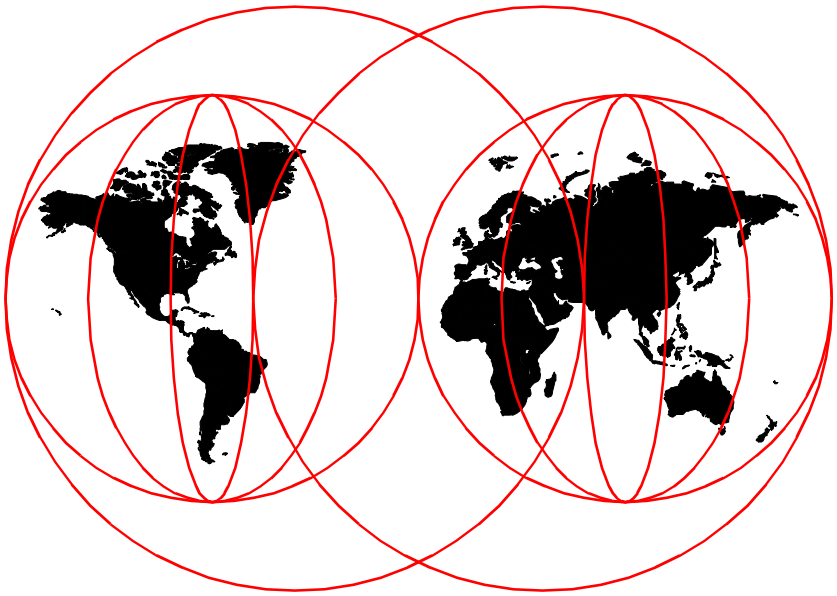
The RS/6000 SP Inside Out

Marcelo R. Barrios, Mario Bono, Michael Hennecke
Teerachai Supapunpinyo, Bernard Woo, Steven Yap



International Technical Support Organization

http://www.redbooks.ibm.com

SG24-5374-00

SG24-5374-00



International Technical Support Organization

The RS/6000 SP Inside Out

May 1999

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix B, “Special Notices” on page 521.

First Edition (May 1999)

This edition applies to PSSP Version 3, Release 1 (5765-D51) for use with the AIX Operating System Version 4, Release 3 Modification 2.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. HYJ Mail Station P099
522 South Road
Poughkeepsie, New York 12601-5400

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

© Copyright International Business Machines Corporation 1999. All rights reserved
Note to U.S Government Users – Documentation related to restricted rights – Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

Figures xi

Tables xvii

Preface xix
The Team That Wrote This Redbook xix
Comments Welcome xxi

Part 1. The Big Picture 1

Chapter 1. History and Design Philosophy 3
1.1 Ultra-Large Computing Problems 3
1.2 Origins of the SP 4
1.3 High-Level System Characteristics 6
1.3.1 Scalability 6
1.3.2 Use of Mainstream Architectures and Technologies 7
1.3.3 Flexibility 8
1.3.4 Manageability 8
1.4 Future Directions for the RS/6000 SP 8

Chapter 2. RS/6000 SP Architecture 11
2.1 The SP as a Cluster 13
2.2 The SP as a Parallel Machine 14

Part 2. System Implementation 17

Chapter 3. Hardware Components 19
3.1 Frames 20
3.1.1 Tall Frames 21
3.1.2 Short Frames 21
3.1.3 SP Switch Frames 22
3.1.4 Power Supplies 23
3.1.5 Hardware Control and Supervision 24
3.2 Standard Nodes 26
3.2.1 Internal Nodes 26
3.2.2 External Nodes 34
3.3 Dependent Nodes 38
3.3.1 SP Switch Router 39
3.3.2 SP Switch Router Attachment 41
3.4 Control Workstation 42
3.4.1 Supported Control Workstations 43

© Copyright IBM Corp. 1999

3.4.2 Control Workstation Minimum Hardware Requirements 44
3.4.3 High Availability Control Workstation 45
3.5 Administrative Ethernet 47
3.5.1 Frame and Node Cabling 47
3.5.2 SP LAN Topologies 49
3.6 SP Switch Communication Network 58
3.6.1 SP Switch Hardware Components 60
3.6.2 SP Switch Networking Fundamentals 65
3.6.3 SP Switch Network Products 70
3.7 Peripheral Devices 72
3.8 Configuration Rules 73
3.8.1 Short Frame Configuration 75
3.8.2 Tall Frame Configuration 77
3.9 Numbering Rules 83
3.9.1 The Frame Numbering Rule 83
3.9.2 The Slot Numbering Rule 84
3.9.3 The Node Numbering Rule 85
3.9.4 The Switch Port Numbering Rule 86

Chapter 4. System Data Repository and System Partitioning 91
4.1 SDR Data Model 92
4.2 Partitioning the SP System 95
4.2.1 Partitioning Rules 96
4.2.2 Partitions and Switchless SP Systems 96
4.2.3 Why Partition 97
4.3 SDR Daemon 98
4.3.1 Client/Server Communication 98
4.3.2 Locking 102
4.4 User Interface to SDR 102
4.4.1 Shadow Files 103
4.4.2 Manipulating SDR Data 103
4.4.3 Useful SDR Commands 105
4.5 The SDR Log File 106
4.6 Backup and Restore of SDR 107

Chapter 5. Hardware Control Subsystem 109
5.1 Components That Can Be Monitored And Controlled 109
5.2 How It Works 110
5.2.1 SP-Attached Servers 112
5.2.2 Security Aspect 116
5.3 User Interfaces 116
5.3.1 Command Line 116
5.3.2 SP Perspectives 119

Ex.1010 / Page 6 of 572

Chapter 6. Time Service 131
6.1 Network Time Protocol (NTP) 131
6.1.1 NTP Architecture 132
6.1.2 Configuring and Using NTP on the SP 133
6.2 Distributed Time Service (DTS) 135
6.2.1 Architecture of DCE DTS 135
6.2.2 Configuring and Using DTS on the SP 137

Chapter 7. SP Security 139
7.1 General Security Concepts 139
7.2 AIX Security 141
7.2.1 Secure Remote Execution Commands 141
7.2.2 Securing X11 Connections 144
7.2.3 Secure Shell 146
7.2.4 Login Control in AIX 147
7.3 How Kerberos Works 148
7.3.1 Kerberos Keys and Initial Setup 149
7.3.2 Authenticating to the Kerberos Server 150
7.3.3 Authenticating to an Application Server 152
7.4 Managing Kerberos on the SP 156
7.4.1 PSSP Kerberos (Version 4) 156
7.4.2 AFS Kerberos (Version 4) 168
7.4.3 DCE Kerberos (Version 5) 168
7.5 SP Services Which Utilize Kerberos 170
7.5.1 Hardware Control Subsystem 171
7.5.2 Remote Execution Commands 173
7.5.3 Sysctl 180

