contains an example of the output of each **SDRGetObjects** command for the **IBM.PSSP.SP_HW.Node.lcd1** resource variable and its associated resource class, structured byte string, instance vector, and resource monitor definitions. For ease of presentation, the output has been reformatted vertically. The actual output of the command is horizontally formatted.

For a complete description of the attributes of Event Management SDR classes, see *RSCT: Event Management Programming Guide and Reference*.

You can also restrict the information you receive. For example, to get only the **rvName** and **rvClass** information for the **IBM.PSSP.SP_HW.Node.lcd1** resource variable, enter:

```

SDRGetObjects -G EM_Resource_Variable \
  rvName==IBM.PSSP.SP_HW.Node.lcd1 rvName rvClass

```

In response, the system displays:

```

rvName          rvClass
IBM.PSSP.SP_HW.Node.lcd1      IBM.PSSP.SP_HW

```

For more information about the **SDRGetObjects** command, see *PSSP Command and Technical Reference*.

Information from EM_Resource_Variable class for Resource Variable IBM.PSSP.SP_HW.Node.lcd1

rvName	IBM.PSSP.SP_HW.Node.lcd1
rvDescription	244
rvValue_type	State
rvData_type	SBS
rvInitial_value	0
rvClass	IBM.PSSP.SP_HW
rvPTX_name	""
rvPTX_description	""
rvPTX_min	""
rvPTX_max	""
rvPredicate	""
rvEvent_description	""
rvLocator	""
rvOrder_group	""
rvDynamic_instance	0
rvIndex_vector	NodeNum

Information from EM_Resource_Class class for Resource Class IBM.PSSP.SP_HW

rcClass	IBM.PSSP.SP_HW
rcResource_monitor	IBM.PSSP.hmrmd
rcObservation_interval	0
rcReporting_interval	0

Information from EM_Structured_Byte_String class for Resource Variable IBM.PSSP.SP_HW.Node.lcd1

sbsVariable_name	IBM.PSSP.SP_HW.Node.lcd1
sbsField_name	STRING
sbsField_type	cstring
sbsField_SN	0
sbsField_init_val	""

Information from EM_Instance_Vector class for Resource Name IBM.PSSP.SP_HW.Node

ivResource_name	IBM.PSSP.SP_HW.Node
ivElement_name	NodeNum
ivElement_description	256

Information from EM_Resource_Monitor class for Resource Monitor IBM.PSSP.hmrmd

rmName	IBM.PSSP.hmrmd
rmPath	/usr/lpp/ssp/bin/haemRM/hmrmd
rmArguments	IBM.PSSP.hmrmd
rmMessage_file	hmrmd_des.cat
rmMessage_set	1
rmConnect_type	server
rmPTX_prefix	0
rmPTX_description	0
rmPTX_asnno	0

Figure 43. Sample Output from PSSP SDRGetObjects commands

Displaying Event Management Descriptions

As shown in Figure 43 on page 475, the description of resource variable **IBM.PSSP.SP_HW.Node.lcd1** is message number 244 of message set 1 of the **hmrmd_des.cat** message catalog.

To display the description, enter:

```
dspmsg -s 1 hmrmd_des.cat 244
```

In response, the system displays:

```
"(string) lcd1: 'LCD line 1'."
```

To list all of the messages in the catalog and direct the output to a file, enter:

```
dspcat hmrmd_des.cat > hmrmdcat
```

For more information on the AIX **dspmsg** and **dspcat** commands, see *AIX Commands Reference*.

Getting Resource Variable Information from SP Perspectives

You can use the SP Event Perspective to get information about the resource variables that have been defined in the system. To do this:

- Start the Event Perspective

Enter:

```
spevent &
```

If you have more than one system partition defined, click on the icon that represents the system partition in which you are interested.

- Open the Create Condition dialog box.

From the Menu bar of the Event Perspective, click on the Actions button and, from the cascading menus, select Event Definitions and then Create. In response, the system displays a notebook that is open to the Create Event Definition page.

On the Create Event Definition page, click on the Create Condition button. In response, the system displays the Create Condition dialog box.

- Click the down arrow next to the Resource Variable Name field. In response, the system displays a list of all of the resource variables that are defined in the system partition you selected.

To get information about any particular resource variable, select it. In response, the system displays the resource variable's name in the Resource Variable Name field, its description in the Resource Variable Description field, and the format of the instance vector in the Resource Identifier Format field.

For complete information on the Event Perspective, see the online help.

Appendix E. The System Data Repository

The System Data Repository (SDR) is where your SP system configuration is stored. Information entered through the SMIT panels during installation goes into the SDR. Much of the information in the SDR can be viewed and some can be entered or changed using SP Perspectives or PSSP commands.

Understanding the SDR

The SDR is composed of objects and their attributes. Objects differ according to class. For example, an object belonging to the node class has a different set of attributes from an object in the adapter class.

Objects are *instances* of the object class. Some classes have many more objects than others. For example, there is only one object in the SP class for each SP system, but there are as many objects in the node class as there are nodes in the system.

An SDR object, however, is really only a specific set of attributes. Object attributes contain the meaningful data. To illustrate, Figure 44 compares two object classes in a sample configuration, the Frame class and the Adapter class.

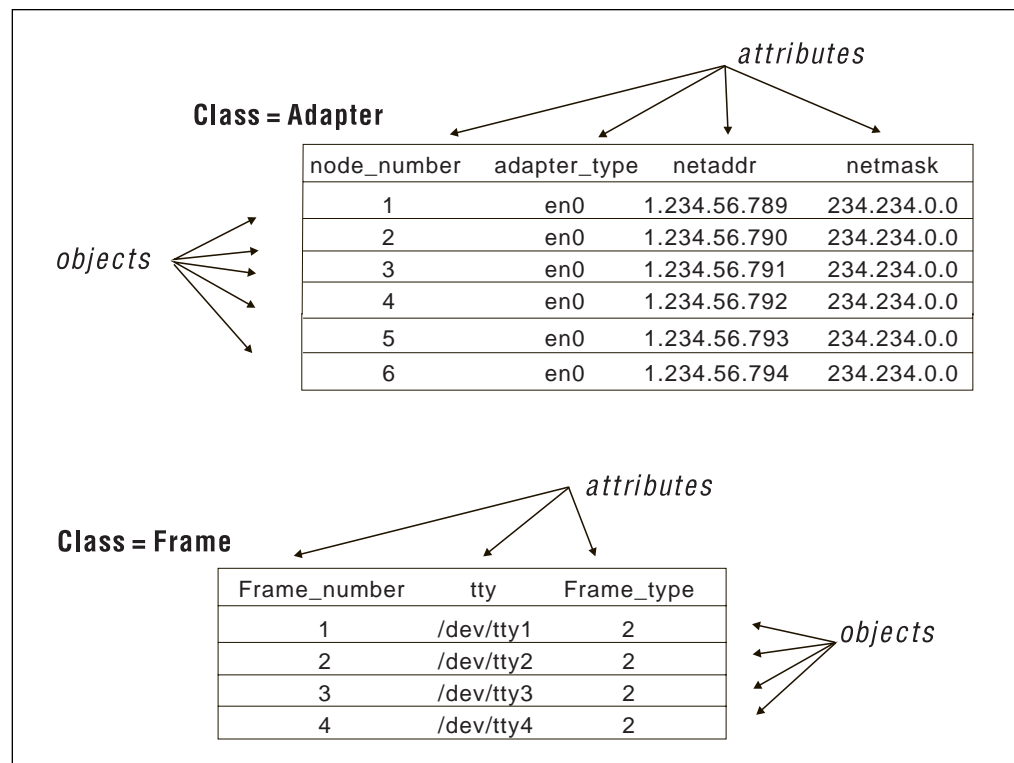


Figure 44. Comparing Two SDR Classes

The SP system provides *system partitioning* as a method for organizing the system into groups of nodes for various purposes such as testing new software and creating multiple production environments. Therefore there are two types of SDR classes: system and partitioned.

- System classes (contain objects that are common to all system partitions)

- Class = NodeControl
 - Class = NodeExpansion
 - Class = Frame
 - Class = SP
 - Class = SP_ports
 - Class = Switch
 - Class = SysNodeGroup
 - Class = Syspar_map
 - Class = TaskGuide
- Partitioned classes (contain objects that are specific to a system partition. The *definition* of a Partitioned class is common to all system partitions, but each system partition contains different objects for the class.)
 - Class = Adapter
 - Class = DependentNode
 - Class = EM_Condition
 - Class = GMT_Global_smt_nds
 - Class = GS_Config
 - Class = host_responds
 - Class = HSD_Minor_Number
 - Class = HSD_Table
 - Class = Network
 - Class = Node
 - Class = pmanrmdConfig
 - Class = PMAN_Subscription
 - Class = ProcessorExtensionNode
 - Class = RVSD_Restrict_Level
 - Class = SPDM
 - Class = SPDM_Nodes
 - Class = Subset
 - Class = switch_responds
 - Class = Switch_partition
 - Class = Syspar
 - Class = Syspar_ports
 - Class = Tec_Agent_Class
 - Class = TS_Config
 - Class = TS_Tunable
 - Class = VSD_Fence
 - Class = Volume_Group

- Class = VSD_Global_Volume_Group
- Class = VSD_Minor_Number
- Class = VSD_Table

For a complete list of attributes for each object, see “System Data Repository Classes and Attributes” on page 486.

In addition, there is a set of partitioned classes that are used by Event Management application programs:

- Class = EM_Resource_Variable
- Class = EM_Structured_Byte_String
- Class = EM_Resource_ID
- Class = EM_Resource_Class
- Class = EM_Resource_Monitor

For more information about these classes and their attributes, see *RSCT: Event Management Programming Guide and Reference*.

Accessing the SDR

There are two methods you can use to access the data in the SDR: SMIT panels and the SDR command line interface. The SDR command line interface is used by various SP commands. There should be no need to use the SDR command line directly and you should not do so unless you are instructed to by an IBM representative. In addition, many system management commands, such as **spbootins**, manipulate SDR data. You must be authorized in order to access the SDR. The SDR also performs authentication before allowing read-write access.

Authorization

The SDR has two levels of authorization, read-write and read-only. Table 17 shows the conditions under which each level of authorization is given.

<i>Table 17. SDR Authorizations</i>		
Machine	Root User	Non-Root User
Control Workstation	read-write	read-only
SP Node	read-write	read-only
Other	read-only	read-only

Command entries made without necessary authorization will fail.

Authentication

Read-write authorization is given after authentication. This consists of checking the user ID. If a root user is running on the same machine as the SDR (the control workstation), then read-write authority is given. For a root user who is not on the control workstation, the SDR server checks that the adapter connecting the client machine has an adapter class object in the SDR. If so, the user is given read-write authority. If not, the user gets read-only authority.

All nonroot users get read-only authority.

Locating the SDR Server

There are three ways in which a process can locate the SDR server. They are selected in the following order of preference but when one method is selected, no other methods are attempted. This means that if method 1 is available, only method 1 is attempted. If it fails, methods 2 and 3 are *not* attempted.

1. The destination is passed as the third parameter in the **SDROpenSession** command. This only works for library routines and therefore may be used by some SP subsystems, such as the Resource Manager. The destination is the hostname or TCP/IP address of the control workstation where the SDR runs.
2. The **SP_NAME** environment variable is set to the hostname or TCP/IP address of the control workstation where the SDR runs.
3. The **/etc/SDR_dest_info** file is present and has the **primary** record set to the hostname or TCP/IP address of the control workstation where the SDR runs. This file is installed on the control workstation and all SP system nodes along with the PSSP software.

The SDR_dest_info File

The **/etc/SDR_dest_info** file is created on the control workstation at system installation, and propagated to all nodes in the SP system. The **/etc/SDR_dest_info** file has the following format:

```
* comments have an asterisk in column 1
default: <TCP/IP address of default system partition>
primary: <TCP/IP address of node's partition>
nameofdefault: <hostname of default system partition>
nameofprimary: <hostname of name of the node's partition>
```

The **default** record identifies the default system partition. The default record is used at boot time so that the node can determine if it has changed system partitions.

Only the **primary** record is used by the SDR to locate the control workstation where the SDR server runs.

The SDR Daemon Log

The SDR daemon writes information to a log named **/var/adm/SPIlogs/sdr/sdrdlog.syspar_ip_addr.pid**, where *syspar_ip_addr* is the IP address of the system partition and *pid* is the process ID of the SDR daemon (**sdrd** process). This log will contain the date and time the process started, as well as problems encountered by the daemon in the course of operation.

Method 1 – Using SMIT

The RS/6000 SP System Management SMIT panel provides options for accessing configuration data in the SDR. To invoke this panel:

TYPE smit

- The System Management menu appears.

SELECT RS/6000 SP System Management

- The RS/6000 SP System Management menu appears.

The RS/6000 SP System Management menu offers the following options:

1. RS/6000 SP Configuration Database Management

The dialogs available through this path allow you to enter, list, and change information during the installation process about your nodes, primary and secondary external LANs, and switch connections, as well as site environment information. These tasks are explained in the *PSSP: Installation and Migration Guide*.

The choices from this menu are:

- *Enter Database Information*
 - *Site Environment Information*
 - *Frame Information*
 - *Node Database Information*
 - *Node Group Information*
 - *System Partition Configuration*
 - *Extension Node Database Information*
 - *Run setup_server Command*
- *List Database Information*
 - *List Site Environment Database Information*
 - *List Frame Database Information*
 - *List Node Database Information*
 - *List Node Configuration Information*
 - *List Node Switch Information*
 - *List Node Boot/Install Information*
 - *List Volume Group Information*
 - *List Node Expansion Information*
 - *List Accounting Information*
 - *List Node Group Database Information*
 - *List System Partition Database Information*
 - *List LAN Database Information*
 - *List Extension Node Database Information*
 - *List Extension Node Adapters Database Information*
- *Delete Database Information*
 - *Delete Frame Information*
 - *Delete Node Information*
 - *Delete Volume Group Information*
 - *Delete Adapter Information*
 - *Delete Node Expansion Information*
 - *Delete Node Group Information*
 - *Delete Extension Node Information*
 - *Delete Extension Node Adapter Information*

2. RS/6000 SP Cluster Management

The choices from this menu are:

- *Run setup_server Command*
- *Select System Tuning Parameters*
- *Perform Switch Operations*
- *Run enadmin Command*

3. RS/6000 SP Configuration Information

Using this path, you can display configuration information about your nodes, networks, file systems, and paging spaces.

The choices from this menu are:

- *List Node Hardware Information*
- *List Node Network Information*
- *List Node File System Information*

4. RS/6000 SP Users

Using this path, you can add and delete users, as well as change user management attributes such as passwords.

The choices from this menu are:

- *Add a User*
- *Change/Show Characteristics of a User*
- *Remove a User*

5. RS/6000 SP Installation/Configuration Verification

Use this path to check that your PSSP software options are installed correctly.

The choices are:

- *System Monitor Installation*
- *System Monitor Configuration*
- *System Data Repository*
- *System Management*
- *Communication Subsystem*
- *System Partition Configuration*
- *Switch Table API Installation*
- *Resource Manager Installation*
- *Resource Manager Configuration*

6. RS/6000 SP Supervisor Manager

The choices are:

- *Check for Supervisors That Require Action (Single Message Issued)*
- *List Status of Supervisors (Report Form)*
- *List Status of Supervisors (Matrix Form)*
- *List Supervisors That Require Action (Report Form)*
- *List Supervisors That Require Action (Matrix Form)*
- *Update *ALL* Supervisors That Require Action (Use Most Current Level)*
- *Update Selectable Supervisors That Require Action (Use Most Current Level)*

7. RS/6000 SP Resource Manager

The choices are:

- *Change/Show Configuration Data*
- *Start the Resource Manager*
- *Reconfigure the Resource Manager*
- *Stop the Resource Manager*

8. RS/6000 SP Log Management

The choices are:

- *AIX Error Log*
- *Syslog*

- *General Log Viewing*
- *Archive Logs*
- *Collect Logs for Service*

The menu selections run standard AIX commands and, in some cases, add information from the Hardware Monitor and reformat the output for usability. Let's examine the options on the list menu in more detail. The Configuration Information menu contains options that invoke the AIX commands listed in Table 18.

<i>Table 18. Commands Invoked by SMIT Panels</i>	
Select:	To:
List Node Hardware Information	Run the AIX lscfg command to display the name, location, and description of the devices related to the nodes
List Node Network Information	Run the AIX netstat -in command to show the state of the nodes' configured interfaces
List Node File System Information	Run the AIX df command to display the total spaces and available space on the node file systems

SELECT Any of the three choices

- SMIT runs the corresponding AIX command for all the nodes and displays the information in a scrollable window.

You can also list the configuration data using the **splstdata** command. See the book *PSSP: Command and Technical Reference* for complete syntax and examples.

Method 2 – Using the SDR Command Line Interface

A command line interface allows you to display, change, or delete the contents of an SDR object without invoking SMIT panels.

These commands are used by the PSSP components to operate on SDR data. You should not need to use these commands directly. Should you choose to use them, do so with caution. SDR contents can be corrupted or made inaccessible.

The following list briefly describes these commands. See the book *PSSP: Command and Technical Reference* for exact syntax.

SDRAddSyspar

The PSSP components use this command to create a new daemon using the System Resource Controller (SRC).

SDRArchive

The PSSP components use this command to create an archives file containing all current SDR classes attributes.

SDRChangeAttrValues

The PSSP components use this command to change the attribute values of an existing object.

SDRClearLock

The PSSP components use this command to unlock a class that is locked, regardless of who has the lock. This is for system administration use only and should be used with caution.

SDRCreateAttrs

The PSSP components use this command to create new attributes for an SDR class.

SDRCreateClass

The PSSP components use this command to create a new class of objects and its attributes.

SDRCreateFile

The PSSP components use this command to create an SDR file from an AIX file.

SDRCreateObjects

The PSSP components use this command to create one or more new objects and define their attribute values.

SDRCreateSystemClass

The PSSP components use this command to create a system class.

SDRCreateSystemFile

The PSSP components use this command to create a file that can be retrieved from any system partition.

SDRDeleteFile

The PSSP components use this command to delete an SDR file.

SDRDeleteObjects

The PSSP components use this command to delete target objects.

SDRGetObjects

The PSSP components use this command to query the values of target objects and attributes and prints them to stdout.

SDRListClasses

The PSSP components use this command to list the class names in the SDR.

SDRListFiles

The PSSP components use this command to first list all the files in the system area, then list all the files in the system partition area.

SDRMoveObjects

The PSSP components use this command to move objects from one system partition to another.

SDRRemoveSyspar

The PSSP components use this command to remove the entire contents of the subdirectory under system partitions. It uses the SRC to remove the daemon that serves the system partition.

SDRReplaceFile

The PSSP components use this command to replace the specified SDR file with the specified AIX file.

SDRRestore

The PSSP components use this command to overwrite the current SDR with the contents of an archived SDR file.

SDRRetrieveFile

The PSSP components use this command to create an AIX file from an SDR file.

SDR_test

The PSSP components use this command to verify that the installation and configuration of the SDR completed successfully.

SDRWhoHasLock

The PSSP components use this command to query the lock transaction ID for a specified object class.

Updating the `host_responds` Class (the `hrd` Daemon)

The `host_responds` class in the SDR is updated automatically by the `hrd` daemon. The `hrd` daemon is called from the `hr` script, which is under SRC control. `hr` will respawn if it, or the `hrd` daemon, is killed. `hr` and `hrd` run on the control workstation. Both the `hr` script and the `hrd` daemon are in directory `/usr/lpp/ssp/bin`. (The `hrctrl` script provides the same function as the `hr` script, but it follows the syntax of the `syspar_ctrl` command.)

The `hrd` daemon monitors the SP nodes using the Event Management subsystem. The `hrd` daemon acts as an Event Management client on the control workstation, using the EMAPI to subscribe for events pertaining to the `IBM.PSSP.Membership.LANAdapter.state` resource variables for the SP ethernet adapter on each node. Event Management notifies the `hrd` daemon when a node's adapters comes up or goes down and `hrd` updates the node's `host_responds` variable based on this value. For example, `host_repondsis` is set to 1 when its SP ethernet adapter is reported as up and is set to 0 when its SP ethernet adapter is reported as down.