Chapter 8. RS/6000 Cluster Technology 185
8.1 "In the Beginning ...". 185
8.2 RSCT Packaging 186
8.3 RSCT Architecture Overview - “The Picture” 187
8.4 Topology Services (TS) 188
8.4.1 Group Leaders, Crown Princes, and Death in the Family 190
8.4.2 Network Connectivity Table - The Network Roadmap 192
8.5 Group Services (GS) 193
8.5.1 GS Components - Functional Overview 195
8.5.2 Creating GS Groups 196
8.5.3 Democracy Comes to the SP - Voting Protocols 198
8.5.4 Other GS Facilities 199
8.6 Event Management (EM) 200
8.6.1 Scalable Design 202
8.6.2 Your Application and EM 204

Ex.1010 / Page 7 of 572

8.6.3 EM and GS 204
8.6.4 Resource Monitors - The SP Implementation 205
8.7 Putting It All Together - A Simple Example of Exploiting RSCT 209
8.7.1 Scenario 209
8.7.2 TS Level 210
8.7.3 GS Level 211
8.7.4 EM Level 212
8.7.5 ES Client - Host_Responds Daemon 213
8.7.6 Perspectives Shows Green 213

Chapter 9. SP Switch Software 215
9.1 Switch Operating Principles 216
9.2 Primary Node Behavior 217
9.3 Secondary Nodes Behavior 218
9.4 Switch Topology File 218
9.5 Routing in SP Switch 223
9.6 Switch Administration and Management 225
9.6.1 Switch Admin Daemon 226
9.6.2 Automatic Node Unfence 231
9.6.3 Switch-Dependent Application Startup 235
9.6.4 SP Switch Management 235
9.7 Switch Clocks 238
9.8 SP Switch Error Logging 240
9.8.1 Consolidated Switch Error Logging 240
9.8.2 Switch Log Files 247
9.9 The out.top File 248
9.10 System Partitioning and the SP Switch 249
9.10.1 Considerations for Partitioning 250
9.10.2 System Partitions Directory Structure 251

Part 3. System Management 255

Chapter 10. Installation and Configuration 257
10.1 Concept of SP Installation 257
10.1.1 Role of Control Workstation and Boot/Install Servers 258
10.1.2 NIM Concept 258
10.1.3 Installation Procedure 261
10.2 Control Workstation Installation 261
10.2.1 AIX Installation 261
10.2.2 Setting Up The AIX Environment 262
10.2.3 Setting Up The PSSP Environment 266
10.3 Frame, Node And Switch Installation 267
10.3.1 Site Environment And Frame Information 267

10.3.2 Node Database Information 272
10.3.3 Setting Up The Switch 286
10.3.4 Node Installation. 287

Chapter 11. System Monitoring 293
11.1 SP Perspectives 293
11.1.1 Hardware Perspective 295
11.1.2 Event Perspective. 296
11.1.3 Virtual Shared Disk Perspective 304
11.1.4 SP Command Line System Monitor 304
11.2 SP System Tuning & Performance Monitoring 305
11.2.1 RS/6000 SP Performance Considerations 305
11.2.2 Performance Management Tools 311
11.3 SP Accounting 319
11.3.1 Configuring SP Accounting 320
11.3.2 LoadLeveler Accounting 323
11.4 Problem Management 326
11.4.1 AIX, BSD and PSSP-Specific Error Logging 327
11.4.2 Problem Management (PMAN) Subsystem 329
11.4.3 IBM Support Tools 333

Chapter 12. User Management 339
12.1 Managing User Databases 340
12.1.1 AIX: files (local) 340
12.1.2 SP User Management (SPUM) 341
12.1.3 SP Access Control (SPAC). 346
12.2 Network Information System (NIS) 348
12.3 File Collections. 355
12.3.1 Predefined File Collections 357
12.3.2 File Collections File Structure 358
12.3.3 How File Collections Work 360
12.3.4 Refusing Files in a File Collection. 361
12.3.5 Adding or Deleting Files in a File Collection 362
12.3.6 Building a File Collection 362
12.3.7 Installing a File Collection. 363
12.3.8 Removing a File Collection 364
12.4 Managing the Automounter 364
12.4.1 The BSD Automounter 365
12.4.2 The AIX Automounter 366
12.4.3 Examples (SP Setup, Migration, Coexistence) 368

Chapter 13. Backup and Recovery 375
13.1 Backup/Recovery of rootvg. 375
13.1.1 Backup of the Control Workstation 377

13.1.2 Backup/Recovery of Nodes 378
13.2 Backup/Recovery of System Database Repository (SDR) 379
13.3 Backup/Recovery of Kerberos Database 380
13.4 Backup/Recovery of User Specified Volume Group 381
13.5 Proper Documentation 382

Part 4. Application Enablers 383

Chapter 14. Data Storage 385
14.1 Local Disks 385
14.2 Global File Systems 387
14.2.1 Network File System (NFS) 387
14.2.2 The DFS and AFS File Systems 390
14.3 Shared Disks 394
14.3.1 Virtual Shared Disks (VSD) 395
14.3.2 Hashed Shared Disks (HSD) 404
14.3.3 Recoverable Virtual Shared Disks (RVSD) 405
14.4 General Parallel File System (GPFS) 409
14.4.1 The mmfsd Daemon 412
14.4.2 GPFS Setup 415
14.4.3 Managing GPFS 427
14.4.4 Migration and Coexistence 431

Chapter 15. High Availability 433
15.1 Components and Relationships 433
15.2 Modes of Operation 434
15.3 Planning Considerations 435
15.3.1 Cluster 435
15.3.2 Application 435
15.3.3 Network 436
15.3.4 Data Loss Protection 438
15.3.5 Resource Group 440
15.4 Cluster Configuration 441
15.4.1 Server 441
15.4.2 Client 450
15.5 Starting and Stopping Cluster Services 450

Chapter 16. Parallel Programming 453
16.1 Parallel Operating Environment (POE) 453
16.1.1 POE Architecture 455
16.1.2 Controlling the POE Runtime Environment 458
16.1.3 The POE Compilation Scripts 459
16.1.4 POE Resource Management 460

Ex.1010 / Page 10 of 572

16.1.5 POE Security Integration 461
16.2 Message Passing Libraries. 463
16.2.1 What is in MPI 463
16.2.2 IBM's Implementation of MPI 466
16.2.3 MPI-IO Subset 474
16.2.4 Parallel Virtual Machine 475
16.2.5 Low-Level API (LAPI) 475
16.2.6 Message Passing Library (MPL) 476
16.3 High Performance Fortran 476
16.3.1 What is in HPF 477
16.3.2 IBM XL High Performance Fortran 478
16.3.3 Portland Group pgHPF 479
16.4 Parallel Programming Tools 480
16.4.1 Parallel Debuggers 480
16.4.2 Profiling and Tracing. 487

Chapter 17. LoadLeveler 497
17.1 Architecture 497
17.2 Configuration of LoadLeveler 501
17.3 Serial Jobs 502
17.4 Parallel Jobs 507
17.5 Scheduling 511
17.6 SMP Features 514
17.7 DCE Security Integration 514