The second method by which `hrd` can get node status is to use `fping`. `fping` is a program that runs `ping` to many nodes asynchronously. This second method also issues `snmpinfo` calls to each node, to detect a situation where `ping` succeeds but the node is not responding to user requests.

The `hr` script determines which method will be used. An environment variable named `HR_FPING` is set to `0` for heartbeat or `1` for `fping`. The default is `0`.

The line that chooses between the methods looks like this:

```
typeset -x HR_FPING=0
```

After making a change to this file, you should reset the `hrd` daemon by issuing `hr reset`. This will cause it to respawn and the change will be put in effect. Other options for configuring `hrd` are documented in comments in the `hr` script (in `/usr/lpp/ssp/bin`).

The reason for the second method (`fping`) is that it has different characteristics than the system heartbeat. It does not require a daemon to be on each node, so it may reduce cpu usage on the nodes by a small amount.

Backing Up the SDR

Backing up the SDR regularly is a good way to insure against the loss of your data. Whenever you change your SP system configuration, use the **SDRArchive** command to take a snapshot of the SDR contents and save it in a **tar** file. The *PSSP: Command and Technical Reference* contains syntax and usage information.

You can restore a saved image of the SDR using the **SDRRestore** command. Before restoring the SDR, be sure to stop the Resource Manager. (Refer to Chapter 6, "Managing SP Resources" on page 87.)

SDR Shadow Files

The SDR classes are stored in subdirectories under directory `/spdata/sys1/sdr`. The subdirectories of interest are `/system/classes` and `/partitions/partition_ip_address/classes`. The names of the files in those directories correspond to the names of the classes.

Before SDR commits changes to a class, the class contents are moved to a backup file in the same directory as the class. The backup file is named `class.shadow`, where `class` is the name of the class being written. If a power loss occurs or the SDR daemon is killed while the SDR class is being written, the class file that was being written at the time may not exist or may be corrupted. You should check for the existence of a `class.shadow` file. If one exists, take the following steps to restore the class.

1. Remove the corrupted `class` file (if one exists).
2. Rename the `class.shadow` file to `class`.
3. Restart the SDR daemon (**sdrd**) by issuing **sdr reset**.

The SDR daemon will restart automatically and recognize the renamed shadow file as the class.

System Data Repository Classes and Attributes

The following tables list all the SDR classes. You can specify these classes as targets in **SDRxxxx** commands. The value in the **Type** column is the datatype of the attribute: S=string, I=integer, F=floating point.

Class = Adapter

Attribute Name	Type	Description	Comments
node_number	I	Relative node number	
adapter_type	S	Adapter type as recorded in ODM (lsp0, enet0, enet1, fi0, fi1 etc.)	Set via customer through CMI. Only en0 is required.
netaddr	S	ip address for this interface.	Set via customer through CMI.

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
netmask	S	netmask for this interface.	Set via customer through CMI.
tok_rate	I	Speed of token rate, (4 or 16 mb)	
enet_type	S	bnc, dix, fiber, tp, or NA	
ucode_version	S	The file name of the microcode that is loaded	
prog_logic_version	S	Programmable logic	Currently not used
other_addrs	S	Alias IP addresses	Used by HACMP
subnet	S	Subnet for the adapter	
enet_rate	S	Ethernet adapter rate (10, 100, 1000, or auto)	10Mb/s, 100Mb/s, 1000Mb/s, or autosense
duplex	S	Ethernet adapter duplex (full, half, or auto)	Full duplex, half duplex, or autosense

Class = DependentNode

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
node_number	I	Represents the relative node number for the dependent node.	This is set by the -n flag of the endefnode command.
switch_node_number	I	Represents the switch port to which the dependent node is connected.	
switch_chip_port	I	Represents the number of the switch chip port to which the dependent node is connected.	
switch_chip	I	Represents the number of the switch chip to which the dependent node is connected.	
switch_number	I	Represents the switch board to which the dependent node is connected.	
switch_partition_number	I	Represents the number of the switch partition for this dependent node.	

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
reliable_hostname	S	Represents the host name associated with the dependent nodes network interface on the administrative network.	This is specified by the -a flag on the endefnode command.
management_agent_hostname	S	Represents the host name of the dependent node's SNMP Agent that interacts with the SNMP Manager on the SP. This host name must resolve to an IP address.	This is specified by the -s flag of the endefnode command. May be the same as or different from the reliable_hostname , depending on the characteristics of the extension node.
extension_node_identifier	S	A unique identifier in the administrative environment of an extension node. For example, for the Ascend GRF switched IP router, the slot number of the SP Switch Router Adapter on the GRF.	This is set by the -i flag of the endefnode command.
snmp_community_name	S	Represents the SNMP community name for the authentication field in the SNMP messages exchanged between the SNMP Agent on the dependent node and the SNMP Manager on the SP	This is specified by the -c flag of the endefnode command.

Class = EM_Condition

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
name	S	The name of this condition	
variable	S	The resource variable name to which this condition applies	
predicate	S	The condition	
rearm	S	The rearm condition	
specified	S	The specified resource ID elements	
unspecified	S	The unspecified resource ID elements	

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
description	S	A description of this condition	
type	S	Indicates if this condition is a default condition. default = default condition blank = a user-created condition	

Class = Frame

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
frame_number	I	Frame number	Assigned with the spframe command.
s1_tty	S	The tty port used for serial (s1term) communications.	A new attribute for S70 nodes assigned with the spframe command. It is undefined for SP frames.
tty	S	The tty port to be used for this frame (like /dev/tty1)	Assigned with the spframe command.
hardware_protocol	S	Type of hardware to be controlled: SP, SAMI.	Assigned with the spframe command. For S70 nodes, SAMI. The default is SP.
frame_type	S	Type of frame: Switch, no switch, all switch, or non-node.	From information returned by the monitor. For an SP Expansion I/O Unit the value is non-node. For SP-attached servers, this value is undefined.
MACN	S	Stands for Monitor and Control Node.	Hostname of the machine that monitors this frame.
backup_MACN	S	The backup monitor and control node for hardware monitoring with High Availability Control Workstation Connectivity function.	
slots	I	The number of slots in the frame.	A full-size frame is 79 inches (2.00 meters) and contains 16 slots. An SP Switch-8 frame is 49 inches (1.25 meters) and contains 8 slots. In a SP-attached server frame, this value is 1.

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
frame_in_config	I	Relative position of frame within expansion frame configuration (1,2,3,4).	In a SP-attached server frame, this value is undefined.
snn_index	I	Internal attribute used to index to determine correct switch node numbers.	In an SP-attached server frame, the value is undefined (it cannot contain a switch).
switch_config	I	Internal attribute used to determine switch configuration.	In an SP-attached server frame, this value is undefined (it cannot contain a switch).

Class = GMT_Global_smt_nds

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
node_1	I	Global Mount uses this node as primary smart node.	
node_2	I	Global Mount uses this node as backup smart node.	
node_3	I	Global Mount uses this node as backup smart node.	
node_4	I	Global Mount uses this node as backup smart node.	
node_5	I	Global Mount uses this node as backup smart node.	

Class = GS_Config

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
node_number	I	The node number	
gs_release_level	I	The Group Services release that the node is running	

Class = host_responds

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
node_number	I	Relative node number	This attribute gets mapped to frame/slot to retrieve DE value
host_responds	I	0 or 1 to indicate host is down or up.	This is really a query into the frame info written by the System Monitor.

Class = HSD_Minor_Number

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
minor_number	I	Unique minor numbers	

Class = HSD_Table

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
minor_number	I	Unique minor numbers	
stripe_size	I	Stripe width	
num_vsds	I	Number of vsds in a hsd.	
option	S	Whether the hsd will skip the first stripe on each underlying vsd in an hsd.	Value is protect_lvcb or not_protect_lvcb
HSD_name	S	Name of the hsd.	
VSD_name	S	Name of the vsd.	
size_in_MB	I	Size of table.	

Class = Network

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
networkName	S		Not currently filled.
networkType	S	The type of network. For example, FDDI, token ring, ATM, ethernet.	Not currently filled.
hbInterval	I	Heartbeat send frequency, in seconds, for this type of network. This overrides the value set in the TS_Tunable class.	
hbTolerance	I	Number of heartbeats from the neighboring node that can be missed before the neighbor is declared inoperative. This overrides the value set in the TS_Tunable class.	
hbEnable	I	Specified whether Topology Services should use this network for heartbeating. 1 = network should be used 0 = network should not be used	
VSD_name	S	Name of the vsd.	
size_in_MB	I	Size of table.	

Class = Node

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
node_number	I	Relative node number ((frame#-1)*16)+slot.	This number is computed from frame information and put here by system management.
hdw_enet_addr	I	Universal hardware ethernet identifier for this node.	Obtained through the system monitor during system initialization.
frame_number	I	Frame number.	Number from 1 to number of frames. Set during object creation.

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
slot_number	I	Node slot in frame.	Number from 1 to number of nodes. Set during object creation. For SP-attached server nodes, this value is set to 1.
slots_used	I	Nodes use 1, 2 or 4 slots. Distinguishes thin, wide or 604 High Node nodes.	Set by data obtained from system monitor. For SP-attached server nodes, this value is set to 1.
switch_node_number	I	Switch relative node number.	Set during object creation from data obtained from system monitor. For SP-attached server nodes, this value is set to the switch node number associated with the switch port on an existing SP frame. In a switchless system, it is assigned by the customer.
switch_chip_port	I	Switch port to which this node is connected.	Set during object creation from data obtained from system monitor. On a SP-attached server node, this value is the switch port on an existing SP frame to which the SP-attached server node is connected.
switch_chip	I	Switch chip number to which this node is connected.	Set during object creation from data obtained from system monitor. On a SP-attached server node, this value is the switch chip number for the switch port to which the SP-attached server node is connected.
switch_number	I	Switch number to which this node is connected.	Set during object creation from data obtained from system monitor. On a SP-attached server node, this value is the switch number for the switch port that the SP-attached server node is connected to.
initial_hostname	S	Hostname assigned to this node during customization.	Customer supplied from CMI at installation.
reliable_hostname	S	Hostname associated with SP ethernet.	Set during CMI processing from name entered for en0 interface name.
default_route	S	IP address of default route.	Customer supplied from CMI at installation.

<i>Attribute Name</i>	<i>Type</i>	<i>Description</i>	<i>Comments</i>
boot_server	I	Relative node number of node that will answer bootp request when necessary. (0 represent control workstation.)	Set during creation to be node 1 of frame. For node 1 in frame set to 0 to represent control workstation.
install_image	S	Path of net install image for this node. This field is initially set to DEFAULT , meaning that the default image is used.	Optionally set by customer from CMI interface.
last_install_time	S	Time and date that this system was last successfully installed.	Set during network installation.
last_install_image	S	server:/filepath of last image installed.	Set during network installation.
switch_protocol	S	This is the string LSP or CSSCI .	
switch_partition_number	I	Switch partition this node belongs to.	Default is 1.
bootp_response	S	install =full net install and customize; disk =use local disk to boot (ignore bootp request); maintenance =on reboot, go into Installation/Maintenance menus; customize =just use information from SDR to customize node's ODM; restore =restore of backup image via network install.	Initially set to customize during object creation. Changeable through CMI interface.
usr_maint	S	If set to true a node will perform usr client maintenance on boot.	Initialized to false , set through CMI.
VSD_adapter	S	The adapter on this node used by the vsd.	This is an adapter type to be joined with the adapter_type attribute of the Adapter class, where the node and Adapter node_number fields match.
VSD_max_buffer_count	I	Buffer cache size for vsd.	
VSD_request_blocks	I	The number of request blocks the vsd should allocate.	

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
VSD_pbufs	I	The number of pbufs the vsd should allocate.	
VSD_init_buffer_count	I	Buffer cache size for vsd.	
VSD_min_buddy_buffer_size	I	The minimum vsd buddy buffer size allocated to a single request.	
VSD_max_buddy_buffer_size	I	The maximum vsd buddy buffer size allocated to a single request.	
VSD_max_buddy_buffers	I	The size of the vsd buddy buffer, given in the number of buffers of max_buddy size.	
cfs_adapter	S	The adapter on this node used by Calypso and Global Mount.	
acct_class_id	S	An ID selected by administrator for grouping and merging acct data.	Default = default .
acct_enable	S	Whether acct is enabled for a node.	Default = default .
acct_job_charge	S	Used to determine the number of charge fee units to charge a user for exclusive use of the node. Its value is in units of "seconds per charge fee unit".	Default = 1.0 .
acct_exclude_enable	S	Indicates whether stop and end job accounting records will be generated for jobs having exclusive use of a node.	Default = false .
usr_server_ip	S	This is either an IP address or local .	Default is local .
usr_client_adapter	S	This will be enX , trX where X can be any number.	Default is en0 .
has_usr_clients	S	This will be either true or false.	Default is false.
code_version	S	Contains the ssp release that is installed on this node.	Default is nothing (unknown).

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
usr_gateway_ip	S	The IP address of the gateway to get to /usr server.	Default is 0 , meaning /usr server is not on a gateway.
VSD_maxIPmsgsz	I	Largest sized block of data that the IBM Virtual Shared Disk sends over the network for an I/O request.	Default is 24576 .
lppsource_name	S	The name of the LPP source resource from which to obtain AIX file sets for the installation of this node.	
processors_installed	I	The total number of processors in the node.	
processor_type	S	Specifies if the node is a uniprocessor or multiprocessor.	Value is UP or MP .
description	S	Provides a description.	
VSD_do_ip_checksum	S	Test to determine when to do IP checksum calculation.	
platform	S	Architecture of the node.	
hardware_control_type	S	Type of hardware control this node has.	If this node has a supervisor, this value is set to the node supervisor card type. For SP-attached server nodes, this value is set to 10 (decimal).
RVSD_version	I	Represents the RVSD version in VRMF format.	
selected_vg	S	The root volume group to install next.	Initially set to "rootvg".
dcehostname	S	Name by which DCE knows the host.	
expansion_list	S	List of <i>expansion_number</i> values from each NodeExpansion object for an expansion unit connected to this node.	Each <i>expansion_number</i> is separated by a comma. A null list means there are no NodeExpansion objects related to this node.

Class = NodeControl

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
type	S	Type of node.	For SP frame nodes, this value is set to the Node supervisor card type. For SP-attached server nodes, this value is set to 10 (decimal).
capabilities	S	Comma-separated list of capabilities (no embedded blanks).	For SP frame nodes, valid values are power, reset, tty, keySwitch, LED, networkBoot. For 332 MHz SMP wide or thin nodes, valid values are power, reset, tty, LED, networkBoot. For S70 nodes, valid values are power, reset, tty, LED, networkBoot.
slots_used	I	Number of slots used by this node type.	The correct number for each type of node. For S70 nodes, the valid value is 1.
platform_type	S	Hardware platform type.	For most SP nodes, the valid value is rs6k. For 332 MHz SMP wide or thin nodes, and S70 nodes, valid value is chrp.
processor_type	S	Specifies whether this is a uni-processor or multi-processor node.	Valid values are UP or MP as appropriate for each type of node. For S70 nodes, valid value is MP.
NC_timeout	I	The timeout value used by the nodecond command.	Allow more time for SMPs and nodes with a lot of I/O capability.
def_copies	I	The default number of copies of the root volume group to create for this node type.	Total number of copies is one or two.
def_pv_list	S	A list of physical volumes to be used for the root volume group and any mirrored copies.	There must be at least one extra disk for each copy of the root volume group.
def_quorum	S	The default quorum setting for this node type.	Valid values are true and false. Set it to false when mirroring.

Class = NodeExpansion

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
expansion_number	I	Unique SP number for this expansion unit.	Like the node_number value for a node but it is for the frame and slot location of this unit. Derived using $(frame_number-1)*16+slot_number$.
frame_number	I	Frame location.	Passed to hardmon by I/O Unit supervisor.
slot_number	I	Slot location.	Passed to hardmon by I/O Unit supervisor.
slots_used	I	Number of slots it occupies (1).	Taken from NodeControl table.
hardware_control_type	S	The supervisor card type (145).	Passed to hardmon by I/O Unit supervisor.
associated_node	I	Node number of node to which this is connected.	Passed to hardmon by I/O Unit supervisor (hardmon gets frame and slot numbers and converts to this node number).

Class = PMAN_Subscription

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
pmTargetType	S	Identifies the type of the parameter in the pmTarget field.	NODE_LIST , NODE_RANGE , or NODE_GROUP .
pmTarget	S	Identifies the set of nodes on which to execute the actions that are specified by the pmCommand, pmTrapid, and pmText fields in response to events and by the pmRearmCommand, pmRearmTrapid, and pmRearmText fields in response to rearm events. This field may contain a node range, a list of hostnames, or the name of a node group.	A pre-defined node group, ALLPMAN , is provided. This node group consists of all the nodes in the current system partition.
pmRvar	S	Specifies the resource variable name for the event.	Required.

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
pmIvec	S	Specifies a fully-qualified resource identifier for the event.	Required.
pmPred	S	Specifies an expression for the event.	Required.
pmCommand	S	The command to run in response to events.	Optional.
pmCommandTimeout	S	The amount of time, in seconds, to wait before assuming that the pmCommand has hung, and killing it.	Optional.
pmHandle	S	A label used to identify the subscription.	Required.
pmTrapId	I	An SNMP trap ID to send in response to events.	Optional.
pmPPSlog	I	Keeps track of whether the pmText field contains message text.	Optional.
pmThrottle			Not currently used.
pmRearmPred	S	Rearm expression which forms part of the event definition.	Optional.
pmRearmTrapId	I	An SNMP trap ID to send in response to rearm events.	Optional.
pmRearmPPSlog	I	Keeps track of whether the pmRearmText field contains message text.	Optional.
pmRearmCommand	S	The command to issue when the rearm event occurs.	Optional.
pmRearmCommandTimeout	S	The amount of time to wait, in seconds, before assuming the command has hung, and killing it.	Optional.
pmUsername	S	The user name of the subscriber event.	Defaults to root.
pmPrincipal	S	The Kerberos principal of the user who creates the subscription. This field establishes ownership of the subscription.	Required.

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
pmHost	S	The hostname of the node where the users issues the pmandef command to create the subscription.	Defaults to the local node.
pmDeactivated	S	Contains either NONE, ALL, or bitmask. NONE indicates all the nodes are activated. ALL indicates that all the nodes are deactivated. bitmask indicates that the subscription contains a mixture of activated and deactivated nodes.	Default is activated.
pmText	S	Text to be written to the AIX error log and BSD syslog facilities in response to events.	Optional.
pmRearmtext	S	Text to be written to the AIX error log and BSD syslog facilities in response to rearm events.	Optional.
pmUserLabel	S	Stores the name of the Named_Condition from the Event Management subsystem, if one was used to subscribe to a problem management record.	Used by the SP Perspectives GUI.
pmInitEval	I	Specifies whether to register a user's event definition with Event Management without prompting for immediate evaluation.	A value of 0 causes pmand to register the event without prompting for immediate evaluation. -1 causes pmand to request immediate evaluation.

Class = pmanrmdConfig

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
pmrmNodenum	I	The node number of the host on which pmanrmd is to instantiate the resource variable being defined.	
pmrmTargetType	S	Identifies the type of the parameter in the pmTarget field.	NODE_LIST, NODE_RANGE, or NODE_GROUP

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
pmrmTarget	S	Lists the targets on which daemons are to be configured. Can be a list of host names (NODE_LIST) separated by commas, a range of nodes (NODE_RANGE) as specified by the -n argument to hostlist , or the name of a node group (NODE_GROUP).	
pmrmRvar	S	Specifies the resource variable being defined.	Must be IBM.PSSP.User_state <i>nn</i> , where <i>nn</i> is 1-16.
pmrmCommand	S	Specifies a command to run at the specified interval.	
pmrmSamplnt	I	Specifies a sampling interval, in seconds, for the resource variable.	

Class = ProcessorExtensionNode

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
hostname	S	The hostname assigned to this object during configuration.	Can be set by the -n flag of the sppenode command or using Perspectives.
short_comment	S	To identify the subject of this specific object, such as an SP-controlled Netfinity server.	Can be set by the -s flag of the sppenode command.
long_comment	S	To further identify or describe the subject or purpose of this specific object.	Can be set by the -l flag of the sppenode command.
node_number	I	The relative node number derived by $((\text{frame\#}-1)*16)+1$.	Is set in this class for objects such as an SP-controlled Netfinity server.

Class = RVSD_Restrict_Level

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
domain	S		Not currently used.
level	I	Represents the functional level that RVSD will run at in VRMF format.	

Class = SP

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
control_workstation	S	Hostname of control workstation.	
cw_ipaddrs	S	IP addresses of control workstation.	
install_image	S	Full path of net install image.	Customer supplied from CMI at installation.
remove_image	S	If set to true the install images are erased after all nodes have been installed from this server.	Customer supplied from CMI at installation.
primary_node	S	Ethernet hostname of primary node.	Default is node 1 frame 1.
ntp_config	S	The kind of NTP installation done: none =Do not configure NTP for this site. consensus =setup NTP to configure Control Workstation as the NTP server and boot servers as NTP peers. timemaster =site has an existing NTP server. Configure NTP to user these. NTP_SERVERS will contain the NTP server hostnames. internet =The Control Workstation has access to the internet. Configure the Control Workstation to be an NTP server using the internet time server(s) defined in NTP_SERVERS.	From customer input from CMI.
ntp_server	S	Hostname of NTP server.	From customer input from CMI.
ntp_version	S	Default is 3.	Customer can override using CMI.
amd_config	S	True or False for whether the SP will provide automounter support	Customer can input in CMI, default is false.

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
print_config	S	false =do not print secure =install print in secure mode open =install print in open mode	Default is false, changed from customer via CMI.
print_id	S	User name to rsh to for secure mode printing	From customer via CMI.
usermgmt_config	S	false =do not install SP user management true =install SP user management code and SMIT interface	Default is false, changed from customer via CMI.
passwd_file	S	File containing password information (/etc/passwd, or other).	Default is /etc/passwd.
passwd_file_loc	S	Location of password file.	
homedir_server	S	Hostname of default user directory server for user management code.	From customer via CMI, default is control workstation.
homedir_path	S	Default path to user home directories.	From customer via CMI, default is /home/\$homedir_server.
filecoll_config	S	True or false if file collection management code should be installed.	From customer via CMI, default is false.
supman_uid	S	UID for supman.	
supfilesrv_port	I	File collections port number.	
spacct_enable	S	Flag indicating whether, by default, accounting is enabled on all nodes that have an accounting enabled attribute of default .	default is false .
spacct_actnode_thresh.	I	The SP accounting active node threshold indicates the percentage of nodes for which accounting data must be available, for merging and reporting of the data for a cycle to take place.	default is 80.
spacct_exclude_enable	S	The cluster exclusive use accounting enabled attribute- indicates whether accounting start and end job records will be generated for jobs having exclusive use of the node.	True or false. Default is false .
acct_master	I	Specifies which node is to act as the accounting master.	default is 0.
cw_has_usr_clients	S	Either true or false .	Default is false .

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
code_version	S	Contains the ssp release that is installed on the CWS.	Default is nothing (unknown).
layout_dir	S	Pathname of layout directory for current system partition configuration.	
authent_server	S	The type of authentication server used. The possible values are ssp (SP authentication services) (ssp.authent server, either primary or slave, on the control workstation), afs (afs server on either the control workstation or elsewhere), krb (SP authentication services or other Kerberos servers on other workstations only).	The default is krb . The value is set by install_cw .
backup_cw	S	Kernel hostname of backup control workstation.	If value is null, configuration is not HACWS.
ipaddrs_bucw	S	IP addresses of backup control workstation that are not service adapters and are not involved with fail over.	If value is null, the backup control workstation is not reachable as the backup during fail over periods.
active_cw	S	Active control workstation, either control workstation or backup control workstation.	If value is null, configuration is not HACWS.
cw_lppsource_name	S	The name of the LPP source resource from which to obtain NIM file sets for the control workstation.	You must ensure that the AIX level of the LPP source (indicated by the value given to cw_lppsource_name) matches the AIX level installed on your control workstation.
cs_dcehostname	S	Name by which DCE knows the control workstation.	

Class = SPDM

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
partition	S	Name of the SP partition using this object.	

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
central_mgr	S	PTPE Central Coordinator for PTPE monitoring hierarchy in this partition.	
active	I	Indicates if PTPE data collection is active.	
archive	I	Indicates if PTPE data archiving is active.	
organization	I		Obsolete field.
sampling_rate	I	Number of seconds between performance data samples when data collection is active.	
archive_rate	I	Number of seconds between performance data recordings when data archiving is active.	

Class = SPDM_NODES

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
reliable_hostname	S	Same as reliable_hostname in Nodes class.	
node_number	I	Same as node_number in Nodes class.	
node_group	I	Indicates to which PTPE monitoring group this node belongs.	
reports_to	S	reliable_hostname of this node's PTPE Data Manager in the PTPE monitoring hierarchy.	
num_reporters	I	Number of nodes reporting to this node in the PTPE monitoring hierarchy.	

Class = SP_ports

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
daemon	S	Name of a daemon.	

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
hostname	S	Host on which the daemon runs.	
port	I	Port that the daemon uses on node specified in hostname .	

Class = Subnet

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
subnet	S	The subnet ID, in "dot" notation (obtained by adding together an IP address for an adapter and its netmask).	All adapters in the same subnet will have the same subnet ID. This field is not currently filled.
networkName	S	Network to which this subnet belongs.	This field is not currently filled.

Class = Switch

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
switch_number	S	Switch Number	If less than 1000, switches connected to nodes. If greater than 1000, intermediate switches.
frame_number	I	Frame number	
slot_number	I	Slot number within frame.	Node switches equal 17. Intermediate switches equal 2,4,6,8,10,12,14,16.
switch_partition_number	I	Switch partition that the switch is on.	Default is 1.
switch_type	I	Switch supervisor card type (49, 50, or 129).	
clock_input	S	Clock input source.	SP Switch possible values: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 27, 28, 29, 30, 31, 32, 33, 34
switch_level	I	Microcode level from the switch supervisor.	

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
switch_name	S	Short model name of the switch.	
clock_source	I	Switch number of the switch providing the clocking signal for the current switch.	
clock_change	S	Clock input has changed since last Eclock .	

Class = Switch_partition

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
switch_partition_number	I	The partition number for High Performance Switch use.	No longer applicable.
topology_filename	S	The filename of the topology file that is stored in the SDR.	
primary_name	S	The reliable hostname of the switch primary node.	
arp_enabled	S	Address resolution protocol.	Default is no .
switch_node_number_used	S	Switch adapter's IP addresses were assigned based on the switch node numbers.	Default is yes .
run_phase_duration	I	Length of time in minutes that the switch stays in run phase. Range is 1–117281 minutes.	Default is 2 .
primary_backup_name	S	Reliable host name of this system partition's primary backup node.	
oncoming_primary_name	S	Reliable host name of the node that will become the primary node on the next Estart .	
oncoming_primary_backup_name	S	Reliable host name of the node that will become the primary backup node on the next Estart .	

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
num_nodes_success	I	Number of nodes in the system partition that successfully received the topology file.	
switch_max_ltu	I	Represents the maximum length of the data portion of an IP packet sent over the switch. This value is determined by internal switch support, and should not be changed.	
switch_link_delay	I	Represents the switch link delay value used by switch support when initializing the switch network. This value is determined by internal switch support, and should not be changed.	
autounfence	I	Tells the fault service daemon on the primary node to enable (1) or disable (0) the automatic unfence feature. Default is 1 to enable autounfence.	The value can be changed using the Estart command. The attribute takes effect at switch initialization or primary node takeover.

Class = switch_responds

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
node_number	I	Relative node number.	
switch_responds	I	0 or 1 if switch is inactive/active on the node.	This is written by switch code.
autojoin	I	Should this node join switch on startup.	
isolated	I	1 or 0 if this node is isolated from the switch or not.	
adapter_config_status	S	Status of the switch adapter in the node.	Status can be one of css_ready, diag_fail, microcode_load_fail, or not_configured.

Class = SysNodeGroup

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
name	S	Name associated with a node group.	The first character of a node group name must be a letter. The remaining characters may be letters, numbers, . (period), or _ (underbar).
defList	S	Nodes and node groups which are members of this node group.	For a node group named "A", a list of nodes and node groups that belong to "A" directly (not through member node groups).

Class = Syspar

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
syspar_name	S	Hostname of system partition.	
ip_address	S	IP address of system partition.	
install_image	S	The image installed on this system partition.	
syspar_dir	S	The directory from which this system partition was created.	
code_version	S	PSSP components code version in this partition.	
haem_cdb_version	S	Event Management Configuration Database (EMCDB) version string.	
auth_install	S	Set of authentication methods to be installed on nodes in this partition.	
auth_root_rcmd	S	Set of root authorization files defined for this partition.	
auth_methods	S	Set of active authentication methods for this partition.	

Class = Syspar_map

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
syspar_name	S	Short hostname of a system partition.	
syspar_addr	S	IP address of a system partition.	
node_number	I	A node number in this system partition.	The node number assigned to the node.
switch_node_number	I	The switch node number of this node.	For SP-attached server nodes, this value is the switch node number for the switch port to which the SP-attached server node is connected. In a switchless system, this value is set by the customer.
used	I	Is this node on IC bus of frame controller?	For a SP-attached server, this value is set to 1.
node_type	S	The type of node: standard, dependent.	For a SP-attached server node, this value is set to standard.

Class = Syspar_ports

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
subsystem	S	Name of a system partition-sensitive subsystem.	
port	I	Port number used for intrasubsystem communication.	

Class = TaskGuide

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
tgsbase	S	Base name of the TaskGuide script that was used.	Set by TaskGuide when creating an object of this class. For example, <i>confnode</i> would correspond to <i>confnode.sgs</i> , the <i>Configure New Nodes</i> TaskGuide.
datetime	S	A data/time stamp recording when the TaskGuide was started (format varies with locale).	Set by TaskGuide when creating an object of this class.
userid	S	The userid that ran the TaskGuide.	Set by TaskGuide when creating an object of this class.
status	S	An indication of whether the TaskGuide was completed or not.	Initialized to <i>incomplete</i> when TaskGuide creates an object of this class. Changed to <i>complete</i> when the TaskGuide is completed.
notes	S	Terse notes that further identify the TaskGuide run to which this object pertains.	Generally left uninitialized when TaskGuide creates an object of this class. Set (and possibly reset) by TaskGuide just as soon as it has some identifying notes formulated. For example, the <i>Add Frames</i> TaskGuide might set <i>Frames 2,3</i> .
tgvals	S	The name of the SDR system file where the TaskGuide checkpoint or resume values have been saved.	Initialized by TaskGuide when creating an object of this class. The format of the SDR system file name is <i>tgsbase.datetime.tglvals</i> . Will be changed to a null string at the same time TaskGuide changes the status attribute to <i>complete</i> .
tglog	S	The name of the SDR system file where the TaskGuide log of the run has been saved.	Set by TaskGuide when creating an object of this class. The format of the SDR system file name is <i>tgsbase.datetime.tglog</i> .

Class = Tec_Agent_Class

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
rv_name	S	Event Manager resource variable name.	
rv_number	I	Number of T/EC class (for indexing).	

Class = TS_Config

For the default settings of the attributes in this class, see the man pages for the **hatsctrl** command in *PSSP Command and Technical Reference*.

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
Frequency	I	The frequency, in heartbeats per second, with which heartbeats are sent out.	
Sensitivity	I	The number of heartbeats from the neighboring node that can be missed before the neighbor is declared inoperative.	
Run_FixPri	I	Whether the execution priority is fixed or floating (1=fixed, 0=floating).	
FixPri_Value	I	The value of the execution priority that is used on the set_priority system call.	
Log_Length	I	The maximum number of lines in a log file. The file is wrapped after it reaches this limit.	
Pinning	S	Determines whether the Topology Services daemon should run with its text or data pinned in memory. Text = pin text area Data = pin data area Proc = pin text and data areas NULL = does not ping any area	This field is not currently filled. The daemon has its text area pinned by default.

Class = TS_Tunable

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
networkType	S	The type of network. For example, FDDI, token ring, ethernet, and so on.	
hbInterval	I	Heartbeat send frequency, in seconds, for this type of network.	
hbTolerance	I	Number of heartbeats from the neighboring node that can be missed before the neighbor is declared inoperative.	

Class = Volume_Group

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
node_number	I	The node number for Volume_Group. It is an integer representing a node number.	
vg_name	S	The customer supplied volume group name.	This is a customer supplied name for easy reference. It is initially set to "rootvg".
pv_list	S	A list of physical volumes. Valid formats are: hdisk, hdisk, connwhere, connwhere, ...connwhere location:location:...location:	Initially set to "hdisk0".
rvg	S	Specifies whether the volume group is a root volume group. Valid values are true or false.	Initially set to "true".
quorum	S	Specifies whether the quorum is set to on for this volume group. Valid values are true or false.	Initially set to "true".
copies	I	Specifies the number of copies for the volume group. Valid values are 1 or 2 or 3.	Initially set to 1.
mapping	S	Specifies whether mapping is set to on. Valid values are true or false.	Initially set to "false".

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
install_image	S	The name of the mksysb install image to install next.	Initially set to "default".
code_version	S	PSSP code version to use for next install.	Derived from Syspar.
lppsource_name	S	Name of the lppsource resource to use for next install.	Initially set to "default".
boot_server	S	The node_number of the boot/install server.	Default depending upon the node location.
last_install_time	S	A date string specifying when the node was last installed.	Initial
last_install_image	S	A string specifying the name of the last image installed on the node.	Initial
last_bootdisk	S	A string specifying the logical device name of the last volume from which the node booted.	Initial

Class = VSD_Fence

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
minor_number	I	The unique minor number of a vsd that is fenced out of more than one node.	
commit	I		0 if in process of being committed; 1 if committed.
Map	S	A hex dump of the map designating the nodes that have IBM VSD fenced out.	

Class = VSD_Global_Volume_Group

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
global_group_name	S	Global Volume Group name.	

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
local_group_name	S	Local Volume Group name.	
primary_node	I	Primary node for this volume group.	This is a relative node number.
secondary_node	I	Secondary node for this volume group.	This is a relative node number and is optional.
eio_recovery	I	Denotes if recovery should be made from a hardware error.	
recovery	I	Denotes if recovery was made from a hardware error.	0 = recovery was not made; 1 = recovery was made.
primary_ts	S	Primary node for a tail.	
secondary_ts	S	Secondary node for a tail.	

Class = VSD_Minor_Number

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
minor_number	I	Unique minor numbers.	

Class = VSD_Table

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
VSD_name	S	Volume group name.	
global_group_name	S	Global Volume Group name.	Same names as global_group_name in VSD_Global_Volume_group.
logical_volume_name	S	The name of the logical volume comprising this vsd.	
minor_number	I	Minor number from minor table.	
option	S	Options associated with this vsd.	cache or nocache are the permissible values.
size_in_MB	I	The size of the logical volume pointed to by a vsd.	

<i>Attribute Name</i>	<i>T y p e</i>	<i>Description</i>	<i>Comments</i>
lv_blk0_pdev	I	The physical device pointed to by the logical volume.	
lv_blk0_pbn	I	The offset a logical volume resides on on the physical device.	

Appendix F. IP Address and Host Name Changes for SP Systems

This appendix describes how to change the primary SP Ethernet IP address and host name for SP nodes and SP-attached servers, or the primary IP address and host name for the control workstation and SP-controlled Netfinity servers. IBM suggests you avoid making any host name or IP address changes if possible. The tasks are tedious and in some cases require rerunning the SP installation steps. For more information on SP networks, see the book *IBM RS/6000 SP: Planning Volume 2, Control Workstation and Software Environment*.

Changing the IP address or host name of SP nodes affects the entire SP system. IP address and host names are located in the SDR using objects and attributes. IP address and host names are also kept in system-related files that are located on SP nodes and the control workstation. It is imperative that all new IP addresses and host names being used are resolvable on your SP system.

The following PSSP components and related products require careful consideration and attention when you change the IP address or host names:

- Network Installation Manager (NIM)
- System partitioning
- IBM Virtual Shared Disk
- High Availability Control Workstation (HACWS)
- High Availability Cluster Multi-Processing (HACMP)
- RS/6000 Cluster Technology (RSCT) services
- Problem management subsystem
- Performance monitor services
- Distributive Computing Environment (DCE)
- Extension nodes
- SP-controlled Netfinity servers

SDR Objects with IP Address or Host Names

The following SDR objects reference the host name and IP address in the SP system for PSSP subsystems:

Adapter	Specifies the IP addresses used with the switch css0 adapter, or the Ethernet, FDDI, or token ring adapters.
Frame	Specifies the Monitor and Control Node MACN , and HACWS backup_MACN attributes on the control workstation that work with host names
JM_domain_info	Works with the host names for Resource Manager domains
JM_Server_Nodes	Works with the host names for Resource Manager server nodes

Node	Works with the initial or reliable host names and uses the IP address for SP nodes and boot servers. The nodes are organized by system partitions.
Pool	Works with host names for Resource Manager pools
SP	Works with control workstation IP address and host names. Uses the host name when working with Network Time Protocol (NTP) printing, user management, and accounting services
SP_ports	Works with the host name used with hardmon and the control workstation
Switch_partition	Works with host name for primary and backup nodes used to support the css SP switch
Syspar	Works with the IP address and SP_NAME with system partitions
Syspar_map	Provides the host name and IP address on the control workstation for system partitions
pmandConfig	Captures the SP node host name data working problem management
SPDM	Works with host name for Performance Monitor status data
SPDM_NODES	Works with host name for SP nodes and organized by system partition
DependentNode	Works with host name for dependent extension node.
DependentAdapter	Works with IP address for dependent extension node adapter.
ProcessorExtensionNode	Works with host name for SP-controlled Netfinity server.

SP System Files with IP Address and Host Names

The following files contain the IP address or host name that exist on SP nodes and the control workstation. IBM suggests that you look through these files when completing the procedures for changing host names and IP addresses for your SP system. The following files are available for PSSP systems:

- /.rhosts** Contains host names used exclusively with **rcmd** services.
- /.klogin** Contains host names used with authentication **rcmd** services.
- /etc/hosts**
Contains IP addresses and host names used with the SP system.
- /etc/resolv.conf**
Contains the IP address for Domain Name Service (DNS) (Optional).
- /var/yp/nis**
References the host name and IP address with Network Information Service (NIS).
- /etc/krb5.conf**
Works with host name for DCE.

/etc/krb.conf

Works with the host name for the authentication server.

/etc/krb.realms

Works with the host name of the SP nodes and authentication realm.

/etc/krb-srvtab

Provides the authentication service key using host name.

/etc/SDR_dest_info

Specifies the IP address of the control workstation and the SDR

/etc/ssp/cw_name

Specifies IP address of control workstation host name on SP nodes that work with node installation and customization.

/etc/ssp/server_name

Specifies IP address and host name of the SP boot/install servers on SP nodes working with node customization.

/etc/ssp/server_hostname

Specifies IP address and host name of the SP install servers on SP nodes working with node installation.

/etc/ssp/reliable_hostname

Specifies IP address and host name of the SP node working with node installation and customization.

/etc/ntp.conf

Works with the IP address of the NTP server (Optional).

/etc/filesystems

Can contain the IP address or host name of NFS systems (mainly used on **/usr** client systems).

/ftpboot/host.config_info

Contains the IP address and host name for each SP node. It is found on the control workstation and boot servers.

/ftpboot/host.intstall_info

Contains the IP address and host name for each SP node. It is found on the control workstation and boot servers.

/ftpboot/host-new-srvtab

Provides authentication service keys using host name. It is found on the control workstation and boot servers.

/etc/rc.net

Contains the alias IP addresses used with system partitions

/etc/niminfo

Works with the NIM configuration for NIM master information

/etc/sysctl.acl

Uses host name that works with Sysctl ACL support

/etc/logmgt.acl

Uses host name that works with Error Log Mgt ACL support

/spdata/sys1/spmon/hmacls

Uses short host name that works with hardmon authentication services

/etc/jmd_config.SP_NAME

Works with host names for Resource Management on the control workstation for all defined SP_NAME syspars.