Appendix A. Currently Supported Adapters 517

Appendix B. Special Notices 521

Appendix C. Related Publications. 525
C.1 International Technical Support Organization Publications 525
C.2 Redbooks on CD-ROMs. 525
C.3 Other Publications 526
C.4 External Publications 527

How to Get ITSO Redbooks 529
IBM Redbook Fax Order Form. 530

List of Abbreviations. 531

Index 533

ITSO Redbook Evaluation. 547

Figures

1. MIMD System's Classification 13
2. Sample RS/6000 SP with External Node 20
3. Front View of Short Components 22
4. SP Switch Frame with 8 Intermediate Switch Boards (ISBs) 23
5. Front and Rear Views of Tall Frame Components 24
6. Frame Supervisor Attachment 25
7. 332 MHz SMP Node Component Diagram 27
8. POWER3 SMP Node Component Diagram 29
9. 332 MHz SMP Node System Architecture 31
10. POWER3 SMP Node System Architecture 33
11. RS/6000 S70/S7A System Scalability 36
12. The SP-Attached Server Connection 38
13. SP Switch Router 40
14. GRF Model 400 and Model 1600 41
15. High Availability Control Workstation (HACWS) Attachment 46
16. Shared 10BASE-2 SP Network 49
17. Segmented 10BASE-2 SP Network with Two Subnets 50
18. Segmented SP Network with Boot/Install Server Hierarchy 52
19. Boot/Install Server Hierarchy with Additional Router 53
20. Switched 10BASE-2 SP Network with Fast Uplink 54
21. Simple 100BASE-TX SP Network 57
22. Heterogeneous 10/100 Mbps SP Network 58
23. SP Switch Board 61
24. Relationship Between Switch Chip Link and Switch Chip Port 62
25. SP Switch Chip Diagram 63
26. SP Switch Adapter 64
27. SP Switch System 65
28. 16-Node SP System 66
29. 32-Node SP System 67
30. SP 48-Way System Interconnection 67
31. 64-Way System Interconnection 68
32. SP 80-Way System Interconnection 69
33. SP 96-way System Interconnection 70
34. Internal Bus Architecture for PCI-Based SMP Nodes 72
35. Minimum Nonswitched Short Frame Configurations 76
36. Example of Nonswitched Short Frame Configuration 76
37. Maximum SP Switch-8 Short Frame Configurations 77
38. Example of SP Switch-8 Tall Frame Configurations 78
39. Example of Single SP-Switch Configurations 81
40. Example of Multiple SP-Switches Configuration 82

© Copyright IBM Corp. 1999

41. Example of Two-Stage SP Switch Configuration . . . 83
42. Slot Numbering for Short Frame and Tall Frame . . . 85
43. Node Numbering for an SP System . . . 86
44. Switch Port Numbering for an SP Switch . . . 88
45. Example of Switch Port Numbering for an SP Switch-8 . . . 89
46. Sample SDR Classes . . . 93
47. SDR Directory Structure . . . 94
48. Node<->sdrd Communication Process (Non-Partitioned Example) . . . 100
49. Node<->sdrd Communication Process (Partitioned Example) . . . 101
50. Hardware Monitor Components and Relationship . . . 112
51. Hardware Monitoring For SP-Attached S70/S7A Servers . . . 114
52. Hardware Monitoring For SP-Attached Netfinity Servers . . . 115
53. SP Perspectives Launch Pad . . . 120
54. Hardware Perspective . . . 121
55. S70 Nodes In Perspectives . . . 122
56. Netfinity Nodes In Perspectives . . . 123
57. Powering Off Selected Nodes . . . 124
58. Power Off Mode Selection/Confirmation Window . . . 125
59. Adding A New Pane . . . 125
60. New Window With Two New Panes . . . 126
61. Set Monitoring Window . . . 127
62. Monitoring Host Responds On The Nodes Pane . . . 128
63. Frame Properties Window . . . 129
64. Changing System Partition In Perspectives . . . 129
65. Powering Off A Frame . . . 130
66. NTP Hierarchy . . . 132
67. DTS Time Clerks and Local Servers . . . 136
68. Local (Courier, Backup Courier, Noncourier) and Global DTS Servers . . . 137
69. Partners in a Third-Party Authentication . . . 149
70. Client's Authentication Request . . . 151
71. Authentication Server's Reply: TGT . . . 152
72. Client's Service Ticket Request . . . 153
73. Ticket Granting Service's Reply: Service Ticket . . . 154
74. Client's Application Service Request . . . 155
75. Application Client Components . . . 160
76. Application Server Components . . . 161
77. Kerberos Server Components . . . 163
78. Remote Shell Structure Before PSSP 3.1 . . . 174
79. Remote Shell Structure in PSSP 3.1 . . . 175
80. RSCT Infrastructure - "The Picture" . . . 187
81. TS and GS Interfaces . . . 189
82. TS Process Flow . . . 192
83. Group Services Structure . . . 194

84. GS Internals 196
85. "Group Awareness" of GS Daemons 197
86. EM Design 201
87. EM Client and Peer Communication 202
88. haemd Joining Group Services 205
89. SP Resource Monitors 206
90. Topology Table for Scenario 211
91. Switch Subsystem Component Architecture 216
92. Sample Topology File 219
93. Topology File Field Description 220
94. Topology File Information Mapped to Physical Switch Layout. 222
95. Typical Routing in the SP Switch 225
96. Logic Flow: Node Down on the css0 Interface. 228
97. Logic Flow: Node Up on host_responds 229
98. cssadm Daemon Response #1 230
99. cssadm Daemon Response #2 231
100.Console Dialog: Node Fence with Autojoin 233
101.Console Dialog #1: Node Reboot 234
102.Console Dialog #2: Node Reboot 234
103.Control Workstation Dialogue 246
104.Individual Node Error Log Entries 247
105.Extract from out.top File 249
106.Example of an 8_8 Configuration 251
107.Directory Structure for Topology Files 252
108./spdata Directory Structure 264
109.Multiple AIX And PSSP Levels 265
110.Site Environment Information SMIT Panel 268
111.SP Frame Information SMIT Panel 269
112.Non-SP Frame Information SMIT Panel 270
113.Successful Verification With spmon_ctest. 271
114.Supervisor Microcode Status 272
115.Setting Up The SP Ethernet Information 273
116.Acquiring The SP Ethernet Hardware Addresses 273
117.Setting Up Additional Adapter Information 275
118.Changing Default Hostname Information 275
119.Selecting The Authorization Method 276
120.Enabling Authentication Methods 277
121.Defining Dependent Node Information 278
122.Defining Dependent Node Adapter Information 278
123.Setting Up Volume Group Information 280
124.General Flow Of setup_server Command. 285
125.Annotating A Switch Topology File 286
126.Network Boot Option 288

127.Changing rootvg For Mirroring 290
128.Initiate Mirroring. 291
129.SP Perspectives Launch Pad 294
130.Event Perspective - Main Window. 298
131.Icon Colors for Event Definitions. 300
132.Sample Event Notification Log 301
133.Event Definition - Resource Variable Selection. 301
134.Event Definition - Condition Selection 302
135.Event Definition - Resource Variable Identification 303
136.Event Notification Log 303
137.PTX Functional Overview 313
138.PTPE Architecture. 314
139.PTPE Monitoring Hierarchy. 315
140.RS/6000 SP System with Defined Roles. 317
141.Site Environment- Accounting Setup. 321
142.SP Nodes - Accounting Setup. 322
143.Problem Management Subsystem Design 330
144.Problem Management Subsystem Daemons 331
145.SMIT Panel for Controlling SPUM Settings. 342
146.SMIT Panel for Adding SP Users 343
147.SMIT Panel for Deleting SP Users 344
148.SMIT Panel for Changing SP User Information. 345
149.Sample of an Entry in /etc/security/user 347
150.SMIT Panel for Setting a NIS Domain Name 351
151.SMIT Panel for Configuring a Master Server 352
152.SMIT Panel for Configuring a Slave Server 353
153.SMIT Panel for Configuring a NIS Client. 354
154.SMIT Panel for Managing NIS Maps. 355
155.SMIT Panel for Setting File Collections. 356
156.Summary of the File Collection Directory Structure 359
157.Sample /etc/auto.master File. 367
158.SMIT Panel for Turning On or Off the Automounter 369
159.Default /etc/auto.master File 370
160.Sample /etc/auto/map/auto.u File 371
161.Contents of an mksysb Backup Image 376
162.SMIT mksysb. 378
163.Conceptual Overview of NFS Mounting Process 388
164.Basic DFS Components 391
165.VSD Architecture. 395
166.Perspectives Panel for Designating VSD Nodes. 398
167.Perspectives Panel for Creating VSDs 399
168.Perspectives Panel for Defining VSDs 400
169.Perspectives Panel for Configuring VSDs. 401

Ex.1010 / Page 16 of 572

170.Perspectives Panel for Activating VSDs 402
171.VSD States and Commands 403
172.HSD Architecture 404
173.Summary of RVSD Failover Events 408
174.Summary of RVSD Recovery Events 409
175.GPFS Software Architecture 411
176.Sample Node List File 417
177.SMIT Panel for Configuring GPFS 418
178.Sample Output of /var/adm/ras/mmfs.log* 420
179.SMIT Panel for Creating Disk Descriptor File 422
180.SMIT Panel for Creating a GPFS File System 423
181.SMIT Panel for Mounting a File System 427
182.Avoiding Adapters and Disks as Single Point of Failure 440
183.SP Administrative Ethernet on Different Subnets 448
184.Starting Clinfo Daemon on Client System 451
185.Clstat Output Screen 452
186.POE Startup for Interactive Use 456
187.POE Startup under LoadLeveler 457
188.Thread Structure of a MPI-POE Task 468
189.MPI Structure 470
190.Structure of AIX Thread Library 471
191.Source File View with pedb 483
192.TotalView Process Window: poe 485
193.TotalView Process Window: Application Program 486
194.TotalView Root Window: Process Status 487
195.Xprofiler Main Window with Subprogram Call Tree 490
196.Vampir Global Timeline View 492
197.Vampir Message Identification 492
198.Vampir Average Message Rate Display 493
199.Vampir Message Length Distribution Histogram 493
200.Vampir Global Execution Statistics 494
201.Vampir Local Execution Statistics (User-Selected Routines) 495
202.Summary of How LoadLeveler Executes a Job 500
203.The xloadl GUI 505
204.Dialog Box for Writing a Job Command File 506
205.Dialog Box for Submitting a Job 507

Tables

1. Flynn's Taxonomy 12
2. Current Nodes Comparison 29
3. Supported Switch Adapters 71
4. RSCT Install Images 186
5. A Node's switch_responds State Table 232
6. IBM.PSSP.CSSlog.errlog is a Structured Byte String with These Fields . 242
7. AIX Error Log Entries 242
8. NIM Objects Classification 259
9. NIM Objects Classification Used By PSSP 260
10. PSSP Levels and Their Corresponding AIX Levels 261
11. Perfagent File Sets 262
12. Adapter Type and Transmit Queue Size Settings 263
13. Recommended Network Option Tunables 263
14. Supported External SSA Boot Devices 280
15. Supported External SCSI Boot Devices 281
16. SP Perspective Application Pathnames 294
17. Pre-Defined Event Definitions 298
18. Network Tuning Variables 306
19. VSD Fileset Names and Descriptions 396
20. RVSD Levels Supported by PSSP Levels 405
21. Supported Network Interface Cards for MCA Nodes 517
22. Supported SCSI and SSA Adapters for MCA Nodes 517
23. Other Supported Adapters for MCA Nodes 517
24. Supported Network Interface Cards for PCI Nodes 518
25. Supported SCSI and SSA Adapters for PCI Nodes 518
26. Other Supported Adapters for PCI Nodes 519

© Copyright IBM Corp. 1999

Preface

The RS/6000 SP is by far the most successful product within the RS/6000 family. With over 5,600 systems sold by year-end 1998, with an average of ten nodes each, the RS/6000 SP community has been increasing at a frantic rate for the past few years.

This redbook applies to PSSP Version 3, Release 1 for use with the AIX Operating System Version 4, Release 3, Modification 2. It describes the bits and pieces that make the RS/6000 SP a successful competitor in the difficult market of high-end computing.

Experienced SP professionals will find extremely useful and detailed information about most of the hardware and software components as well as recommendations for implementing real-life scenarios.

People new to the RS/6000 SP will find valuable and extensive introductory information to help them understand how this system works. This redbook provides solutions for a variety of business environments.

The book is organized in four parts following a bottom-up approach. It includes a section on history and evolution, where you will find the philosophy behind the design, as well as some thoughts about future development.

Each chapter is fairly independent so it can be used as reference material. Several examples and sample configurations are included as a way to provide a practical approach to the concepts being described.

In the last year of this millennium, the RS/6000 SP has proven to be prepared for the challenges of the next millennium. Are you prepared to take advantage of this?

The Team That Wrote This Redbook

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

Marcelo R. Barrios is a project leader at the International Technical Support Organization, Poughkeepsie Center. He has been with IBM for five years working in different areas related to RS/6000. Currently, he focuses on RS/6000 SP technology by writing redbooks and teaching IBM classes worldwide.

© Copyright IBM Corp. 1999

Mario Bono is an Advisory I/T Specialist working in the Business Servers Technology Group at IBM Australia. In this position he has been working with AIX and RS/6000 SP products since 1994, in both consulting and services roles throughout Australia and South-East Asia. He joined IBM in 1986 and worked in Large Systems (MVS and VM) as a Systems Engineer and later in the Australian Programming Centre as a Technical Project Leader. He is a certified RS/6000 SP Specialist, and holds a bachelor's degree in Electrical Engineering from the University of Queensland, and a master's degree in Technology Management from Griffith University.