/usr/lpp/csd/vsdfiles/VSD_ipaddr

Contains the SP node IBM Virtual Shared Disk adapter IP address

/spdata/sys1/ha/cfg/em.SP_NAMEcdb.data

Uses Syspar host name that works with configuration files for Event Management services.

/etc/ha/cfg

Uses Syspar host name that works with configuration files for Event Management services.

/var/ha/run/rsct_services

Uses Syspar host name that contains the run files for the RSCT services, which include **haem** (Event Manager), **hats** (Topology Services) **hags** (Group Services - Ethernet), and **hagsglam** (Group Services - switch).

/var/ha/log/rsct_services

Uses Syspar host name that contains the log files for the RSCT services.

/var/adm/SPIlogs/pman/data

Uses Syspar host name that contains the log files for the Problem Management subsystem.

/etc/services

Specifies short host name based on *SP_NAME* partition that work with RSCT services port numbers

/etc/auto/maps/auto.u

Contains host names of the file servers providing NFS mounts to Automount

/etc/amd/amd-maps/amd.u

Contains host names of the file servers providing NFS mounts to Amd

/var/sysman/sup/file_collection/host

Contains host names of the control workstation and of the SP nodes, servers, and other workstations that are clients of this *file_collection*.

Changing the IP Address and Host Names for PSSP Systems

Changing the IP address and host names for your SP system will cause changes to the control workstation and SP nodes. To change the IP address and host names, do the following:

1. Create a **mksysb** backup for the control workstation and backup critical file systems on the control workstation and SP nodes.
2. Run the procedures based on the configuration of the SP system. There are network and SP configuration updates required on both the control workstation and SP nodes. IBM suggests that you document and understand all the required IP address and host name changes prior to performing this activity. It is necessary to understand which SP nodes will be the boot server nodes in your SP configuration.

3. Update the **en0** network interface and update system files on each SP node that have IP address or host name changes. You will need to know the new selected IP addresses and host names being used by the SP nodes, control workstation, and gateway servers. For each node you must ensure its `reliable_hostname` attribute is identical to the host name returned by the `host` command for its SP Ethernet IP address. For example, if the **en0** IP address of a node is 123.45.678.90, and 'host 123.45.678.90' is changed to give a default host name of v64n90.xen.kry.arg.com, then the `reliable_hostname` in the SDR must be identical. This is required so that the SP node can properly communicate with the control workstation during reboot activity.
4. Work on the control workstation and make the necessary changes to the SDR. This includes updates to reset PSSP daemons, PSSP system resource objects, and system partition files. You need to update files on the control workstation to reflect the new host names and IP addresses. During this activity, the SP system should not be available to users. The SP system is not ready until the SP nodes reference the new IP address and host name changes.
5. Set the SDR boot response to **customize** and then execute a reboot for the SP nodes. After the SP nodes complete the reboot and are customized, you should verify that the files on the SP nodes reflect the new IP address and host name changes.
6. The PSSP installation support requires the **admin** user to replace the NIM master objects. It might be necessary to update the NIM network and machine resources to match the new IP address or host name. The NIM updates are needed for the control workstation and boot/install server nodes. There are installation commands in PSSP that support the implementation of NIM interfaces.
7. The PSSP system provides RSCT services which are subsystems and interfaces that are installed on the control workstation and SP nodes. These subsystems (**hags, hats, haem, pman, sp_configd, hr, and hb**) have replaced the host responds (**hr**) and heartbeat (**hb**) interfaces.
8. If you use system partitioning on your SP system, you might want to rerun the system partitioning steps on your SP system. This allows the necessary Syspar config files to reflect all updates.
9. If your SP configuration supports HACWS, you must reconfigure the current HACWS configuration with the new IP address and host name changes. You will need to execute specific HACWS procedures to re-create the HACWS configuration. See "Working with HACWS Configurations" on page 528 for more information.
10. If the node supports HACMP, you might need to reconfigure the Alias addresses to match the updated host name or IP addresses.
11. If your SP system supports IBM Virtual Shared Disk configurations, you must execute some IBM Virtual Shared Disk commands on the SP nodes. You might need to stop and unconfigure the IBM Virtual Shared Disk, make IP address and host name changes, and then configure and start the IBM Virtual Shared Disk on the SP nodes.
12. If your SP system supports extension nodes (such as a switched IP router), you need to reconfigure the extension node's host name and IP address.
13. If your SP system supports a DCE configuration, you will need to update hostname and IP address on DCE servers and DCE clients.

14. If your SP system has an SP-controlled Netfinity server, you might need to update the Netfinity server host name and IP address.

Continue with “Updating the PSSP Node Interfaces.”

Updating the PSSP Node Interfaces

Make the following updates on all PSSP nodes prior to making updates to the control workstation.

On each PSSP node, perform each of the following steps that apply to your SP system. You can perform some of the steps easily by using SP Perspectives. You can automate others by implementing user shell scripts and by using the SP distributed shell (**dsh**) services from the control workstation to the target nodes.

1. Stop/Remove the RSCT services

Shutdown the daemons and source master objects that work with RSCT services. The RSCT services include Topology Services, Group Services, Event Manager, Problem Management, and Performance Monitor. Execute the **syspar_ctrl** command to stop and remove these resources.

```
/usr/lpp/ssp/bin/syspar_ctrl -c
```

2. Update the NIM files and configuration

If available, update the **/etc/niminfo** file to provide the new host name and IP address changes for the NIM client, NIM master, and NIM file references. If the SP node is currently a NIM master (boot/install server), remove the NIM ODM objects and NIM master configuration files from the SP node by issuing the **delnimmast** command:

```
/usr/lpp/ssp/bin/delnimmast -l node_number
```

3. Unconfigure Virtual Shared Disks

Determine if there are any virtual shared disk configurations available on the SP node. Using the IBM Virtual Shared Disk Perspective GUI, you can visually tell which nodes are configured for virtual shared disks and you can unconfigure them all at once. Otherwise, issue the **/usr/lpp/csd/bin/vsdata1st -n** command. If you receive an error message indicating “not found” or this SP node is not listed, continue with the next step. If your SP node is listed as having a virtual shared disk configured, then suspend, stop, and unconfigure the virtual shared disk for this SP node using the commands:

```
/usr/lpp/csd/bin/suspendvsd -a  
/usr/lpp/csd/bin/stopvsd -a  
/usr/lpp/csd/bin/ucfgvsd -a
```

4. Disable HACMP on nodes

Use instructions from the HACMP publications to disable the HACMP configuration. The nodes will not be able to communicate during this activity. After nodes are backed up and running again, reconfigure or re-enable the HACMP configuration.

5. Update DCE client information

Use the SMIT DCE client panels and dialog boxes to make proper changes to refer to DCE clients and DCE servers, IP addresses, and hostnames.

6. Update PSSP files for updates on each PSSP node

Update the following files to reflect the new host name and IP address for the SP node and servers:

- a. Update the **/etc/SDR_dest_info** file using the new control workstation and SDR IP addresses and host names.
- b. Update the **/etc/ssp/cw_name**, **server_hostname** and **server_name** files to reference the new boot server host name and IP address.
- c. Update the **/etc/ssp/reliable_hostname** file to reference the new SP node client host name and IP address.
- d. Make sure that the SP_NAME environment variable is updated or set to blank (export SP_NAME=).
- e. Issue **/usr/lpp/ssp/kerberos/bin/kdestroy** to remove any active Kerberos ticket-granting-tickets. You might need to update the **/etc/krb.conf** file if the authentication server is changed. IBM suggests renaming the **/etc/krb-srvtab** file on the SP node.

7. Update the PSSP node **en0** interface

Update the **en0** interface on each SP node to allow for proper communication paths between the SP node and the control workstation and you need to change the **en0** interface to the new IP address and host name. You must also supply the proper gateway and route information for the **en0** adapter. Make sure that all new IP addresses and host names are resolvable on the SP node. You do not need to modify any other adapter interfaces on the SP node. To accomplish this task, use the **mktcpip** command. You can do this using SMIT **mktcpip** or by issuing the **mktcpip** command on each SP node.

The following examples use the **mktcpip** command:

```
mktcpip -h new node hostname -a \  
new node IP addr -m subnet mask \  
-i en0 -n name server -d domain \  
-g gateway IP addr -t bnc
```

The following example changes the host name of node k22n06 to a new IP address 129.40.88.70 and host name k88n06 in domain ppd.pok.ibm.com:

```
/usr/sbin/mktcpip -h'k88n06.ppd.pok.ibm.com' -a'129.40.88.70' \  
-m'255.255.255.192' -i'en0' -n'129.40.70.1' -d'ppd.pok.ibm.com' \  
-g'129.40.88.126' -t'bnc'
```

After issuing **mktcpip**, the node loses the current network connection to the control workstation and the other SP nodes. Any further updates to the SP node need to be performed using the **tty0 console** using the **s1term** command. You might be able to kill the "telnet" or "rlogin" process from the control workstation or the home machine to regain the window.

That completes the updates required on nodes. **Do not reboot the nodes until you are instructed to do so.** Continue with "Execution on the Control Workstation" on page 524 to make the appropriate changes on the control workstation.

Execution on the Control Workstation

The following tasks need to be performed by the **admin** (root) user on the control workstation when changing the SP Ethernet IP address or host name.

You might want to backup the current **/spdata** file system, and create a **mksysb** backup before making any IP address and host name changes to the control workstation.

If you want to save the current SDR attributes, issue the command **/usr/lpp/ssp/bin/SDRArchive**. The support in PSSP allows the **admin** user to work with multiple system partition configurations.

If you are currently in an HACWS configuration, you should properly disable the backup control workstation and unconfigure HACWS prior to the changing any IP address and host names for the control workstation. Refer to “Working with HACWS Configurations” on page 528 for more information.

The following steps 1 through 10 are required steps when there are IP address and host name changes made to the control workstation. If there are only changes being made to the SP nodes, go to step 13 on page 526 in this section.

1. Stop PSSP daemons and remove current source master objects (**sdrd**, **hardmon**, and RSCT services) on the control workstation. The **syspar_ctrl -c** command will stop and remove all active resources that work with the RSCT services. The **stopsrc** command will stop the other PSSP resources. Remove the SDR source master object for each defined system partition (**SP_NAME**).

```
/usr/lpp/ssp/bin syspar_ctrl -G -c
/bin/stopsrc -g sdr
/bin/stopsrc -s hardmon
/bin/stopsrc -s supfilesrv
/usr/lpp/ssp/bin/sdr -spname SP_NAME rmsrc
```

2. Using **mktcpip** or the **chdev** command, specify IP address or host name changes required for the control workstation. This includes changes being made for any affected adapter interfaces. It is also important to make appropriate updates to the route tables, netmasks, and gateway servers.
3. If multiple system partitions exist, update the **/etc/rc.net** file to reference the new alias address used for each system partition. Issue **/etc/rc.net** or execute **ifconfig** command to configure the new alias IP addresses. You can issue the **netstat -ni** command to validate the new alias addresses. The old alias addresses used with system partitions will be removed during the next reboot of the control workstation.
4. Manually update the **/etc/SDR_dest_info** and **/spdata/sys1/spmon/hmacls** files to the new IP address and host name for the SDR and hardmon interfaces.

Manually update the SDR system partition map file **/spdata/sys1/sdr/system/classes/Syspar_map** to reflect the new IP address and their **SP_NAME** values with each system partition.

Move the system partition directories found at location **/spdata/sys1/sdr/partitions** from the old IP address to the new IP address for each system partition being modified.

```
/bin/mv old_IP_addr new_IP_addr
```

Manually update the SDR system partition file

/spdata/sys1/sdr/partitions/new_IP_addr/classes/Syspar to reflect the new IP address and their SP_NAME host name values for each defined system partition.

5. This step should only be performed for host name and domain changes for the control workstation. It is not required for changes made to control workstation IP addresses or to updates for SP nodes.

Issue the **setup_authent** script to create authentication services for the new host names being used. See the step “Initialize RS/6000 SP Authentication Services” in *PSSP: Installation and Migration Guide*.

Manually check that the authentication files **/etc/krb.conf /etc/krb.realms** reference the proper host names and domain. You can issue **/usr/lpp/ssp/kerberos/bin/ksrvutil list** to make sure that the **rcmd** and **hardmon** services reference the new host names in the **/etc/krb-srvtab** file. You might also need to recreate Kerberos principals for any users that were previously defined in the Kerberos database. You can use the **lskp**, **mkkp**, and **add_principal** commands to list, make, and add Kerberos principals.

6. Create the new source master resources for each SDR object, and then start the SDR and hardmon daemons.

```
sdr -spname SP_NAME mksrc new_IP_addr (for each Syspar)
startsrc -g sdr
startsrc -s hardmon
```

7. After the SDR daemon is properly activated on the SP system, manually issue **SDRChangeAttrValues** for the control workstation.

- To change the **hostname** attribute using the new host name for **SP_ports**, enter:

```
SDRChangeAttrValues SP_ports hostname=new_CWS_hostname
```

- To change the **MACN** attribute using the new host name for **Frame**, enter:

```
SDRChangeAttrValues Frame MACN=new_CWS_hostname
```

- To change the **control_workstation** attribute using the short host name for **SP**, enter:

```
SDRChangeAttrValues SP control_workstation=new_CWS_short_hostname
```

8. Remove the current Network Installation Manager (NIM) ODM database and configuration files on the control workstation. You need to issue:

- **/usr/lpp/ssp/bin/delnimmast -l 0**

It might be necessary to update the **/etc/niminfo** file to reference any changes to the new IP address and host names.

You need to remove entries from the NFS export list that reference the old IP address and host names. You can do this by updating the **/etc/exports** file, and then stopping and starting NFS subsystem.

9. Using CMI or the **spsitenv** command, specify any host name changes that might be referenced for NTP, printing, and user management. See the step “Enter Site Environment Information” in *PSSP: Installation and Migration Guide* (Optional).

10. Reboot the control workstation now. This establishes a clean system to reflect the IP address and host name changes. You should verify that all PSSP daemons are activated from **/etc/inittab** and the system resource master.

Usage Note

The remaining steps for the control workstation involve updating the SDR and SP system files for the SP node objects. Based on your SP configuration changes, some of the steps might be optional.

11. If your SP supports an SP-controlled Netfinity server, you might need to update its host name. You can use CMI or the **sppenode** command.

12. Update configuration files on the control workstation

If your SP system supports extension nodes, you might need to update an IP address or host name for an extension node. You can use CMI or commands **endefnode** and **endefadapter** to update extension node configuration information. See Chapter 17, “Managing Extension Nodes” on page 251.

Various AIX and SP files might need to be updated to reflect IP address or host name changes. You should look through the following files for required updates.

- a. Update any files that are involved with host name resolution. The files are **/etc/hosts**, **/etc/resolv.conf** (DNS) and **/var/yp/*** (NIS).
 - b. Update the **.rhosts** or **.klogin** files.
 - c. Update the **/etc/filesystems** and **/etc/jmd_config.SP_NAME** files for your SP configuration.
 - d. Make sure your **/tftpboot/script.cust** file and **/tftpboot/firstboot.cust** file are updated to reflect IP address and host name changes. Instead of hardcoding host names, you can reference the \$SERVER and \$CWS variables.
 - e. Update any DCE client and server files if supporting a DCE configuration.
13. Update the SDR node objects by using commands or the SMIT-based SP Configuration Management Interface (CMI). You can perform the following tasks for each system partition by exporting the SP_NAME variable (for example, export SP_NAME=SP_NAME).
 - a. Using CMI or the **spethernt** command, specify the new SP Ethernet IP address or host name changes required for the SP nodes. See the step “Enter Required Node Information” in *PSSP: Installation and Migration Guide*.
 - b. Using CMI or the **spadaptrs** command, reset the **css0** and other additional adapters that need to reference the new IP address or host names being changed. See the step “Configure Additional Adapters for Nodes” in *PSSP: Installation and Migration Guide*.
 - c. Using CMI or the **spostnam** command, reset the initial host name that you want to use in your system. See the step “Configure Initial Host Names for Nodes” in *PSSP: Installation and Migration Guide* (Optional).
 - d. Using CMI or the **spchvgobj** and **spbootins** commands, reset all boot/install servers to reference the new SP Ethernet IP address and host names. It is important that you set your SP node boot/install servers into

the proper configuration for your SP system. If you have system partitions, you should have already designated the proper PSSP boot server nodes.

```
spchvgobj -r rootvg -n boot_node -l node_list
```

You need to set the bootp response to **customize** for all the SP nodes.

```
spbootins -r customize -l node_list
```

You can issue the **splstdata -G -b** command to verify the correct boot information for the SP nodes.

The **spbootins** command runs the **setup_server** command which creates all the NIM-based files and resources required for installation. It also creates the authentication **rcmd** principals for the new SP node host names.

You might want to validate that the following files have the proper IP addresses and host names defined.

- **/etc/ntp.conf**
- **/ftpboot/host.config_info**
- **/ftpboot/host.install_info**
- **/ftpboot/host-new-srvtab**

14. If your SP system supports system partitions, reissue the system partitioning steps found in Chapter 15, “Managing System Partitions” on page 221. The system partitioning **spapply_config** command re-creates the proper PSSP daemons and resynchronizes the SDR objects to reflect any IP address and host name changes.

If you do not execute the system partitioning step, you will then need to create the source master objects and start the daemons for the RSCT services using the **syspar_ctrl -G -A** command.

```
/usr/lpp/ssp/bin/syspar_ctrl -G -A
```

15. For each PSSP boot server node, you should have already unconfigured NIM during PSSP node update activities. During the node customization, **setup_server** creates the proper install files for the nodes they are to customize. Since the SP nodes are in **customize** mode, most configuration files are updated to reflect the IP address and host name changes.

Attention: Some files will not be updated on the nodes during the customization. You can **rcp** these files from the control workstation to the SP nodes by including them in the **/ftpboot/script.cust** file or the **/ftpboot/firstboot.cust**. These can include the **/etc/resolv.conf**, **/rhosts**, and other SP customer-owned files.

16. Perform a **REBOOT** for each PSSP node. You can do this by using Perspectives or by issuing the **hmcnds** command.

Follow the proper install sequence by customizing each of the boot server nodes first. After the SP boot/install server node completes the installation setup, you can customize the remaining SP nodes.

Now verify that the customization was successful.

Working with HACWS Configurations

The following additional information for working with an HACWS configuration relates to working with steps 1 through 10 in “Execution on the Control Workstation” on page 524.

Note: Consider deferring the HACWS reconfiguration until your SP system has been properly updated and is stable on the new IP addresses and host names. Also, you need to be familiar with the HACWS information in the *PSSP: Installation and Migration Guide*.

When you are ready to reconfigure HACWS, do the following:

1. Stop daemons. Stop HACMP on both the primary and backup control workstations (this automatically stops all HACWS related daemons).
2. Use **mkctcpip**. Manually issue the appropriate **ifconfig** commands to configure the control workstation service addresses (backup CWS) on the primary control workstation. Make sure the host name and IP address are configured on the backup control workstation. If you are using an alias IP address, make the required changes to the **/etc/rc.backup_cw_alias** script.

Manually vary on the external volume group which contains the **/spdata** file system, and then mount the **/spdata** filesystem.

3. Update **rc.net**. Make updates to the **/spdata/sys1/hacws/rc.syspar_aliases** file to reference the new alias address used for system partition.
4. Update **SDR_dest_info**. No updates required.
5. Issue **setup_authent**. No updates required.
6. Create the new source master. Update the new SDR source master objects on the backup control workstation as well as the primary control workstation.

```
sdr -spname SP_NAME mksrc new_IP_addr (for each Syspar)
```

You can then start the SDR and hardmon daemons on the primary control workstation.

7. Update the SDR. Update the Frame object to include the host name for the **backup_MACN** using **SDRChangeAttrValues**.

```
SDRChangeAttrValues Frame backup_MACN=backup_CWS_hostname
```

You should also now execute the HACWS installation script **install_hacws** on the primary control workstation.

```
/usr/sbin/hacws/install_hacws -p primary_name -b backup_name  
-s
```

8. Remove the current NIM. No updates required.
9. Update the SDR node objects. No updates required.
10. Reboot.

To reconfigure the cluster topology, follow the standard HACMP procedures documented in *HACMP: Administration Guide*.

Verify that the HACWS configuration is properly setup by executing the **/usr/sbin/hacws/hacws_verify** command on the control workstation.