Michael Hennecke is a certified RS/6000 SP specialist at the Computing Center of the University of Karlsruhe, Germany. He is responsible for the system management of Karlsruhe's 256-node SP. He has eight years of experience with parallel computing, including four years on the RS/6000 SP. Michael has also been teaching university classes on parallel programming for the last four years. He holds a degree in Physics from the University of Bochum (RUB), Germany.

Teerachai Supapunpinyo is an Advisory Sales Specialist in Business Server Group at IBM Thailand. He has been with IBM for seven years working in different areas. He started his career with IBM as a Systems Engineer responsible for RS/6000 and AIX support to public sector accounts for two years. He moved into the marketing field in 1994 as a Client Representative responsible for government, telecommunication and utility accounts for three years. Currently he focuses on RS/6000 for large customers in Thailand. Teerachai holds a degree in Computer Engineering from Chulalongkorn University and an M.B.A. from Thammasat University.

Bernard Woo, P.Eng., is an IBM I/T Support Specialist who has been working in the Canadian RS/6000 Customer Assist Centre (CAC) for three years. For the past two years, he has specialized in supporting the SP. He holds a degree in Electrical Engineering from the University of Waterloo in Canada.

Steven Yap has been working in IBM Singapore as an IT specialist for the past six years. His job responsibilities include account support, help desk, project planning and implementation, and on-site problem determination for both software and hardware in all areas of RS/6000 systems. A certified RS/6000 SP specialist, his main focus is on the RS/6000 SP and HACMP. He holds a degree in Electrical Engineering from the National University of Singapore.

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

Ex.1010 / Page 22 of 572

Hans-Juergen Kitzhoefer
ITSO Poughkeepsie

Klaus Gottschalk
IBM Germany

Frauke Boesert
Klaus Geers
Horst Gernert
Roland Laifer
Reinhard Strebler
University of Karlsruhe, Germany

Mark Atkins
Joe Banas
Pat Caffrey
Lisa Case-Hook
Michael K Coffey
Richard Coppinger
Chris DeRobertis
Ron Goering
Jan Ranck-Gustafson
Barbara Heldke
Nancy Mroz
Norman Nott
Robert Palmer
Larry Parker
Kevin Redman
Richard Treumann
IBM Poughkeepsie

John Maddalozzo
IBM Austin

Scott Vetter
ITSO Austin

Comments Welcome

Your comments are important to us!

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

- Fax the evaluation form found in “ITSO Redbook Evaluation” on page 547 to the fax number shown on the form.
- Use the electronic evaluation form found on the Redbooks Web sites:

For Internet users	`http://www.redbooks.ibm.com`
For IBM Intranet users	`http://w3.itso.ibm.com`

- Send us a note at the following address:

  `redbook@us.ibm.com`

Ex.1010 / Page 24 of 572

Part 1. The Big Picture

© Copyright IBM Corp. 1999

Chapter 1. History and Design Philosophy

Since its introduction to the marketplace five and a half years ago, the Scalable Powerparallel (SP) supercomputer has built up an impressive resume:

- It has been installed in over 70% of the US Fortune 500 companies, as well as in scientific and technical institutions worldwide.
- It has beaten humanity's chess champion, Gary Kasparov.
- It has expanded from 0 to approximately 28% of IBM's total UNIX-based system revenue.
- It has been selected by the US Department of Energy for the Accelerated Strategic Computing Initiative (ASCI) project, which will result in the creation of the most powerful computer in the world.
- It has repeatedly set world records for Internet Web-serving capability, most recently at the Nagano Winter Olympics.

In this chapter we discuss the types of computing problems that prompted IBM to create the SP. We review the history of the SP's development, and present its key design points.

1.1 Ultra-Large Computing Problems

The development pace and raw capabilities of computer systems over the last forty years astonish most of us; however, the progress of computer technology is no match for our ability to generate new, bigger, more complex problems for computers to solve. The body of human knowledge and creation continues to grow exponentially, as does the planet-wide infrastructure - the Internet - to share it all. The globalization of world economies has forced businesses to seek any competitive advantage, such as extracting business intelligence from the vast data reservoir of their operational transactions. Consider the data processing requirements of the following problems:

- Accurately simulate the explosion of a nuclear weapon
- Rapidly analyze 3D seismic surveys, each involving 1 to 10 terabytes of data, for a resource exploration client
- Integrate all core business transactions - financial, materials management, inventory, sales, and so forth - of a global enterprise comprising tens of thousands of users
- Analyze 200 million chess board positions *each second* to be competitive with human grandmasters

© Copyright IBM Corp. 1999

Talk about job security for a well-designed ultra-large computer!

Traditionally, *supercomputers* have solved these types of problems. A supercomputer is typically described as having an immensely powerful, expensive, esoteric, complex, and delicate “personality”. It is usually a single, monolithic system, managed by graduate-level scientists and researchers, and somewhat of an elite tool for academia, government, and perhaps some large corporations.

Today, clustering of computers is emerging as a technological solution for these ultra-large scale problems. Although the SP is central to IBM’s mid-range clustering development, it was conceived in the supercomputer era.

1.2 Origins of the SP

In the late 1980s, IBM set out to build a supercomputer for large, technical customers. The High Performance Supercomputer Systems Development Laboratory (HPSSDL) was formed within IBM’s Large Systems Division in Kingston and Poughkeepsie, New York. HPSSDL intended to create a supercomputer with a more familiar personality, one based on widespread, non-exotic technology. Not surprisingly, IBM’s ES/9000 mainframe vector processor architecture initially provided the basis for development. This architecture eventually proved to be too limiting. Implementation aspects such as guaranteed instruction execution order, special interrupt handling, and a unique floating point representation (incompatible with the emerging IEEE-based standard) restricted the speed and interoperability of the design.

In 1990 IBM’s advanced workstation division in Austin, Texas introduced the RISCSystem/6000 (RS/6000) family of UNIX-based workstations and servers. These early RS/6000 machines boasted stellar floating point performance for their time, owing to the strength of the Performance Optimization with Enhanced RISC (POWER) CPU architecture. The fact that they ran UNIX was of great interest to HPSSDL, as UNIX was entrenched in their target market - large scientific and technical customers. HPSSDL, which was at an impasse with mainframe processor technology, experimented with off-the-shelf RS/6000 machines by adding ESCON adapters and interconnecting them with an ESCON Director. The RS/6000 machines were repackaged as nodes and placed in frames. Only five of the large, sheet metal drawers for the nodes could be placed in one frame. The drawers were judged to be too small to contain a person, so they were nicknamed “dog coffins”.

Ex.1010 / Page 28 of 572

As HPSSDL was creating dog coffins, an IBM research group in Yorktown, New York was working on a high-speed switch code-named Vulcan. Yet another group in Yorktown was trying to solve both the problem of deploying these hot new RS/6000 workstations to the desktops of IBM workers, and the system management headaches that come with LAN administration. This group developed a frame that could house 16 RS/6000 machines, as well as management software called Agora to create a true “LAN-in-a-can”.