Also verify that you can properly **fail over** from the control workstation to the backup control workstation.

Execution on PSSP Nodes

To verify the changes, do the following steps on each node:

1. After reboot is complete and the SP nodes are customized, verify that the following have the correct IP address and host name specified:
 - The updated SP Ethernet address specified for the SP node
 - The updated default route and gateway are correct
 - The additional adapters are correct (**css0**, Ethernet, FDDI, and token ring)
 - The host name resolution files were updated correctly (**/etc/hosts**)
 - The **/etc/SDR_dest_info** file points to the control workstation and SDR Syspar IP address.
 - The **acct_master** host name is updated for the NFS export list directory **/var/adm/acct** (SP Accounting)
 - The files in the **/etc/ssp** directory *cw_name*, *server_hostname*, *server_name*, and *reliable_hostname* reference the correct IP address and host names.
 - The **/etc/krb.conf**, **/etc/krb.realms**, and **/etc/krb-srvtab** files have the correct host names defined
2. Verify that the following files on the SP nodes reflect the updated IP addresses and host name definitions:
 - **/etc/filesystems**
 - **/etc/ntp.conf**
 - **/etc/resolv.conf**
 - **/.rhosts**
 - **/.klogin**
 - **/etc/krb5.conf** (optional)

If any of the files are incorrect, make the proper updates for the correct IP address or host name.

3. Verify that the NIM resources were built on the PSSP boot server nodes by executing the **lsnim** command. If there were NIM problems, remove the current NIM database and configuration files.

```
/usr/lpp/ssp/bin/delnimmast -l node_number
```

Then issue **setup_server** on the PSSP boot server node.

4. Verify that the RSCT services have been activated on the PSSP nodes. Execute **lssrc -a** command to list all the active subsystems.

```
/bin/lssrc -a | grep rsct_subsystem
```

If you suspect that any of the expected RSCT services are inoperative or not available on the SP node, it is best to remove and then add the RSCT services using the **syspar_ctrl** command:

```
/usr/lpp/ssp/bin/syspar_ctrl -c  
/usr/lpp/ssp/bin/syspar_ctrl -A
```

5. Reconfigure and start the IBM Virtual Shared Disk for the SP nodes that support the IBM Virtual Shared Disk configuration. You can do this easily for all

nodes by using the IBM Virtual Shared Disk Perspective. If you are implementing a script, you might prefer to issue the following commands on each node:

```
/usr/lpp/csd/bin/cfgvsd -a  
/usr/lpp/csd/bin/startvsd -a
```

You can validate that the IP address and host names are correctly specified by issuing the following command:

```
/usr/lpp/csd/bin/vsdata1st -n
```

6. If there were modifications made to any SP node system files, reboot the PSSP nodes to reflect the IP address and host name changes. When the SP nodes are initialized, your SP system should be activated using the new IP addresses and host names.

Glossary of Terms and Abbreviations

This glossary includes terms and definitions from:

- The *IBM Dictionary of Computing*, New York: McGraw-Hill, 1994.
- The *American National Standard Dictionary for Information Systems*, ANSI X3.172-1990, copyright 1990 by the American National Standards Institute (ANSI). Copies can be purchased from the American National Standards Institute, 1430 Broadway, New York, New York 10018. Definitions are identified by the symbol (A) after the definition.
- The *ANSI/EIA Standard - 440A: Fiber Optic Terminology* copyright 1989 by the Electronics Industries Association (EIA). Copies can be purchased from the Electronic Industries Association, 2001 Pennsylvania Avenue N.W., Washington, D.C. 20006. Definitions are identified by the symbol (E) after the definition.
- The *Information Technology Vocabulary* developed by Subcommittee 1, Joint Technical Committee 1, of the International Organization for Standardization and the International Electrotechnical Commission (ISO/IEC JTC1/SC1). Definitions of published parts of this vocabulary are identified by the symbol (I) after the definition; definitions taken from draft international standards, committee drafts, and working papers being developed by ISO/IEC JTC1/SC1 are identified by the symbol (T) after the definition, indicating that final agreement has not yet been reached among the participating National Bodies of SC1.

The following cross-references are used in this glossary:

- Contrast with.** This refers to a term that has an opposed or substantively different meaning.
- See.** This refers the reader to multiple-word terms in which this term appears.
- See also.** This refers the reader to terms that have a related, but not synonymous, meaning.
- Synonym for.** This indicates that the term has the same meaning as a preferred term, which is defined in the glossary.

This section contains some of the terms that are commonly used in the SP publications.

IBM is grateful to the American National Standards Institute (ANSI) for permission to reprint its definitions from the American National Standard *Vocabulary for Information Processing* (Copyright 1970 by American National Standards Institute, Incorporated), which was prepared by Subcommittee X3K5 on Terminology and Glossary of the American National Standards

Committee X3. ANSI definitions are preceded by an asterisk (*).

Other definitions in this glossary are taken from *IBM Vocabulary for Data Processing, Telecommunications, and Office Systems* (SC20-1699) and *IBM DATABASE 2 Application Programming Guide for TSO Users* (SC26-4081).

A

adapter. An adapter is a mechanism for attaching parts. For example, an adapter could be a part that electrically or physically connects a device to a computer or to another device. In the SP system, network connectivity is supplied by various adapters, some optional, that can provide connection to I/O devices, networks of workstations, and mainframe networks. Ethernet, FDDI, token-ring, HiPPI, SCSI, FCS, and ATM are examples of adapters that can be used as part of an SP system.

address. A character or group of characters that identifies a register, a device, a particular part of storage, or some other data source or destination.

AFS. A distributed file system that provides authentication services as part of its file system creation.

AIX. Abbreviation for Advanced Interactive Executive, IBM's licensed version of the UNIX operating system. AIX is particularly suited to support technical computing applications, including high function graphics and floating point computations.

Amd. Berkeley Software Distribution automount daemon.

API. Application Programming Interface. A set of programming functions and routines that provide access between the Application layer of the OSI seven-layer model and applications that want to use the network. It is a software interface.

application. The use to which a data processing system is put; for example, a payroll application, an airline reservation application.

application data. The data that is produced using an application program.

ARP. Address Resolution Protocol.

ATM. Asynchronous Transfer Mode. (See *TURBOWAYS 100 ATM Adapter*.)

Authentication. The process of validating the identity of a user or server.

Authorization. The process of obtaining permission to perform specific actions.

B

batch processing. * (1) The processing of data or the accomplishment of jobs accumulated in advance in such a manner that each accumulation thus formed is processed or accomplished in the same run. * (2) The processing of data accumulating over a period of time. * (3) Loosely, the execution of computer programs serially. (4) Computer programs executed in the background.

BMCA. Block Multiplexer Channel Adapter. The block multiplexer channel connection allows the RS/6000 to communicate directly with a host System/370 or System/390; the host operating system views the system unit as a control unit.

BOS. The AIX Base Operating System.

C

call home function. The ability of a system to call the IBM support center and open a PMR to have a repair scheduled.

CDE. Common Desktop Environment. A graphical user interface for UNIX.

charge feature. An optional feature for either software or hardware for which there is a charge.

CLI. Command Line Interface.

client. * (1) A function that requests services from a server and makes them available to the user. * (2) A term used in an environment to identify a machine that uses the resources of the network.

Client Input/Output Sockets (CLIO/S). A software package that enables high-speed data and tape access between SP systems, AIX systems, and ES/9000 mainframes.

CLIO/S. Client Input/Output Sockets.

CMI. Centralized Management Interface provides a series of SMIT menus and dialogues used for defining and querying the SP system configuration.

connectionless. A communication process that takes place without first establishing a connection.

connectionless network. A network in which the sending logical node must have the address of the receiving logical node before information interchange can begin. The packet is routed through nodes in the network based on the destination address in the packet. The originating source does not receive an acknowledgment that the packet was received at the destination.

control workstation. A single point of control allowing the administrator or operator to monitor and manage the SP system using the IBM AIX Parallel System Support Programs.

css. Communication subsystem.

D

daemon. A process, not associated with a particular user, that performs system-wide functions such as administration and control of networks, execution of time-dependent activities, line printer spooling and so forth.

DASD. Direct Access Storage Device. Storage for input/output data.

DCE. Distributed Computing Environment.

DFS. distributed file system. A subset of the IBM Distributed Computing Environment.

DNS. Domain Name Service. A hierarchical name service which maps high level machine names to IP addresses.

E

Error Notification Object. An object in the SDR that is matched with an error log entry. When an error log entry occurs that matches the Notification Object, a user-specified action is taken.

ESCON. Enterprise Systems Connection. The ESCON channel connection allows the RS/6000 to communicate directly with a host System/390; the host operating system views the system unit as a control unit.

Ethernet. (1) Ethernet is the standard hardware for TCP/IP local area networks in the UNIX marketplace. It is a 10-megabit per second baseband type LAN that allows multiple stations to access the transmission medium at will without prior coordination, avoids contention by using carrier sense and deference, and resolves contention by collision detection (CSMA/CD). (2) A passive coaxial cable whose interconnections contain devices or components, or both, that are all active. It uses CSMA/CD technology to provide a best-effort delivery system.

Ethernet network. A baseband LAN with a bus topology in which messages are broadcast on a coaxial cabling using the carrier sense multiple access/collision detection (CSMA/CD) transmission method.

event. In Event Management, the notification that an expression evaluated to true. This evaluation occurs each time an instance of a resource variable is observed.

expect. Programmed dialogue with interactive programs.

expression. In Event Management, the relational expression between a resource variable and other elements (such as constants or the previous value of an instance of the variable) that, when true, generates an event. An example of an expression is $X < 10$ where X represents the resource variable `IBM.PSSP.aixos.PagSp.%total free` (the percentage of total free paging space). When the expression is true, that is, when the total free paging space is observed to be less than 10%, the Event Management subsystem generates an event to notify the appropriate application.

F

failover. Also called failover, the sequence of events when a primary or server machine fails and a secondary or backup machine assumes the primary workload. This is a disruptive failure with a short recovery time.

fall back. Also called fallback, the sequence of events when a primary or server machine takes back control of its workload from a secondary or backup machine.

FDDI. Fiber Distributed Data Interface.

Fiber Distributed Data Interface (FDDI). An American National Standards Institute (ANSI) standard for 100-megabit-per-second LAN using optical fiber cables. An FDDI local area network (LAN) can be up to 100 km (62 miles) and can include up to 500 system units. There can be up to 2 km (1.24 miles) between system units and concentrators.

file. * A set of related records treated as a unit, for example, in stock control, a file could consist of a set of invoices.

file name. A CMS file identifier in the form of 'filename filetype filemode' (like: TEXT DATA A).

file server. A centrally located computer that acts as a storehouse of data and applications for numerous users of a local area network.

File Transfer Protocol (FTP). The Internet protocol (and program) used to transfer files between hosts. It is

an application layer protocol in TCP/IP that uses TELNET and TCP protocols to transfer bulk-data files between machines or hosts.

foreign host. Any host on the network other than the local host.

FTP. File transfer protocol.

G

gateway. An intelligent electronic device interconnecting dissimilar networks and providing protocol conversion for network compatibility. A gateway provides transparent access to dissimilar networks for nodes on either network. It operates at the session presentation and application layers.

H

HACMP. High Availability Cluster Multi-Processing for AIX.

HACWS. High Availability Control Workstation function, based on HACMP, provides for a backup control workstation for the SP system.

| **HAL.** Hardware Abstraction Layer, a communication
| device interface that provides communication channels
| for processes.

Hashed Shared Disk (HSD). The data striping device for the IBM Virtual Shared Disk. The device driver lets application programs stripe data across physical disks in multiple IBM Virtual Shared Disks, thus reducing I/O bottlenecks.

help key. In the SP graphical interface, the key that gives you access to the SP graphical interface help facility.

High Availability Cluster Multi-Processing. An IBM facility to cluster nodes or components to provide high availability by eliminating single points of failure.

HiPPI. High Performance Parallel Interface. RS/6000 units can attach to a HiPPI network as defined by the ANSI specifications. The HiPPI channel supports burst rates of 100 Mbps over dual simplex cables; connections can be up to 25 km in length as defined by the standard and can be extended using third-party HiPPI switches and fiber optic extenders.

home directory. The directory associated with an individual user.

host. A computer connected to a network, and providing an access method to that network. A host provides end-user services.

I

instance vector. Obsolete term for resource identifier.

Intermediate Switch Board. Switches mounted in the Sp Switch expansion frame.

Internet. A specific inter-network consisting of large national backbone networks such as APARANET, MILNET, and NSFnet, and a myriad of regional and campus networks all over the world. The network uses the TCP/IP protocol suite.

Internet Protocol (IP). (1) A protocol that routes data through a network or interconnected networks. IP acts as an interface between the higher logical layers and the physical network. This protocol, however, does not provide error recovery, flow control, or guarantee the reliability of the physical network. IP is a connectionless protocol. (2) A protocol used to route data from its source to its destination in an Internet environment.

IP address. A 32-bit address assigned to devices or hosts in an IP internet that maps to a physical address. The IP address is composed of a network and host portion.

ISB. Intermediate Switch Board.

K

Kerberos. A service for authenticating users in a network environment.

kernel. The core portion of the UNIX operating system which controls the resources of the CPU and allocates them to the users. The kernel is memory-resident, is said to run in "kernel mode" and is protected from user tampering by the hardware.

L

LAN. (1) Acronym for Local Area Network, a data network located on the user's premises in which serial transmission is used for direct data communication among data stations. (2) Physical network technology that transfers data at a high speed over short distances. (3) A network in which a set of devices is connected to another for communication and that can be connected to a larger network.

local host. The computer to which a user's terminal is directly connected.

log database. A persistent storage location for the logged information.

log event. The recording of an event.

log event type. A particular kind of log event that has a hierarchy associated with it.

logging. The writing of information to persistent storage for subsequent analysis by humans or programs.

M

mask. To use a pattern of characters to control retention or elimination of portions of another pattern of characters.

menu. A display of a list of available functions for selection by the user.

Motif. The graphical user interface for OSF, incorporating the X Window System. Also called OSF/Motif.

MTBF. Mean time between failure. This is a measure of reliability.

MTTR. Mean time to repair. This is a measure of serviceability.

N

naive application. An application with no knowledge of a server that fails over to another server. Client to server retry methods are used to reconnect.

network. An interconnected group of nodes, lines, and terminals. A network provides the ability to transmit data to and receive data from other systems and users.

NFS. Network File System. NFS allows different systems (UNIX or non-UNIX), different architectures, or vendors connected to the same network, to access remote files in a LAN environment as though they were local files.

NIM. Network Installation Management is provided with AIX to install AIX on the nodes.

NIM client. An AIX system installed and managed by a NIM master. NIM supports three types of clients:

- Standalone
- Diskless
- Dataless

NIM master. An AIX system that can install one or more NIM clients. An AIX system must be defined as a NIM master before defining any NIM clients on that system. A NIM master manages the configuration database containing the information for the NIM clients.

NIM object. A representation of information about the NIM environment. NIM stores this information as objects in the NIM database. The types of objects are:

- Network
- Machine
- Resource

NIS. Network Information System.

node. In a network, the point where one or more functional units interconnect transmission lines. A computer location defined in a network. The SP system can house several different types of nodes for both serial and parallel processing. These node types can include thin nodes, wide nodes, 604 high nodes, as well as other types of nodes both internal and external to the SP frame.

Node Switch Board. Switches mounted on frames that contain nodes.

NSB. Node Switch Board.

NTP. Network Time Protocol.

O

ODM. Object Data Manager. In AIX, a hierarchical object-oriented database for configuration data.

P

parallel environment. A system environment where message passing or SP resource manager services are used by the application.

Parallel Environment. A licensed IBM program used for message passing applications on the SP or RS/6000 platforms.

parallel processing. A multiprocessor architecture which allows processes to be allocated to tightly coupled multiple processors in a cooperative processing environment, allowing concurrent execution of tasks.

parameter. * (1) A variable that is given a constant value for a specified application and that may denote the application. * (2) An item in a menu for which the operator specifies a value or for which the system provides a value when the menu is interpreted. * (3) A name in a procedure that is used to refer to an argument that is passed to the procedure. * (4) A particular piece of information that a system or application program needs to process a request.

partition. See system partition.

Perl. Practical Extraction and Report Language.

perspective. The primary window for each SP Perspectives application, so called because it provides a unique view of an SP system.

pipe. A UNIX utility allowing the output of one command to be the input of another. Represented by the | symbol. It is also referred to as filtering output.

PMR. Problem Management Report.

POE. Formerly Parallel Operating Environment, now Parallel Environment for AIX.

port. (1) An end point for communication between devices, generally referring to physical connection. (2) A 16-bit number identifying a particular TCP or UDP resource within a given TCP/IP node.

predicate. Obsolete term for expression.

Primary node or machine. (1) A device that runs a workload and has a standby device ready to assume the primary workload if that primary node fails or is taken out of service. (2) A node on the SP Switch that initializes, provides diagnosis and recovery services, and performs other operations to the switch network. (3) In IBM Virtual Shared Disk function, when physical disks are connected to two nodes (twin-tailed), one node is designated as the primary node for each disk and the other is designated the secondary, or backup, node. The primary node is the server node for IBM Virtual Shared Disks defined on the physical disks under normal conditions. The secondary node can become the server node for the disks if the primary node is unavailable (off-line or down).

Problem Management Report. The number in the IBM support mechanism that represents a service incident with a customer.

process. * (1) A unique, finite course of events defined by its purpose or by its effect, achieved under defined conditions. * (2) Any operation or combination of operations on data. * (3) A function being performed or waiting to be performed. * (4) A program in operation. For example, a daemon is a system process that is always running on the system.

protocol. A set of semantic and syntactic rules that defines the behavior of functional units in achieving communication.

R

RAID. Redundant array of independent disks.

rearm expression. In Event Management, an expression used to generate an event that alternates with an original event expression in the following way: the event expression is used until it is true, then the

rearm expression is used until it is true, then the event expression is used, and so on. The rearm expression is commonly the inverse of the event expression (for example, a resource variable is on or off). It can also be used with the event expression to define an upper and lower boundary for a condition of interest.

rearm predicate. Obsolete term for rearm expression.

remote host. See *foreign host*.

resource. In Event Management, an entity in the system that provides a set of services. Examples of resources include hardware entities such as processors, disk drives, memory, and adapters, and software entities such as database applications, processes, and file systems. Each resource in the system has one or more attributes that define the state of the resource.

resource identifier. In Event Management, a set of elements, where each element is a name/value pair of the form name=value, whose values uniquely identify the copy of the resource (and by extension, the copy of the resource variable) in the system.

resource monitor. A program that supplies information about resources in the system. It can be a command, a daemon, or part of an application or subsystem that manages any type of system resource.

resource variable. In Event Management, the representation of an attribute of a resource. An example of a resource variable is IBM.AIX.PagSp.%totalfree, which represents the percentage of total free paging space. IBM.AIX.PagSp specifies the resource name and %totalfree specifies the resource attribute.

RISC. Reduced Instruction Set Computing (RISC), the technology for today's high performance personal computers and workstations, was invented in 1975. Uses a small simplified set of frequently used instructions for rapid execution.

rlogin (remote LOGIN). A service offered by Berkeley UNIX systems that allows authorized users of one machine to connect to other UNIX systems across a network and interact as if their terminals were connected directly. The rlogin software passes information about the user's environment (for example, terminal type) to the remote machine.

RPC. Acronym for Remote Procedure Call, a facility that a client uses to have a server execute a procedure call. This facility is composed of a library of procedures plus an XDR.

RSH. A variant of RLOGIN command that invokes a command interpreter on a remote UNIX machine and passes the command line arguments to the command interpreter, skipping the LOGIN step completely. See also *rlogin*.

S

SCSI. Small Computer System Interface.

Secondary node. In IBM Virtual Shared Disk function, when physical disks are connected to two nodes (twin-tailed), one node is designated as the primary node for each disk and the other is designated as the secondary, or backup, node. The secondary node acts as the server node for the IBM Virtual Shared disks defined on the physical disks if the primary node is unavailable (off-line or down).

server. (1) A function that provides services for users. A machine may run client and server processes at the same time. (2) A machine that provides resources to the network. It provides a network service, such as disk storage and file transfer, or a program that uses such a service. (3) A device, program, or code module on a network dedicated to providing a specific service to a network. (4) On a LAN, a data station that provides facilities to other data stations. Examples are file server, print server, and mail server.

shell. The shell is the primary user interface for the UNIX operating system. It serves as command language interpreter, programming language, and allows foreground and background processing. There are three different implementations of the shell concept: Bourne, C and Korn.