In December 1991, these independent efforts began to come together. HPSSDL was reorganized and renamed to HPSSL (the “Development” part of the name was dropped) under the leadership of Irving Wladawsky-Berger. (Mr. Wladawsky-Berger went on to become head of the RS/6000 Division, and currently is the General Manager of the Internet Division.) HPSSL’s mission was to ship a product in 12 months. Designing a system from scratch was out of the question given the time constraint, so a task force was created, which selected the necessary system components from available IBM technology. The RS/6000 Model 370 furnished the node technology. The Yorktown LAN consolidators provided their knowledge and experience in packaging the nodes. The Vulcan switch, now code-named Envoy (the origin of the “E” commands for the switch), was chosen over the ESCON interconnect, which was experiencing problems with excessive latency. Work from the ESCON interconnect experiment formed the basis for the first iteration of the Vulcan switch software. The total product was introduced to the marketplace as the SP1.

In September, 1993, Argonne National Laboratories in Argonne, Illinois received shipment of the first SP1, a 128-node system. Cornell University in Ithaca, New York, bought a 512-node system shortly thereafter. Next came the petroleum industry. By the end of the year, 72 systems had been installed around the world, all with scientific and technical customers.

Initially, IBM had no intention of positioning the SP1 in the commercial marketplace, but commercial customers started knocking on IBM’s door. In the early 1990’s, the death of the mainframe was accepted as conventional wisdom. Therefore, many large commercial enterprises were looking for alternatives to the mainframe in order to deploy new applications. IBM formed an application solutions group for the SP1, which, among other things, ported a parallel version of Oracle’s database to the SP1. In 1994, SP development absorbed personnel from the discontinued AIX/ESA product, who bolstered the system’s manageability and helped spawn the Parallel System Support Program (PSSP, described in Part 2, “System Implementation” on page 17). By the end of 1994, the commercial segment accounted for 40% of all installed SPs. By the end of 1996, the share climbed to 70%.

The SP2 was announced in 1994. It incorporated new node types from Austin and a faster switch, code-named Trailblazer (the origin of the tb2 and tb3 nomenclature of the switch adapters). The SP2 had moved out from under the umbrella of the Large Systems Division and was its own little enterprise within IBM. SP2 sales were strong: 352 systems were installed by the end of 1994, and 1,023 by the end of 1995.

In 1996, the SP2 was renamed to simply the SP and formally became a product of the RS/6000 Division. It represents the high-end of the RS/6000 family. IBM secured a number of large SP contracts, of particular note the ASCI project of the US Department of Energy. These contracts, coupled with the broad marketplace acceptance of the product, have fuelled SP development. In 1996, IBM introduced a faster version of the Trailblazer switch, more than doubling the bandwidth of its predecessor, new nodes, including Symmetric Multiprocessor (SMP) versions, and more robust and functional PSSP software. As of the end of 1997, there were over 3770 SP systems installed throughout the world. From 1994 to 1997, the SP install base has been growing at an annual rate of 169%.

1.3 High-Level System Characteristics

The marketplace acceptance of the SP rests on the underlying design philosophy of the machine. From the product's inception, IBM strove to create a supercomputer that had the following high-level system characteristics:

- Scalability to ultra-large sizes
- Use of mainstream architectures and technologies
- Flexibility
- Manageability

We describe these characteristics in this section.

1.3.1 Scalability

The SP efficiently scales in all aspects, including:

- Hardware and software components, to deliver predictable increments of performance and capacity
- Network bandwidth, both inside and outside the machine
- Systems management, to preserve the investment in tools, processes, and skills as the system grows

Importantly, the SP can scale both up and down. It can be subdivided, either logically or physically, into smaller SPs. In addition, scaling is a consistent process regardless of the initial size of an SP implementation. A customer

Ex.1010 / Page 30 of 572

with a one-node SP can grow to 512 nodes the same way a customer with a 400-node system does. The simple erector set style of expanding the system also suits many customers' procurement environments. A researcher or department with some funds could easily add a node or two or upgrade the memory in an existing node, without incurring a large upgrade or box swap cost.

1.3.2 Use of Mainstream Architectures and Technologies

1.3.2.1 Architectures

Unlike many other supercomputers, the SP does not implement exotic, special-purpose architectures. It supports the key programming models in both the technical and commercial areas. Enablers, tools, and skills to program the SP are widely available.

1.3.2.2 Technologies

The SP relies heavily on mainstream hardware and software components. This bolsters the SP product business case by minimizing development costs and integration efforts.

The system runs Advanced Interactive Executive (AIX), IBM's standards-compliant implementation of UNIX, and as a result inherits a portfolio of over 10,000 off-the-shelf applications and enablers. The hardware building blocks of the system are simply RS/6000 machines. The SP attaches the same disk and tape subsystems as any RS/6000 machine, and is managed with the same enterprise-level tools, such as Tivoli and Adstar Distributed Storage Manager (ADSM).

1.3.2.3 High-End Technology Flows into Volume Marketplace

The SP's affinity with the mainstream RS/6000-AIX marketplace allows innovations from the ultra-large SP implementations to flow into the mainstream volume marketplace. A prime example of this is the U.S. Department of Energy ASCI project: switch, node and software technology that IBM has committed to deliver in the ASCI contract will be available to *all* SP customers a year or two later.

1.3.2.4 Marketplace Volumes Fuel High-End Development

The marketplace success of SP technology, developed to meet the requirements of the high-end customers, generates the required revenue to sustain high-end development.

1.3.3 Flexibility

The SP is a multipurpose platform addressing a broad range of customer requirements. Not only can it help solve problems across a range of industries, but a single SP can also be used for a variety of applications within a single organization. A customer can easily configure an SP to support diverse workloads such as serial batch jobs, parallel jobs, interactive applications, communication gateways and routing, and serving of all kinds.

1.3.4 Manageability

The SP is production-worthy. It has good reliability, availability and serviceability, allowing it to host mission-critical applications. It has a single point-of-control and is managed with consistent tools, processes and skills regardless of system size.

1.4 Future Directions for the RS/6000 SP

As the SP business expands in many different segments, SP systems are increasingly being used for business critical applications. Robust RAS (Reliability, Availability, and Serviceability) characteristics are essential attributes in these environments. The SP hardware and software has improved the system RAS capabilities in a number of areas:

- Improved hardware component reliability
- Improved hardware component testing processes
- Development of software services for a scalable availability infrastructure
- Improved error logging
- Improved switch error handling

The RS/6000 Cluster Technology (RSCT) provides mechanisms for scalable, reliable detection and notification of hardware and software events (Event Management), and efficient coordination and notification among instances of a distributed or parallel subsystem (Group Services). These services are built upon a reliable messaging and heartbeating framework that guarantees delivery of messages, and detects and routes around node and adapter failures. This cluster technology is used by SP subsystems as well as middleware and applications.

This cluster technology has been used outside the SP as part of the HACMP/ES product. HACMP provides high availability through failure detection and failover for any RS/6000 servers or SP nodes. The HACMP/ES product extends the capabilities of HACMP to 32 node clusters. This is an example where technology developed on the SP is extended to provide customer value throughout the RS/6000 product line.