Small Computer System Interface (SCSI). An input and output bus that provides a standard interface for the attachment of various direct access storage devices (DASD) and tape drives to the RS/6000.

Small Computer Systems Interface Adapter (SCSI Adapter). An adapter that supports the attachment of various direct-access storage devices (DASD) and tape drives to the RS/6000.

SMIT. The System Management Interface Toolkit is a set of menu driven utilities for AIX that provides functions such as transaction login, shell script creation, automatic updates of object database, and so forth.

SNMP. Simple Network Management Protocol. (1) An IP network management protocol that is used to monitor attached networks and routers. (2) A TCP/IP-based protocol for exchanging network management information and outlining the structure for communications among network devices.

socket. (1) An abstraction used by Berkeley UNIX that allows an application to access TCP/IP protocol functions. (2) An IP address and port number pairing. (3) In TCP/IP, the Internet address of the host computer on which the application runs, and the port number it uses. A TCP/IP application is identified by its socket.

standby node or machine. A device that waits for a failure of a primary node in order to assume the identity of the primary node. The standby machine then runs the primary's workload until the primary is back in service.

subnet. Shortened form of subnetwork.

subnet mask. A bit template that identifies to the TCP/IP protocol code the bits of the host address that are to be used for routing for specific subnetworks.

subnetwork. Any group of nodes that have a set of common characteristics, such as the same network ID.

subsystem. A software component that is not usually associated with a user command. It is usually a daemon process. A subsystem will perform work or provide services on behalf of a user request or operating system request.

SUP. Software Update Protocol.

switch capsule. A group of SP frames consisting of a switched frame and its companion non-switched frames.

Sysctl. Secure System Command Execution Tool. An authenticated client/server system for running commands remotely and in parallel.

syslog. A BSD logging system used to collect and manage other subsystem's logging data.

System Administrator. The user who is responsible for setting up, modifying, and maintaining the SP system.

system partition. A group of nonoverlapping nodes on a switch chip boundary that act as a logical SP system.

T

tar. Tape ARchive, is a standard UNIX data archive utility for storing data on tape media.

TaskGuides. SP TaskGuides are a form of advanced online assistance designed to walk you through complex or infrequently performed tasks. Each TaskGuide does not simply list the required steps. It actually performs the steps for you, automating the steps to the highest degree possible and prompting you for input only when absolutely necessary. You might recognize them as *wizards*.

Tcl. Tool Command Language.

TclIX. Tool Command Language Extended.

TCP. Acronym for Transmission Control Protocol, a stream communication protocol that includes error recovery and flow control.

TCP/IP. Acronym for Transmission Control Protocol/Internet Protocol, a suite of protocols designed to allow communication between networks regardless of the technologies implemented in each network. TCP provides a reliable host-to-host protocol between hosts in packet-switched communications networks and in interconnected systems of such networks. It assumes that the underlying protocol is the Internet Protocol.

Telnet. Terminal Emulation Protocol, a TCP/IP application protocol that allows interactive access to foreign hosts.

Tk. Tcl-based Tool Kit for X Windows.

TMPCP. Tape Management Program Control Point.

token-ring. (1) Network technology that controls media access by passing a token (special packet or frame) between media-attached machines. (2) A network with a ring topology that passes tokens from one attaching device (node) to another. (3) The IBM Token-Ring LAN connection allows the RS/6000 system unit to participate in a LAN adhering to the IEEE 802.5 Token-Passing Ring standard or the ECMA standard 89 for Token-Ring, baseband LANs.

transaction. An exchange between the user and the system. Each activity the system performs for the user is considered a transaction.

transceiver (transmitter-receiver). A physical device that connects a host interface to a local area network, such as Ethernet. Ethernet transceivers contain electronics that apply signals to the cable and sense collisions.

transfer. To send data from one place and to receive the data at another place. Synonymous with move.

transmission. * The sending of data from one place for reception elsewhere.

TURBOWAYS 100 ATM Adapter. An IBM high-performance, high-function intelligent adapter that provides dedicated 100 Mbps ATM (asynchronous transfer mode) connection for high-performance servers and workstations.

U

UDP. User Datagram Protocol.

UNIX operating system. An operating system developed by Bell Laboratories that features multiprogramming in a multiuser environment. The UNIX operating system was originally developed for use on minicomputers, but has been adapted for mainframes and microcomputers. **Note:** The AIX operating system is IBM's implementation of the UNIX operating system.

user. Anyone who requires the services of a computing system.

User Datagram Protocol (UDP). (1) In TCP/IP, a packet-level protocol built directly on the Internet Protocol layer. UDP is used for application-to-application programs between TCP/IP host systems. (2) A transport protocol in the Internet suite of protocols that provides unreliable, connectionless datagram service. (3) The Internet Protocol that enables an application programmer on one machine or process to send a datagram to an application program on another machine or process.

user ID. A nonnegative integer, contained in an object of type *uid_t*, that is used to uniquely identify a system user.

V

Virtual Shared Disk, IBM. The function that allows application programs executing at different nodes of a system partition to access a raw logical volume as if it were local at each of the nodes. In actuality, the logical volume is local at only one of the nodes (the server node).

W

workstation. * (1) A configuration of input/output equipment at which an operator works. * (2) A terminal or microcomputer, usually one that is connected to a mainframe or to a network, at which a user can perform applications.

X

X Window System. A graphical user interface product.

Bibliography

This bibliography helps you find product documentation related to the RS/6000 SP hardware and software products.

You can find most of the IBM product information for RS/6000 SP products on the World Wide Web. Formats for both viewing and downloading are available.

PSSP documentation is shipped with the PSSP product in a variety of formats and can be installed on your system. The man pages for public code that PSSP includes are also available online.

You can order hard copies of the product documentation from IBM. This bibliography lists the titles that are available and their order numbers.

Finally, this bibliography contains a list of non-IBM publications that discuss parallel computing and other topics related to the RS/6000 SP.

Finding Documentation on the World Wide Web

Most of the RS/6000 SP hardware and software books are available from the IBM RS/6000 Web site at:

<http://www.rs6000.ibm.com>

You can view a book or download a Portable Document Format (PDF) version of it. At the time this manual was published, the Web address of the "RS/6000 SP Product Documentation Library" page was:

http://www.rs6000.ibm.com/resource/aix_resource/sp_books

However, the structure of the RS/6000 Web site can change over time.

Accessing PSSP Documentation Online

On the same medium as the PSSP product code, IBM ships PSSP man pages, HTML files, and PDF files. In order to use these publications, you must first install the **ssp.docs** file set.

To view the PSSP HTML publications, you need access to an HTML document browser such as Netscape. The HTML files and an index that links to them are installed in the **/usr/lpp/ssp/html** directory. Once installed, you can also view the HTML files from the RS/6000 SP Resource Center.

If you have installed the SP Resource Center on your SP system, you can access it by entering the **/usr/lpp/ssp/bin/resource_center** command. If you have the SP Resource Center on CD-ROM, see the **readme.txt** file for information about how to run it.

To view the PSSP PDF publications, you need access to the Adobe Acrobat Reader 3.0.1. The Acrobat Reader is shipped with the AIX Version 4.3 Bonus Pack.

To successfully print a large PDF file (approximately 300 or more pages) from the Adobe Acrobat reader, you may need to select the "Download Fonts Once" button on the Print window.

Manual Pages for Public Code

The following manual pages for public code are available in this product:

SUP	/usr/lpp/ssp/man/man1/sup.1
NTP	/usr/lpp/ssp/man/man8/xntpd.8 /usr/lpp/ssp/man/man8/xntpd.8
Perl (Version 4.036)	/usr/lpp/ssp/perl/man/perl.man /usr/lpp/ssp/perl/man/h2ph.man /usr/lpp/ssp/perl/man/s2p.man /usr/lpp/ssp/perl/man/a2p.man
Perl (Version 5.003)	Man pages are in the /usr/lpp/ssp/perl5/man/man1 directory

Manual pages and other documentation for **Tcl**, **TclX**, **Tk**, and **expect** can be found in the compressed **tar** files located in the **/usr/lpp/ssp/public** directory.

RS/6000 SP Planning Publications

This section lists the IBM product documentation for planning for the IBM RS/6000 SP hardware and software.

IBM RS/6000 SP:

- *Planning, Volume 1, Hardware and Physical Environment, GA22-7280*
- *Planning, Volume 2, Control Workstation and Software Environment, GA22-7281*

RS/6000 SP Hardware Publications

This section lists the IBM product documentation for the IBM RS/6000 SP hardware.

IBM RS/6000 SP:

- *Planning, Volume 1, Hardware and Physical Environment, GA22-7280*
- *Planning, Volume 2, Control Workstation and Software Environment, GA22-7281*
- *Maintenance Information, Volume 1, Installation and Relocation, GA22-7375*
- *Maintenance Information, Volume 2, Maintenance Analysis Procedures, GA22-7376*
- *Maintenance Information, Volume 3, Locations and Service Procedures, GA22-7377*
- *Maintenance Information, Volume 4, Parts Catalog, GA22-7378*

RS/6000 SP Switch Router Publications

The RS/6000 SP Switch Router is based on the Ascend GRF switched IP router product from Ascend Communications, Inc.. You can order the SP Switch Router as the IBM 9077.

The following publications are shipped with the SP Switch Router. You can also order these publications from IBM using the order numbers shown.

- *Ascend GRF Getting Started, GA22-7368*
- *Ascend GRF Configuration Guide, GA22-7366*
- *Ascend GRF Reference Guide, GA22-7367*
- *IBM SP Switch Router Adapter Guide, GA22-7310.*

RS/6000 SP Software Publications

This section lists the IBM product documentation for software products related to the IBM RS/6000 SP. These products include:

- IBM Parallel System Support Programs for AIX (PSSP)
- IBM LoadLeveler for AIX (LoadLeveler)
- IBM Parallel Environment for AIX (Parallel Environment)
- IBM General Parallel File System for AIX (GPFS)
- IBM Engineering and Scientific Subroutine Library (ESSL) for AIX
- IBM Parallel ESSL for AIX
- IBM High Availability Cluster Multi-Processing for AIX (HACMP)
- IBM Client Input Output/Sockets (CLIO/S)
- IBM Network Tape Access and Control System for AIX (NetTAPE)

PSSP Publications

IBM RS/6000 SP:

- *Planning, Volume 2, Control Workstation and Software Environment, GA22-7281*

PSSP:

- *Installation and Migration Guide, GA22-7347*
- *Administration Guide, SA22-7348*
- *Managing Shared Disks, SA22-7349*
- *Performance Monitoring Guide and Reference, SA22-7353*
- *Diagnosis Guide, GA22-7350*
- *Command and Technical Reference, SA22-7351*
- *Messages Reference, GA22-7352*

RS/6000 Cluster Technology (RSCT):

- *Event Management Programming Guide and Reference, SA22-7354*
- *Group Services Programming Guide and Reference, SA22-7355*

As an alternative to ordering the individual books, you can use SBOF-8587 to order the PSSP software library.

LoadLeveler Publications

LoadLeveler:

- *Using and Administering, SA22-7311*
- *Diagnosis and Messages Guide, GA22-7277*

GPFS Publications

GPFS:

- *Installation and Administration Guide, SA22-7278*

Parallel Environment Publications

Parallel Environment:

- *Installation Guide, GC28-1981*

- *Hitchhiker's Guide*, GC23-3895
- *Operation and Use, Volume 1*, SC28-1979
- *Operation and Use, Volume 2*, SC28-1980
- *MPI Programming and Subroutine Reference*, GC23-3894
- *MPL Programming and Subroutine Reference*, GC23-3893
- *Messages*, GC28-1982

As an alternative to ordering the individual books, you can use SBOF-8588 to order the PE library.

Parallel ESSL and ESSL Publications

- *ESSL Products: General Information*, GC23-0529
- *Parallel ESSL: Guide and Reference*, SA22-7273
- *ESSL: Guide and Reference*, SA22-7272

HACMP Publications

HACMP:

- *Concepts and Facilities*, SC23-4276
- *Planning Guide*, SC23-4277
- *Installation Guide*, SC23-4278
- *Administration Guide*, SC23-4279
- *Troubleshooting Guide*, SC23-4280
- *Programming Locking Applications*, SC23-4281
- *Programming Client Applications*, SC23-4282
- *Master Index and Glossary*, SC23-4285
- *HANFS for AIX Installation and Administration Guide*, SC23-4283
- *Enhanced Scalability Installation and Administration Guide*, SC23-4284

CLIO/S Publications

CLIO/S:

- *General Information*, GC23-3879
- *User's Guide and Reference*, GC28-1676

NetTAPE Publications

NetTAPE:

- *General Information*, GC23-3990
- *User's Guide and Reference*, available from your IBM representative

AIX and Related Product Publications

For the latest information on AIX and related products, including RS/6000 hardware products, see *AIX and Related Products Documentation Overview*, SC23-2456. You can order a hard copy of the book from IBM. You can also view it online from the "AIX Online Publications and Books" page of the RS/6000 Web site at:

http://www.rs6000.ibm.com/resource/aix_resource/Pubs

Red Books

IBM's International Technical Support Organization (ITSO) has published a number of redbooks related to the RS/6000 SP. For a current list, see the ITSO Web site at:

<http://www.redbooks.ibm.com>

Non-IBM Publications

Here are some non-IBM publications that you may find helpful.

- Almasi, G., Gottlieb, A., *Highly Parallel Computing*, Benjamin-Cummings Publishing Company, Inc., 1989.
- Foster, I., *Designing and Building Parallel Programs*, Addison-Wesley, 1995.
- Gropp, W., Lusk, E., Skjellum, A., *Using MPI*, The MIT Press, 1994.
- Message Passing Interface Forum, *MPI: A Message-Passing Interface Standard, Version 1.1*, University of Tennessee, Knoxville, Tennessee, June 6, 1995.
- Message Passing Interface Forum, *MPI-2: Extensions to the Message-Passing Interface, Version 2.0*, University of Tennessee, Knoxville, Tennessee, July 18, 1997.
- Ousterhout, John K., *Tcl and the Tk Toolkit*, Addison-Wesley, Reading, MA, 1994, ISBN 0-201-63337-X.
- Pfister, Gregory, F., *In Search of Clusters*, Prentice Hall, 1998.

Index

Special Characters

- `/etc/.profile` file 53
- `/etc/cshutSeq`
 - definition of 18
 - example 27
 - if empty 18
 - if none exists 18
 - syntax of 25
- `/etc/cstartSeq`
 - definition of 18
 - example 27
 - if empty 18
 - if none exists 18
 - syntax of 25
- `/etc/environment` file 56, 57
- `/etc/ha/cfg` directory
 - EMCDB version string files 348
 - run-time EMCDB files 348
- `/etc/inittab` file
 - use by Event Management 350
 - use by Group Services 331
- `/etc/krb.conf` file 148
- `/etc/krb.realms` file 148
- `/etc/ntp.conf` file 89
- `/etc/SDR_dest_info` file 480
- `/etc/services` file 147
 - use by Event Management 342
 - use by Group Services 328
- `/etc/subsysSeq`
 - definition of 18
 - example 28
 - if none exists or if empty 18
 - syntax of 25
- `/etc/syslog.conf` file 283
- `/spdata/sys1/ha/cfg` directory
 - staging area for EMCDB files 346
- `/usr/local/fonts` file 60
- `/usr/lpp/ssp/bin/newpass.log` file 76
- `/usr/lpp/ssp/config/cron_template` 122
- `/usr/lpp/ssp/install/config/haemloadlist` file
 - listing resource variable information 473
- `/var` file system
 - and Event Management tracing 351
 - and Group Services tracing 332
- `/var/adm/acct/nite` directory 124
- `/var/adm/cacct/` directory 125
- `/var/adm/cacct/fiscal` directory 125
- `/var/adm/cacct/sum` directory 125
- `/var/adm/SPlogs/cs/cshut.MMDDhhmmss.pid` 24
- `/var/adm/SPlogs/cs/cstart.MMDDhhmmss.pid` 24

- `/var/ha/lck` directory
 - Group Services lock files 329
- `/var/ha/lck/haem` directory
 - Event Management lock files 346
- `/var/ha/log` directory
 - Event Management log files 347
 - Group Services log files 329
- `/var/ha/run` directory
 - Event Manager daemon working files 347
 - Group Services daemon working files 329
 - registration cache 347
 - resource monitor working directories 347
- `/var/ha/soc` directory
 - Event Management socket files 346
 - Group Services socket files 329
- `/var/sysman/file.collections` file 43
- `/var/sysman/sup/.resident` file 43
- `/var/sysman/sup/lists` file 43
- `/var/sysman/sup/node.root/prefix` file 57
- `/var/sysman/sup/user.admin/list` file 57
- `/var/sysman/sup/user.admin/prefix` file 57
- `/var/sysman/sup.resident` file 61
- `.profile` 65
- `.resident` file 61

A

- access to the configuration data 479
- accounting 11
 - accounting 119
 - additional setup tasks 121
 - by node or by class 115
 - differences between SP and standard AIX 116
 - entering node accounting information 119
 - entering system information 118
 - files 122
 - for exclusive use of node jobs 116
 - HACWS and 306
 - installation 121
 - report and summary files 123
 - setting up SP 117
 - system partitioning 118
 - template for crontabs 122
- Active Message
 - C example 412
 - definition of function 401
 - Fortran example 416
 - infrastructure 396
 - writing handlers 396
 - writing handlers example 397
- adapter failures 301

- Adapter Membership
 - resource monitor 341
- adapters, extension node
 - adding 253
 - changing 255
 - listing 259
 - reconfiguring 257
 - removing 256
- adapters, network connectivity 8
- adding and deleting files in a file collection 56
- adding extension nodes 253
- adding subsystems
 - Event Management (haemctrl) 350
 - Group Services (hagsctrl) 331
- adding user accounts 76
- AFS
 - adding principals to 160
 - administering AFS authentication services 174
 - changing passwords in 172
- AFS authentication services 174
- afscient.cust 443
- AIX 9
- AIX error log
 - Group Services entries 333
 - use by Event Manager daemon 347
- AIX error log facility
 - configuring
 - SMIT, from 385
 - creating SNMP traps from 376
 - generating reports 385
 - managing 384
 - SMIT, from 384
 - trimming logs 385
 - SMIT, from 385
- AIX operating system statistics
 - obtained from SPMI 348
- aixos 341
- alternate root volume group
 - installing 439
 - removing 440
 - switching to 440
 - using 439
- application windows of SP Perspectives 269
- applications 39
- authentication
 - adding IBM RS/6000 workstations to your realm 146
 - adding principals 158
 - administering SP authentication services
 - adding principals 158
 - changing admin instance password 161
 - changing passwords 161
 - setting passwords 158
 - administration roles 154
 - AFS
 - adding principals 160
 - administering 174
 - authentication (*continued*)
 - AFS (*continued*)
 - changing passwords in 172
 - AFS services 174
 - authentication client system 150
 - authorization as an authentication database
 - administrator 160
 - backing up database 154
 - changing admin instance password 161
 - changing passwords 161
 - changing your password 172
 - configuration files 147
 - credentials 172
 - destroying tickets 171
 - directory paths for services 151
 - HACWS and 307
 - identifying yourself 169
 - implementations 146
 - listing tickets 171
 - managing information for authenticated services
 - changing service keys 166
 - managing the database 154
 - primary server 149
 - protecting your database 153
 - realms 137
 - secondary servers 146
 - service keys 164
 - services on the SP 131
 - services, using 168
 - setting passwords 158
 - setting up a secondary server 150
 - setting up an authentication client system 150
 - terminology 133
 - understanding 131
 - using the SP authentication services
 - changing your password 172
 - destroying tickets 171
 - identifying yourself 169
 - listing tickets 171
- authentication, SDR 479
- authority levels for SP System Monitor
 - control authority 282
 - monitor authority 282
- authorization, SDR 479
- automatic unfence 214
- automount daemon
 - restarting 102
 - stopping 102
- automounter 98
 - control of other file systems 99
 - customizing function
 - files 103
 - diagnostics 98
 - file collections 98
 - HACWS and 306
 - managing 93