Ex.1010 / Page 32 of 572

Another example of the blending of RS and SP is the incorporation of the SP-attached S70 and S7A servers as part of the SP system. The S70 and S7A are large SMP systems. Their inclusion as part of the SP allows the benefits of SP Switch attachment and PSSP management to be extended to these systems that are physically too large to be packaged in SP frames. Including these systems as part of the SP brings the benefits of large capacity to run the largest single system jobs to SP customers.

Incorporating the SP availability infrastructure within HACMP, and including large SMP servers as part of the SP system umbrella, are examples of the merging and cross-pollination between SP and the rest of the RS/6000 servers. Currently under development is additional clustering work that will bring the benefits of PSSP management and HACMP availability to all RS servers. Stand-alone servers, clusters of servers and SP nodes, and SP systems will have consistent capabilities for availability, resource monitoring and control, workload balancing and easy-to-use system management. Customers can cluster together SP nodes and stand-alone servers in configurations that make the most sense for their business problems and computing environment, and get the increased value and benefit across all RS/6000 systems.

Chapter 2. RS/6000 SP Architecture

The ultimate question we want to answer in this chapter is: how does the RS/6000 SP fit in the broad spectrum of computer architectures?

If you want a straight answer, then you need to read the rest of the book to understand how hard is to classify the SP in a single category.

Moreover, classification of computer architectures can be done in many ways. Michael Flynn's taxonomy classifies computers based on the way streams of instructions interact with streams of data. Based on this classification, there can be four possible types of computers, as shown in Table 1 on page 12.

This classification is widely accepted and we can find practical implementations for most of these computer types. As a matter of fact, the RS/6000 SP could be classified in more than one of these categories.

Modern computer architectures, in general, cross many of the lines that divide each category, thus making computer classification an "art".

Although Single Instruction Single Data (SISD) architecture is part of what the RS/6000 SP can offer, we concentrate on the multiprocessor aspect of the system, adding notes along the way when an explicit discussion of uniprocessor systems is needed.

We understand that multiple streams of instructions are independent and executed by independent executing units (processors). A MIMD machine then is a multiprocessor machine by definition.

Within multiprocessor systems, we can further divide the classification, shown in Table 1 on page 12, based on the way how data is shared between processors. At this point we have two categories: Shared Nothing and Shared Data. We could have said "shared memory" and that term would probably ring a bell, but sharing data can be achieved at memory level as well as at disk level, so a more generic term is useful to avoid being too specific.

The term *shared nothing*, in this context, refers to data access. In this type of architecture, processors do not share a common repository for data (memory or disk), so data sharing has to be carried out through messages. Systems using this architecture are also called *message-based* systems.

In contrast, shared data architectures have processing units sharing data in one way or another. One of the most popular shared data multiprocessor implementations is Symmetric Multiprocessing (SMP). In this architecture, all

© Copyright IBM Corp. 1999

processors share a common memory bank and I/O unit(s). All the processors, memory and I/O units are connected through a system bus (there can be multiple system buses).

The advantage of an SMP architecture is having minimum impact to applications while taking advantage of parallel processing. Applications developed for uniprocessor systems can “easily” run on SMP machines while they using a single processor. However, the system is able to handle multiple applications like this in parallel, which will not necessarily benefit the particular application, but the system in its entirety.

Inherent in SMP architectures is the concept of *threads*. Streams of instructions can be broken into several “portions” that are relatively independent and executed in different processors, thus “speeding up” the application as a whole. This is in theory, because in reality the effectiveness of multi-threading an application will ultimately depend on the interaction between threads and data.

One limitation though, in SMP architectures, is scalability. While adding processors to an SMP system may improve its performance (by adding more executing units), the shared elements within this architecture will slow down the system to a point where adding more processors will prove to be counterproductive.

Table 1. Flynn's Taxonomy

		No. of Data Streams	
		Single	Multiple
Number of Instruction Streams	Single	**SISD**	**SIMD**
	Multiple	**SIMD**	**MIMD**

However, this self-imposed limitation has been overcome year after year, thus increasing the practical number of processors in an SMP machine to numbers we would not have imagined a few years ago.

Figure 1 on page 13 shows how MIMD architectures can be further classified based on data access.

Shared nothing systems are also refer as *message-based* or *message-passing* systems. In such systems, communication is completed when data has been received by the target node.

If you analyze the RS/6000 SP from a system point of view, you see that processor nodes communicate each other through communication networks

(Ethernet, SP Switch, or any other supported network) so, by definition, the SP architecture is a Shared Nothing architecture. However, each node could be classified as a SISD node (old uniprocessor nodes), or a MIMD node (such as SMP nodes). Furthermore, the RS/6000 SP flexibility allows to have clusters of nodes running within the RS/6000 SP umbrella.

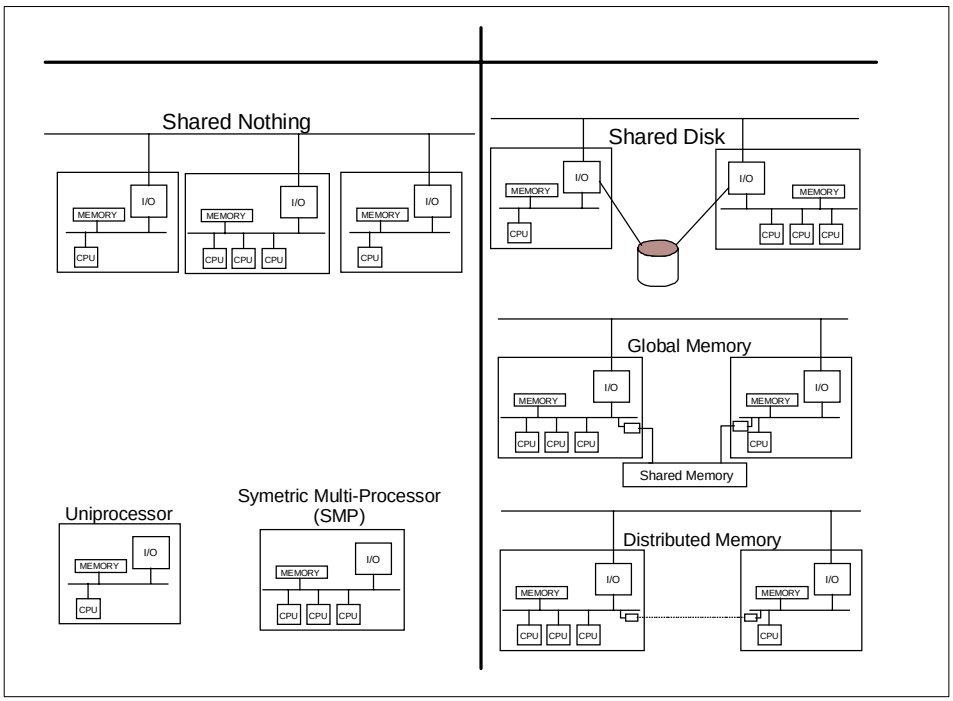


Figure 1. MIMD System's Classification

Besides the fact that we can use the RS/6000 SP nodes as standalone machines for running "serial" and independent applications in what is called "server consolidation", the RS/6000 SP can be also viewed as a cluster or even as a parallel machine (a massive parallel machine).