- automounter (*continued*)
 - map file
 - creating 100
 - map files 95
 - adding to file collections 101
 - distributing to boot/install servers and nodes 102
 - map for users' directories 99
 - master map file
 - updating 101
 - master map files 97
 - requesting support 94
 - understanding 93
- availability 11

B

- backing up authentication database 154
- boot/install server
 - distributing automounter map files to 102
 - sequencing of startup and shutdown 17
- bootptab 446
- BSD syslog facility 389
 - generating reports 388
 - managing 388
- building a file collection 58
 - installing 61

C

- cache, registration
 - Event Management 347
- cfgauto.cust script 104
- change management 11
- changing extension nodes 255
- changing user accounts 77
- changing your password 78
- characteristics of SP system 3
- checkauto.cust script 107
- chkp 162
- cleaning subsystems
 - Event Management (haemctrl) 351
 - Group Services (hagsctrl) 331
- client
 - type of resource monitor 340
- client communication
 - with Event Management subsystem 339
 - with Group Services subsystem 327
- client, Group Services
 - definition 325
- cluster
 - and Event Management 339
- cluster workstation
 - and Event Management 339
- cmonacct 116
- collecting system data for IBM service 391

- command
 - type of resource monitor 340
- command files, nrunacct 123
- command line, from the 386
- command line, using the 227, 230, 232, 235, 237, 238, 241, 242
- commands 35, 37, 39
 - cshutdown 21
 - cstartup command 18
 - dsh 31
 - dspcat
 - displaying event management descriptions 476
 - dspmmsg
 - displaying event management descriptions 476
 - Eprimary 206
 - Estart 206
 - hostlist 33
 - p_cat 35
 - pcp 35
 - pdf 37
 - pexec 35
 - pfind 36
 - pfps 37
 - pls 36
 - pmv 36
 - ppred 37
 - pps 36
 - prm 36
 - SDRGetObjects
 - listing resource variable information 473
 - output example 474
 - spevent
 - listing resource variable information 476
 - output example 476
- communication subsystem support 11
- communication, client
 - with Event Management subsystem 339
 - with Group Services subsystem 327
- Communications Low-level Application Programming Interface (LAPI) 395
- communications, Event Management
 - between Event Manager daemons 342
 - local EM clients 342
 - remote EM clients 342
 - resource monitors 342
- communications, Group Services
 - between Group Services daemons 328
 - local GS clients 328
- configuration data
 - activating 358
 - changing in the SDR 358
 - changing resource variable instance limits 359
 - for third-party resource monitors 344, 345, 357
- configuration data (SDR)
 - component of Event Management 343

- configuration database
 - component of Event Management 344
- configuration load utility
 - component of Event Management 345
- configuration management 10
- configuration utility
 - component of Event Management 344
- configuring the system monitor 282
- connectivity adapters, network 8
- connectivity, switch 214
- control script
 - component of Event Management 345
 - component of Group Services 328
- control workstation 7
 - booting for HACWS 300
 - failovers 302
 - high availability 295
- controlling login 81
- core file
 - Event Manager daemon 347
 - Group Services daemon 329
- cprdaily 116
- create class command 185
- create proc command 185
- create var command 184
- creating custom layouts 228
- credentials
 - deleting 171
 - displaying 171
 - managing 172
 - obtaining 169
- crontabs file 46
- crontabs, accounting template for 122
- cshutdown command
 - log file 24
 - using 21
- CSS subsystem
 - resource monitor 341
- css.summlog daemon 218
- cssMembership group 341
 - use by Event Management 349
- cstartup command
 - log file 24
 - safety warning 18
- customizing
 - automounter function 103
 - system partitions 233

D

- daemon
 - type of resource monitor 340
- daemons 441
- data files 122
- default configuration data 345
 - component of Event Management 346
- default configuration data (*continued*)
 - loading into the SDR with haemloadcfg 345
 - location for Event Management 346
- default expression
 - changing 344
- default system partition 223
- deleting subsystems
 - Event Management (haemctrl) 351
 - Group Services (hagsctrl) 331
- deleting user accounts 78
- dependent nodes, switch considerations 207
- descriptions, displaying event management
 - using the dspcat command 476
 - using the dspmsg command 476
- determining switch connectivity 214
- df command 483
- diagnostics
 - automounter 98
 - dialog boxes of SP Perspectives 271
 - disks for HACWS 297
- displaying event management descriptions
 - using the dspcat command 476
 - using the dspmsg command 476
- distributed shell 31
- domain, operational
 - for Event Management 339
 - for Group Services 327
- dsh 31
- dsh based 34
- dsh command 31
- dspcat command
 - displaying event management descriptions 476
- dspmsg command
 - displaying event management descriptions 476

E

- EM client (Event Management client)
 - definition 337
 - registration request cache 347
 - restrictions on number 354
- EM_Instance_ID SDR class
 - use by Event Management 343
- EM_Resource_Class SDR class
 - key attributes for loading 345
 - use by Event Management 343
- EM_Resource_Monitor SDR class
 - key attributes for loading 345
 - use by Event Management 343
- EM_Resource_Variable SDR class
 - key attributes for loading 345
 - use by Event Management 343
- EM_Structured_Byte_String SDR class
 - key attributes for loading 345
 - use by Event Management 343

- EMAPI (Event Management Application Programming Interface)
 - component of Event Management 339
- EMAPI libraries
 - location 339
- embedded resource monitor
 - type of resource monitor 340
- EMCDB (Event Management Configuration Database)
 - and Event Manager daemon initialization 353
 - compiling with haemcfg 344
 - placing a new version in production 354
 - run-time directory 348
 - run-time file 345
 - staging directory 346
 - staging file 344
 - version string 344
 - version string and joining peer group 353
 - version string and peer group state 349
 - version string directory 348
- Emonitor 216
- EMResource_ID SDR class
 - key attributes for loading 345
- en_pend method flow 386
- enMembership group 341
 - use by Event Management 349
- Eprimary command 206
- Equiesce 219
- error log management 383
 - archiving error logs 390
 - collecting system data for IBM service 391
 - creating archives 390
 - creating service collections 391
 - gathering archives 391
 - gathering service collections 392
 - installing and configuring 383
 - removing archives 390
 - removing service collections 392
 - viewing error logs 390
- error logs
 - archiving 390
 - creating archives 390
 - gathering archives 391
 - managing 383
 - removing archives 390
 - viewing 390
- error message output log
 - Event Management 347
- error notification objects, managing 385
- error templates
 - managing 387
- Estart command 206
- Event Management
 - abnormal termination of haemctrl add 351
 - disk space and tracing 351
 - performance and tracing 351
 - shared variables with performance monitoring 348
- Event Management API (EMAPI)
 - component of Event Management 339
- Event Management client (EM client)
 - definition 337
- Event Management communications
 - between Event Manager daemons 342
 - local EM clients 342
 - remote EM clients 342
 - resource monitors 342
- Event Management configuration
 - changing 343
 - SDR classes 343
- Event Management Configuration Database (EMCDB)
 - and Event Manager daemon initialization 353
 - compiling with haemcfg 344
 - creating 344
 - placing a new version in production 354
 - run-time directory 348
 - staging directory 346
 - version string and joining peer group 353
 - version string and peer group state 349
 - version string directory 348
- Event Management directories
 - /etc/ha/cfg 348
 - /spdata/sys1/ha/cfg 346
 - /var/ha/lck/haem 346
 - /var/ha/log 347
 - /var/ha/run 347
 - /var/ha/soc 346
- Event Management SDR classes
 - key attributes for loading 345
- Event Management subsystem
 - activating configuration data 358
 - adding with haemctrl command 350
 - changing configuration data in the SDR 358
 - cleaning with haemctrl command 351
 - client communication 339
 - component summary 338
 - components 338, 348
 - configuration 349, 351
 - configuring and operating 337, 359
 - deleting with haemctrl command 351
 - dependencies 348, 349
 - Event Manager daemon initialization 351, 354
 - Event Manager daemon operation 354, 355
 - getting subsystem status 355, 357
 - installation 349
 - introducing 337
 - loading non-PSSP data into the SDR 357
 - operational domain 339
 - recovery from failure (automatic) 354
 - starting with haemctrl command 351
 - stopping with haemctrl command 351
 - tracing with haemctrl command 351
 - unconfiguring the subsystem 351

- Event Manager daemon
 - abnormal termination core file 347
 - communications 342
 - component of Event Management 339
 - current working directory 347
 - definition 337
 - error message log file 347
 - getting status 355, 357
 - group membership 349
 - group subscriptions 349
 - haemctrl control script 345
 - initialization 351, 354
 - message trace output log file 347
 - operation 354, 355
 - recovery from failure (automatic) 354
 - trace output log file 347
 - use of AIX Error Log 347
- event processing 310
- examples
 - resource data
 - output from the SDRGetObjects command 474
 - using SP Perspectives 476
- expansion I/O unit 8
- extension nodes
 - adding 253
 - changing 255
 - implementations of 251
 - listing 259
 - managing 251
 - reconfiguring 257
 - removing 256
 - resetting 258
 - SNMP 253
 - transferring configuration information for 253
 - understanding 251

F

- failovers of control workstation 302
- file collection
 - adding and deleting files 56
 - adding automounter map to 101
 - automounter 98
 - building 58
 - HACWS and 305
 - installing 61
 - managing 41
 - removing 65
 - structure 46
 - updating 52
 - verifying using scan 51
- file management 10
- files 98
 - /usr/lpp/ssp/config/cron_template 122
 - accounting report and summary 123
 - afscient.cust 443

- files (*continued*)
 - ibmSPDepNode.my 446
 - ibmSPMIB.my 452
- files and directories
 - component of Event Management 346
 - component of Group Services 329
- files, accounting 122
- files, data 122
- files, nrunacct command 123
- filesets, Event Management
 - ssp.clients 349
 - ssp.ha 349
- filesets, Group Services
 - ssp.ha 330
- frame
 - frame description 6
- frame supervisor cables 312
- frame supervisor cards for HACWS 298
- from the command line 386

G

- group membership list
 - definition 325
- Group Services
 - abnormal termination of hagsctrl add 331
 - disk space and tracing 332
 - performance and tracing 332
- Group Services API (GSAPI)
 - component of Group Services 327
- Group Services client
 - definition 325
- Group Services communications
 - between Group Services daemons 328
 - local GS clients 328
- Group Services daemon
 - abnormal termination core file 329
 - communications 328
 - component of Group Services 327
 - current working directory 329
 - getting status 334, 335
 - hagsctrl control script 328
 - initialization 332, 333
 - initialization errors 333
 - operation 333, 334
 - recovery from failure (automatic) 333
 - trace output log file 329
- Group Services directories
 - /var/ha/lck 329
 - /var/ha/log 329
 - /var/ha/run 329
 - /var/ha/soc 329
- Group Services subsystem
 - adding with hagsctrl command 331
 - and Event Manager daemon initialization 352
 - cleaning with hagsctrl command 331

- Group Services subsystem (*continued*)
 - client communication 327
 - component summary 326
 - components 326, 330
 - configuration 330
 - configuring and operating 325, 335
 - deleting with hagsctrl command 331
 - dependencies 330
 - dependency by Event Management 349
 - getting subsystem status 334, 335
 - Group Services daemon initialization 332, 333
 - Group Services daemon operation 333, 334
 - initialization errors 333
 - installation 330
 - introducing 325
 - operational domain 327
 - recovery from failure (automatic) 333
 - starting with hagsctrl command 331
 - stopping with hagsctrl command 331
 - system groups 341
 - tracing with hagsctrl command 332
- group state
 - and joining the ha_em_peers group 353
 - EMCDB version string 349
- group state value
 - definition 325
- group, Group Services
 - definition 325
- groups
 - Group Services
 - restrictions on number per client 334
 - restrictions on number per domain 334
- GS client (Group Services client)
 - restrictions on number 333
- GS nameserver
 - establishing 332
- GSAPI (Group Services Application Programming Interface)
 - component of Group Services 327
- GSAPI libraries
 - location 327

H

- ha_em_peers group
 - joining 352
 - use by Event Management 349
- HACMP
 - configuring resources for 310
 - customizing for HACWS 310
 - event processing 310
 - HACWS foundation 298
 - operating for the control workstation 300
- HACWS
 - See also* High Availability Control Workstation (HACWS)
- HACWS (*continued*)
 - managing 295
- HACWS (High Availability Control Workstation)
 - and Event Management 350
 - and Group Services 331
- haemcfg command
 - compiling the EMCDB 344
- haemctrl command
 - adding the Event Management subsystem 350
 - cleaning the Event Management subsystem 351
 - control script for Event Management 345
 - deleting the Event Management subsystem 351
 - starting the Event Management subsystem 351
 - stopping the Event Management subsystem 351
 - summary of functions 350
 - tracing the Event Management subsystem 351
 - unconfiguring the subsystem 351
- haemd daemon
 - location 339
- haemloadcfg command
 - differences from SDRCreateObjects 345
 - Event Management configuration load utility 345
 - use of loadsdr program 346
- haemloadlist file
 - default Event Management configuration data 346
- haemrpcdb script
 - use by Event Management 354
- hagsctrl command
 - adding the Group Services subsystem 331
 - cleaning the Group Services subsystem 331
 - control script for Group Services 328
 - deleting the Group Services subsystem 331
 - starting the Group Services subsystem 331
 - stopping the Group Services subsystem 331
 - summary of functions 330
 - tracing the Group Services subsystem 332
- hagsd daemon
 - location 327
- halting the SP 21
- hardware monitor resource monitor 341
- hardware overview 3, 4
- help for SP Perspectives 274
- High Availability Control Workstation (HACWS) 295
 - and Event Management 350
 - and Group Services 331
 - configuration changes 303
 - customizing HACMP for 310
 - failovers of control workstations 302
 - hardware 297
 - managing subsystems for 304, 306, 308
 - accounting 306
 - automounter 306
 - dependent nodes 309
 - extension nodes 309
 - file collections 305
 - mail service 306
 - parallel management commands 305

- High Availability Control Workstation (HACWS)
 - (continued)
 - managing subsystems for (continued)
 - security features 307
 - starting up and shutting down system 304
 - Sysctl 308
 - system partitions 309
 - time synchronization 306
 - user accounts 305
 - software 298
 - terminology 296
 - understanding 296
- high availability services
 - Event Management subsystem 337
 - Group Services subsystem 325, 337
 - Topology Services subsystem 325, 337
- high node
 - time to power on 19
- high performance switch 7
- highlights of PSSP software 9
- Host Membership
 - resource monitor 341
- host name changes for SP systems 517
- host_responds
 - updating 485
- host_responds SDR class
 - resource monitor 341
- hostlist command 33
- HostMembership group 341
 - use by Event Management 349

I

- IBM parallel system support programs for AIX (PSSP) 9
- IBM products, other 12
- IBM.PSSP.CSSLogMon 341
- IBM.PSSP.harmlid resource monitor 341
- IBM.PSSP.harmpd resource monitor 341
- IBM.PSSP.hmrmd resource monitor 341
- IBM.PSSP.Membership.LANAdapter.state resource variable 349
- IBM.PSSP.Membership.Node.state resource variable 349
- IBM.PSSP.pm.User_state resource variables 370
- IBM.PSSP.pmanrmd resource monitor 341
- IBM.PSSP.Response.Switch.state resource variable 349
- IBM.PSSP.SDR 341
- ibmSPDepNode.my 446
- ibmSPMIB.my 452
- include command 184
- inetd.conf file 147
- installation 10, 121
 - Event Management subsystem 349
 - Group Services subsystem 330

- installing a file collection 61
- internal resource monitor
 - definition 340
 - supplied by PSSP 341
- introduction 3
- IP address changes for SP systems 517

J

- job management 11
- joining a Group Services group
 - Event Management subsystem 352

K

- Kerberos 140
 - client and server interaction 140
 - DES 142
 - password, master 155
 - principals 136
 - tickets 138
- Kerberos V4
 - restarting 167
 - stopping 167
- key attributes
 - Event Management SDR objects 345
- keys
 - changing 166
 - creating 164
 - displaying information for 165
 - distributing to client systems 167
 - managing 164

L

- LAPI
 - Active Message 401
 - C example 412
 - Fortran example 416
 - infrastructure 396
 - writing handlers example 397
 - address manipulation 403
 - advantages 395
 - allocating buffers 406
 - blocking and non-blocking calls 400
 - characteristics of functions 401
 - communication behaviors 400
 - compiling programs 407
 - completion checking 402
 - completion of communication operation 400
 - concepts 399
 - core dumps 410
 - data transfer 401
 - defined functions 399
 - environment 403
 - error handling 401, 403

- LAPI (*continued*)
 - executing programs 406
 - execution model 403
 - general functions 396
 - Get
 - C example 420
 - Fortran example 421
 - hangs 410
 - hints 408, 409
 - linking with libraries 410
 - message ordering and atomicity 400
 - messages 403
 - non-thread-safe-libraries 410
 - origin and target 399
 - overview 395
 - programming examples 412
 - progress 401, 403
 - PUT
 - C example 423
 - Fortran example 425
 - Rmw
 - C example 427
 - Fortran example 429
 - running programs 408
 - segment registers 411
 - setup 403
 - specific functions 401
 - synchronizing 402
 - understanding 396
 - using 401
 - why use 395
 - writing handlers 396
- LAPI_Address 403
- LAPI_Address_init: 403
- LAPI_Amsend 401
- LAPI_Getcptr 402
- LAPI_Init 403
- LAPI_Msg_String 403
- LAPI_Probe 403
- LAPI_Qenv 403
- LAPI_Rmw 402
- LAPI_Senv 403
- LAPI_Setcptr 402
- LAPI_Term 403
- LAPI_Waitcptr 402
- last file 48
- Launch Pad
 - bringing up 267
 - description of 268
 - elements of the 268
- line, from the command 386
- line, using the command 227, 230, 232, 235, 237, 238, 241, 242
- list file 56
- list file keywords and operands 66
- list node file system information 483
- list node hardware information 483
- list node network information 483
- listing a user account 78
- listing extension nodes 259
- listing resource variable information
 - from SP Perspectives 476
 - using load list files 473
 - using the SDRGetObjects command 473
- load command 186
- LoadLeveler
 - resource monitor 341
- loadsdr program
 - use by haemloadcfg command 346
- local EM clients
 - Event Management communications 342
- local GS clients
 - Group Services communications 328
- lock file
 - Event Management 346
 - Group Services 329
- log file
 - Event Management 347
 - Group Services 329
- log_event 368
- logical partition
 - definition 435
- logical volume
 - definition 435
- login control 81
- logs file 54
- lsautoent.cust script 111
- lscfg command 483
- lskp 156
- lssrc command
 - getting Event Management status 355
 - getting Group Services status 334

M

- mail 113
- mail services 113
 - HACWS and 306
- maintaining the crontabs file 127
- Management Information Base
 - SP 378
- management, configuration 10
- management, job 11
- management, user 11
- managing
 - file collection 41
 - user accounts 69
- managing error notification objects 385
- managing mail 113
- managing user information 69

- manual pages for public code 540
- map files for automounter 95
 - adding to file collections 101
 - creating 100
 - distributing to boot/install servers and nodes 102
- master map files for automounter 97
 - updating 101
- Membership resource monitor 341
- message boxes of SP Perspectives 273
- message trace output log
 - Event Management 347
- MIB (Management Information Base)
 - understanding 374
- mirroring
 - AIX support of 436
 - and quorum 435
 - definition 435
 - PSSP support of 436
 - restrictions 439
 - root volume group 437
- mkautoent.cust script 109
- mkkp 159
- modifying the file collection hierarchy 63
- monitoring and operating the SP system 279

N

- nameserver, Group Services
 - establishing 332
- Netfinity server 5
- netstat command 483
- Netview for AIX
 - configuring for the SP system 381
- network connections for HACWS 298
- network connectivity adapters 8
- Network File System (NFS) 220
- NFS
 - mounts over a switch 220
- node groups
 - commands for creating, modifying, and deleting 247
 - commands for examining 247
 - commands that take node groups as arguments 249
 - creating 246
 - creating, modifying, and deleting 247
 - definition list 245
 - deleting 248
 - examining 247, 248
 - example of resolving 246
 - flattening 246
 - global 245
 - managing 245
 - modes 245
 - names 245
 - partition-bound 245
 - resolving 246

- node groups (*continued*)
 - SDR information 245
 - SMIT panels for creating, modifying, and deleting 247
 - SMIT panels for deleting 248
 - SMIT panels for examining 248
 - understanding 245
- node jobs, accounting for exclusive use of 116
- node recovery 217
- node.root file collection 45
- nodes
 - Group Services
 - restrictions on number per domain 333
 - monitoring on a switch 216
 - overview of sequencing 17
 - nodes, primary 209
 - nodes, primary backup 209
 - nodes, processor 5
 - notebooks of SP Perspectives 273
 - notification object, installing a 386
 - notification objects, managing error 385
 - notify_event 368
 - notifying users of impending shutdown 23
 - nrunacct command files 123
 - NTP 89
 - NTP options 89

O

- objects, managing error notification 385
- observation interval
 - changing 344
- omit command 66
- omitany command 66
- operation 10
- overview of RS/6000 SP 3
- overview, hardware 4
- overview, software 8

P

- p_cat 35
- packaging 12
- parallel management commands 31, 34
 - HACWS and 305
 - sample 34
- Parallel System Support Programs for AIX (PSSP), IBM 9
- partitioning, system 225
 - applying a configuration 237
 - configuration
 - displaying information for 230
 - selecting 228
 - HACWS and 309
 - layouts
 - selecting 232

- partitioning, system (*continued*)
 - overview 224
 - steps for 227
 - understanding 221
- partitions, system
 - changed
 - rebooting 239
 - changing 235
 - customizing 233
 - default system partition 223
 - displaying configuration information 239
 - managing system partition-sensitive subsystems 242
 - verifying configuration information 241
- partitions, system
 - managing 221
- passwd command 79
- password, Kerberos master 155
- pcp 35
- pdf 37
- peer group, Event Management
 - joining 352
- Performance Aide for AIX
 - dependency by Event Management 348
- performance monitoring
 - shared variables with Event Management 348
- Performance Toolbox for AIX
 - dependency by Event Management 348
 - use of shared resource variables 348
- Performance Toolbox Parallel Extensions (PTPE)
 - use of shared resource variables 348
- Perspective
 - description of 269
 - elements of 269
- Perspectives 267
- pexec 35
- pexscr 35
- pfck 39
- pfind 36
- pfps 37
- physical partition
 - definition 435
- physical volume
 - definition 435
- pls 36
- pman 361
- pmand
 - controlling 364
 - understanding 363
- pmandef
 - issuing an SNMP trap for an event 368
 - logging an event 369
 - purpose 365
 - running a command for an event 366
 - security considerations 365
- pmandefaults 369
- pmanquery 369
- pmanrmd
 - configuring 371
 - controlling 370
 - SDR information 372
 - understanding 370
- pmanrmd.conf 370, 371
- pmanrmdloadSDR 372
- pmanrminput 370
- pmv 36
- port numbers
 - component of Event Management 341
 - component of Group Services 328
- power_system file collection 45
- powering on the SP
 - safety warning 18
- pps 36
- predicate
 - definition 337
- prefix file 56
- primary authentication server 149
- primary node takeover 206
- principals
 - adding and assigning passwords 158
 - deleting 163
 - displaying information 157
 - listing 156
 - managing 156
 - service 136
 - types 136
 - user 136
- print management 9
- prm 36
- problem determination
 - Event Management subsystem
 - abnormal termination core file 347
 - abnormal termination of haemctrl add 351
 - getting subsystem status 355
 - tracing 351
 - Group Services subsystem
 - abnormal termination core file 329
 - abnormal termination of hagsctrl add 331
 - getting subsystem status 334
 - tracing 332
- problem management 11
 - resource monitor 341
- Problem Management subsystem
 - components of 362
 - creating resource monitors 370
 - creating subscriptions 365
 - default events 369
 - environment variables 367
 - IBM.PSSP.pm.User_state resource variables 370
 - log_event 368
 - logging an event 369

Problem Management subsystem (*continued*)

- notify_event 368
- pmand 363
- pmandef 365
- pmandefaults 369
- pmanquery 369
- pmanrmd 370
- pmanrmd.conf 370
- pmanrmdloadSDR 372
- pmanrminput 370
- requesting information about a subscription 369
- resource monitor daemon 370
- resource variables and values 370
- running a command for an event 366
- SNMP trap, issuing 368
- understanding 361

process resource monitor 341

processor nodes 5

protocol, Group Services

- definition 325

protocol, switch communication 204

provider

- definition 325

PSSP 9

Q

quiescing the SP Switch 219

R

realms 137

reboot

- and Event Management 350
- and Group Services 331

rebooting the SP 22

reconfiguring 359

- Event Management subsystem 358

reconfiguring extension nodes 257

recovery from failure

- Event Management 354
- Group Services 333

refauto.cust script 106

refuse file 62

refusing files in a file collection 62

registration cache

- and Event Manager daemon 352
- Event Management 347

remote EM clients

- Event Management communications 342

removing a file collection 65

removing extension nodes 256

report and summary files, accounting 123

reporting file collection information 50

required file collections 44

resetting extension nodes 258

resource center 10

resource class

- configuration in SDR 343

resource data, listing

- using the SDRGetObjects command 473

resource IDs

- configuration in SDR 343

resource monitor

- component of Event Management 340
- configuration in SDR 343
- connection types 340
- current working directory 347
- definition 337
- Event Management communications 342

Resource Monitor Application Programming Interface (RMAPI)

- component of Event Management 340
- use of EMCDB version string directory 348
- use of shared memory key 346

resource monitor, non-PSSP

- loading configuration data for 344, 345, 357

resource monitor, third-party

- loading configuration data for 344, 345, 357

resource monitors

- creating 370

resource monitors supplied by PSSP

- aixos 341
- IBM.PSSP.CSSLogMon 341
- IBM.PSSP.harmlid 341
- IBM.PSSP.harmpd 341
- IBM.PSSP.hmrmd 341
- IBM.PSSP.pmanrmd 341
- IBM.PSSP.SDR 341
- Membership (internal) 341
- Response (internal) 341

resource variable

- configuration in SDR 343
- definition 337
- IBM.PSSP.Membership.LANAdapter.state 349
- IBM.PSSP.Membership.Node.state 349
- IBM.PSSP.Response.Switch.state 349

resource variable information, displaying

- using the spevent command 476

resource variable instance limits

- changing resource variable instance limits 359

resource variable, PSSP-supplied

- listing 473

Response resource monitor 341

restarting the SP 18, 22

restrictions

- Event Management
 - number of EM clients 354
 - observation of resource variable values 355
- Group Services
 - groups per client 334
 - groups per domain 334

- restrictions (*continued*)
 - Group Services (*continued*)
 - nodes per domain 333
 - number of GS clients 333
- RMAPI (Resource Monitor Application Programming Interface)
 - component of Event Management 340
 - use of EMCDB version string directory 348
 - use of shared memory key 346
- RMAPI libraries
 - location 340
- rmautoent.cust script 110
- root volume group
 - definition 435
 - description 435
 - managing 435
 - mirroring 437
 - PSSP support of 436
- RS/6000 SP
 - character istics of 3
 - overview 3
- RS/6000 SP Switch Router 251
- run-time EMCDB
 - creating 345, 353

S

- sample files 443
- SBS (structured byte string)
 - configuration in SDR 343
- SDR 477, 486
 - archiving 227
 - pmanrmd information 372
 - shadow files 486
- SDR (System Data Repository)
 - and haemctrl clean 351
 - and haemctrl delete 351
 - and hagsctrl clean 331
 - and hagsctrl delete 331
 - dependency by Event Management 349
 - dependency by Group Services 330
 - key attributes for Event Management objects 345
 - use by Event Management 337
- SDR authentication 479
- SDR authorization 479
- SDR daemon log 480
- SDR server 480
- SDRCreateObjects command
 - differences from haemloadcfg command 345
- SDRGetObjects command
 - example of output 474
 - listing resource variable information 473
- secondary authentication server 150
- secondary servers 146
- security 11, 131
- selecting nodes
 - primary 209
 - primary backup 209
- selecting primary and primary backup nodes 209
- sequencing nodes
 - about 17
 - default 18, 26
 - defining 25
- sequencing of nodes
 - overview 17
- server
 - type of resource monitor 340
- service collections
 - creating 391
 - gathering 392
 - removing 392
- service, collecting system data for IBM 391
- set command 186
- setting time 91
- setting up a primary authentication server 149
- setting up a secondary authentication server 150
- setting up SP accounting 117
- shadow files 486
- shared memory key
 - use by RMAPI 346
- shutting down and rebooting the SP 22
- shutting down the SP 21
 - checking the status of 24
 - customized shutdown script 23
 - notifying users before 23
 - overview 17
 - phases of shutdown processing 23
- single point of control. 31
- SMIT 9
- SMIT interface access 480
- SMIT, using 230
- SNMP
 - contents of traps 377
 - extension nodes 253
 - traps from AIX Error Log events 375
 - traps from Event Management events 377
 - understanding 374
- SNMP Support subsystem
 - and extension nodes 253
- SNMP traps
 - pmandef to specify issuing 368
- snmp_trap_gen 375
- sockets
 - component of Event Management 341
 - component of Group Services 328
 - names used by Event Management 342
- software overview 8
- SP
 - overview of starting up and shutting down 17
 - powering on 18
 - restarting 18

- SP (*continued*)
 - starting 18
- SP hardware state
 - resource monitor 341
- sp perspectives 10
 - applications 267
 - authorizing users for 265
 - dialog boxes 271
 - example of displaying resource data 476
 - getting started 264
 - graphical elements 267
 - help for 274
 - Launch Pad 268
 - listing resource variable information 476
 - message boxes 273
 - notebooks 273
 - Perspective application windows 269
 - Perspectives 269
 - SP Resource Center 268
 - starting 267
 - tasks that can be performed with 264
 - understanding the interface 267
 - Using 263
- SP Resource Center
 - Perspectives 268
- SP Switch 6
- SP Switch Router Adapter 251
- SP user management options 74
- SP-attached server 5
 - frame 6
- SP-Attached Servers 207
- sp_configd 373, 375
- SP_NAME environment variable
 - and haemcfg command 344
 - and haemctrl command 345
 - and haemloadcfg command 345
 - and hagsctrl command 328
- SP_ports SDR class
 - use by Event Management 342
- SP/Netfinity server consolidation 12
- SP—attached servers for HACWS 298
- spchuser command 75
 - home attribute 75
- special subsystems
 - creating 28
- spevent command
 - example of output 476
 - listing resource variable information 476
- SPMI (System Performance Monitor Interface)
 - dependency by Event Management 348
- SPMI library
 - shared memory key 346
- spmuser command 75
 - home attribute 75
- SRC (System Resource Controller)
 - and Event Manager daemon 352
- SRC (System Resource Controller) (*continued*)
 - and Group Services daemon 332
 - dependency by Group Services 330
- ssp.clients filesset, PSSP
 - and Event Management 339, 349
 - and Group Services 330
- ssp.ha filesset, PSSP
 - and Event Management 349
 - and Group Services 330
- staging file for EMCDB 344
- startauto.cust script 105
- starting subsystems
 - Event Management (haemctrl) 351
 - Group Services (hagsctrl) 331
- starting the SP 18
 - overview 17
- starting the switch 213
- starting up the SP
 - checking the status of 24
- status, Event Management
 - output of lssrc command 355, 357
- status, Group Services
 - output of lssrc command 334, 335
- stopping subsystems
 - Event Management (haemctrl) 351
 - Group Services (hagsctrl) 331
- structured byte string (SBS)
 - configuration in SDR 343
- structured field
 - configuration in SDR 343
- subscriber
 - definition 325
- subsystem
 - Event Management 337, 359
 - Group Services 325, 335
 - system partition-sensitive 328, 345
- subsystem status
 - for Event Management 355, 357
 - for Group Services 334, 335
- subsystems
 - special 28
 - See also* special subsystems
- subsystems, system partition-sensitive
 - listing 243
 - managing 242
- summary and report files, accounting 123
- sup.admin file collection 44
- supper activate command 46
- supper install command 48
- supper offline command 63
- supper program 42
- supper reset command 48
- supper scan command 48
- supper status command 44
- supper subcommands table 50

- supper update command 49
- switch 204
 - clock reset 209
 - clocks 208
 - description of 6
 - setting switch clock source
 - SMIT panels 208
 - switch topology files 211, 212
- switch admin daemon 216
- switch clocks
 - reset of 209
 - selecting 208
- switch connectivity 214
- switch considerations 207
- switch protocol 204
- switch topology files
 - annotating 212
 - managing 211
- switch_responds SDR class
 - and Event Management 349
 - resource monitor 341
- switch, automatic unfence 214
- switch, high performance 7
- switch, SP 6
- switch, SP-attached server 7
- switch, starting 213
- switch, system partitioning 206
- switch, using 203
- switches
 - monitoring nodes on 216
 - quiescing 219
- symlinkall command 67
- synchronization, time 91
- syntax rules 66
- sysctl 186
 - application development
 - passing parameters to a script 196
 - sending a script 194
 - specifying server options 195
 - architecture 179
 - Authentication Services, relationship to 179
 - authorization
 - adding new callbacks 187
 - components 180
 - levels 180
 - variables 193
 - built-in commands 197
 - commands
 - built-in sysctl commands 197
 - Extended Tcl command summary 197
 - Tcl command summary 197
 - components, illustration of 179
 - configuration
 - debugging problems 183
 - procedure 182
 - configuration commands
 - create class 185
 - sysctl (*continued*)
 - configuration commands (*continued*)
 - create proc 185
 - create var 184
 - include 184
 - load 186
 - set 186
 - description of 177
 - executables
 - sysctl 181
 - sysctld 181
 - Extended Tcl command summary 197
 - files 181
 - sysctl.conf 181
 - sysctld.log 182
 - HACWS and 308
 - installation 182
 - Tcl command summary 197
 - using
 - directing commands to remote hosts 189
 - interactive mode 188
 - non-interactive mode 189
 - prerequisites 188
 - sending commands to a collection of nodes 190
 - sending commands to a server 189
 - Sysctl-based 37
 - Syspar Controller
 - managing system partition-sensitive subsystems 242
 - Syspar SDR class 344
 - syspar_ctrl command
 - and Event Management subsystem 345
 - and Group Services subsystem 328
 - listing system partition-sensitive subsystems 243
 - managing system partition-sensitive subsystems 242
 - Syspar_ports SDR class
 - use by Event Management 342
 - use by Group Services 328
 - System Data Repository 477, 486
 - default sequencing based on 18
 - System Data Repository (SDR)
 - and haemctrl clean 351
 - and haemctrl delete 351
 - and hagsctrl clean 331
 - and hagsctrl delete 331
 - archiving 227
 - configuration data for Event Management 343
 - dependency by Event Management 349
 - dependency by Group Services 330
 - key attributes for Event Management objects 345
 - use by Event Management 337
 - System Management Interface Tool (SMIT) 9
 - system partition
 - and Event Management 339
 - and Event Management SDR classes 343

- system partition (*continued*)
 - and Group Services 327
 - specifying for haemcfg command 344
 - specifying for haemctrl command 345
 - specifying for haemloadcfg command 345
 - specifying for hagsctrl command 328
- system partitioning and the switch 206
- System Performance Monitor Interface (SPMI)
 - dependency by Event Management 348
- System Resource Controller (SRC)
 - and Event Manager daemon 352
 - and Group Services daemon 332
 - dependency by Group Services 330
- system resources
 - listing PSSP-supplied resource variables 473

T

- takeover, primary node 206
- TCP port
 - use by Event Management 341
- TCP socket
 - Event Management client communication 339
- third-party resource monitor
 - loading configuration data for 344, 345, 357
- tickets
 - cache files 138
 - lifetimes 139
 - understanding 138
- time limits
 - Event Management
 - connection to Group Services 352
 - observation intervals 355
 - peer group joining 352
 - reconnection to resource monitors 354
 - Group Services
 - connection to Topology Services 332
- time setting 91
- time synchronization 91
 - HACWS and 306
- Topology Services subsystem
 - and Group Services daemon initialization 332
 - dependency by Group Services 330
- trace output log
 - Event Management 347
 - Group Services 329
- tracing subsystems
 - Event Management (haemctrl) 351
 - Group Services (hagsctrl) 332
- trademarks xiv
- trimming log files 389
- troubleshooting
 - Event Management subsystem
 - abnormal termination core file 347
 - abnormal termination of haemctrl add 351
 - getting subsystem status 355
 - tracing 351

- troubleshooting (*continued*)
 - Group Services subsystem
 - abnormal termination core file 329
 - abnormal termination of hagsctrl add 331
 - getting subsystem status 334
 - initialization errors 333
 - tracing 332
- tuning 9

U

- UDP port
 - use by Event Management 341
 - use by Group Services 328
- unconfiguring subsystems
 - Event Management (haemctrl) 351
- understanding file collections 42
- understanding the file collection structure
 - directory and master files 46
- understanding the system data repository 477
- Unix domain socket
 - Event Management client communication 339
 - Group Services client communication 327
 - resource monitor communication 339
 - use by Event Management 342
 - use by Group Services 328
- user accounts
 - adding 69
 - changing 69
 - deleting 69
 - HACWS and 305
- user directories
 - automounter map for 99
 - changing 99
- user management 11
- user.admin file collection 45
- using a switch 203
- using SMIT 227, 228, 230, 232, 233, 236, 238, 240, 241
- using the command line 227, 230, 232, 235, 237, 238, 241, 242

V

- verifying file collections using scan 51
- version string for EMCDB 344
- virtual shared disk management 10
- volume group
 - definition 435
- VSD subsystem
 - resource monitor 341

W

- when file 48

workstation outside SP
and Event Management 339
workstation, control 7

Y

yppasswd command 79

Communicating Your Comments to IBM

Parallel System Support Programs for AIX
Administration Guide
Version 3 Release 1.1
Publication No. SA22-7348-01

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM. Whichever method you choose, make sure you send your name, address, and telephone number if you would like a reply.

Feel free to comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. However, the comments you send should pertain to only the information in this manual and the way in which the information is presented. To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

If you are mailing a reader's comment form (RCF) from a country other than the United States, you can give the RCF to the local IBM branch office or IBM representative for postage-paid mailing.

- If you prefer to send comments by mail, use the RCF at the back of this book.
- If you prefer to send comments by FAX, use this number:
 - FAX: (International Access Code)+1+914+432-9405
- If you prefer to send comments electronically, use one of these network IDs:
 - IBM Mail Exchange: USIB6TC9 at IBMMAIL
 - Internet e-mail: mhvrcfs@us.ibm.com

Make sure to include the following in your note:

- Title and publication number of this book
- Page number or topic to which your comment applies

Optionally, if you include your telephone number, we will be able to respond to your comments by phone.

Reader's Comments — We'd Like to Hear from You

**Parallel System Support Programs for AIX
Administration Guide
Version 3 Release 1.1**

Publication No. SA22-7348-01

You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you. Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

Note: Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

Today's date: _____

What is your occupation?

Newsletter number of latest Technical Newsletter (if any) concerning this publication:

How did you use this publication?

- | | | | |
|--------------------------|-------------------------------|--------------------------|------------------------|
| <input type="checkbox"/> | As an introduction | <input type="checkbox"/> | As a text (student) |
| <input type="checkbox"/> | As a reference manual | <input type="checkbox"/> | As a text (instructor) |
| <input type="checkbox"/> | For another purpose (explain) | | |

Is there anything you especially like or dislike about the organization, presentation, or writing in this manual? Helpful comments include general usefulness of the book; possible additions, deletions, and clarifications; specific errors and omissions.

Page Number: Comment:

Name

Address

Company or Organization

Phone No.



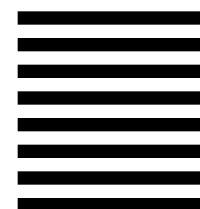
Fold and Tape

Please do not staple

Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES



BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Department 55JA, Mail Station P384
522 South Road
Poughkeepsie NY 12601-5400



Fold and Tape

Please do not staple

Fold and Tape



Program Number: 5765-D51



Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SA22-7348-01