2.1 The SP as a Cluster

The concept of a cluster within the RS/6000 SP was initially developed as a *system partition*. These system partitions were intended for switch isolation and software coexistence. The main idea was to divide the SP into several functional SPs, controlled by the same control workstation but having different software levels and system management realms.

Currently, partitions have evolved to what are called “domains”. This is true for some of the RS/6000 SP software components, but not for all of them. Some subsystems remain global to the system even after they have been “partitioned”.

The term “domain” is replacing the concept of partition as the logical subdivision of RS/6000 SP systems. At the core of partitioning is the System Data Repository and the partition-sensitive subsystems which include the RS/6000 Cluster Technology (RSCT).

At the high level, applications can take advantage of the RS/6000 Cluster Technology (described in Chapter 8, “RS/6000 Cluster Technology” on page 185) by using the notification and coordination services provides by RSCT. At the system level, administrators can take advantage of the high availability functions provided by the basic PSSP components as well as the Enhanced Scalability version of the High Availability Cluster Multiprocessing (HACMP ES, described in Chapter 15, “High Availability” on page 433).

The RS/6000 SP is, by definition, a cluster. If you consider each node as an independent machine, running its own instance of the operating system, having its own address space and data access but tied together by a set of software layers, then the cluster classification comes natural.

2.2 The SP as a Parallel Machine

The RS/6000 SP provides several facilities to develop and run parallel applications. In this redbook, we dedicate an entire chapter to parallel programing (Chapter 16, “Parallel Programming” on page 453). However, an application needs to be especially designed and developed to take full advantage of the true parallelism that the RS/6000 SP can offer.

The SP Switch network provides redundant high-bandwidth low-latency paths between nodes. Applications can use these paths to share data and state information at high speed.

Parallelism is achieved by using multiple nodes to execute a job where each task within this job can be allocated to a different node. The RS/6000 SP provides facilities for data sharing such as the General Parallel File System (GPFS) discussed in Chapter 14, “Data Storage” on page 385 and intertask communication such as Message Passing Interface (MPI) and other protocols described in Chapter 16, “Parallel Programming” on page 453. You can even add workload balancing and job management to the equation by using LoadLeveler described in Chapter 17, “LoadLeveler” on page 497.

Ex.1010 / Page 38 of 572

One of the advantages that the RS/6000 SP architecture offers to parallel application is scalability. Scalability can be measured in two ways: vertical and horizontal. *Vertical scalability* is the ability to increased the processing power of a single node. Assuming that a job will execute in multiple nodes, vertical scalability benefits the application by increasing the processing power of each node. This is the easiest way to improve performance of an application because it does not require any changes to the application itself.

Horizontal scalability is the ability to add more processing power to the application by adding additional nodes. This is the most natural way to scale a parallel application but it may involve modification to the application to take full advantage of the additional processing nodes.

Part 2. System Implementation

© Copyright IBM Corp. 1999

Chapter 3. Hardware Components

This chapter provides information about the hardware components of the RS/6000 SP.

The basic components of the RS/6000 SP are:

- The frame with its integral power supply
- Processor nodes
- Optional dependent nodes that serve a specific function such as high-speed network connections
- Optional SP Switch and Switch-8 to expand your system
- Control workstation (a high-availability option is also available)
- Network connectivity adapters and peripheral devices such as tape and disk drives

These components connect to your existing computer network through a local area network (LAN), making the RS/6000 SP system accessible from any network-attached workstation.

Figure 2 on page 20 shows a sample of RS/6000 SP components.

© Copyright IBM Corp. 1999

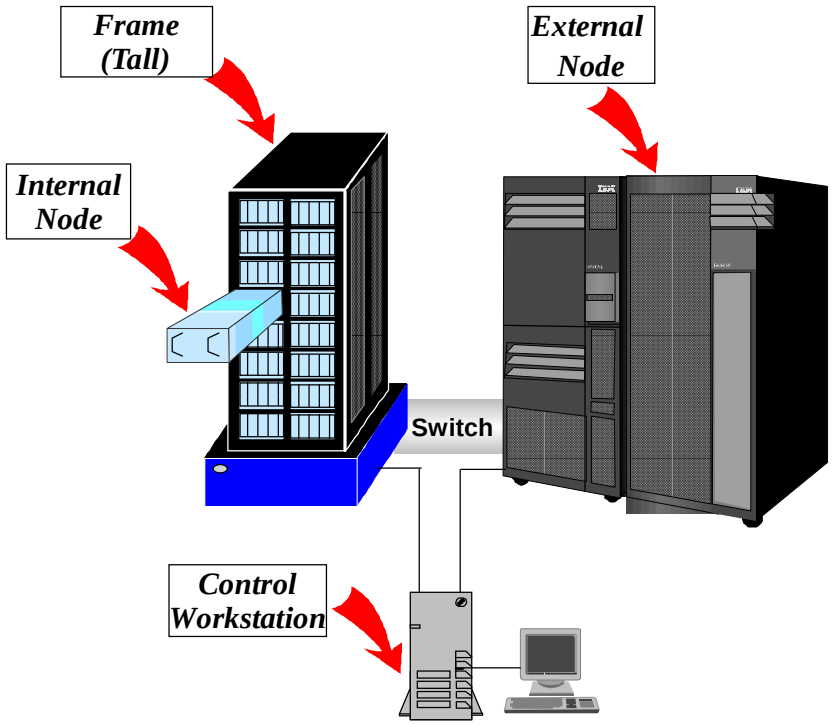


Figure 2. Sample RS/6000 SP with External Node

3.1 Frames

The building block of RS/6000 SP is the “frame”. There are two sizes: the Tall frame (75.8in high) and the Short frame (49in high). RS/6000 SP internal nodes are mounted in either a Tall or Short frame. A Tall frame has eight drawers, while a Short frame has four drawers. Each drawer is further divided into two slots. A Thin node occupies one slot; a Wide node occupies one drawer (two slots) and a High node occupies two drawers (four slots). An internal power supply is included with each frame. Frames get equipped with optional processor nodes and switches.

There are five current types of frames:

- The Tall model frame
- The Short model frame
- The Tall expansion frame
- The Short expansion frame
- The SP Switch frame

The model frame is always the first frame in an SP system. It designates the type or “model class” of your SP system. The optional model types are either a Tall frame system or a Short frame system. Other frames that you connect to the model frame are known as expansion frames. The SP Switch frame is used to host switches or Intermediate Switch Boards (ISBs), which are described later in this chapter. This special type of frame can host up to eight switch boards.

Since the original RS/6000 SP product was made available in 1993, there have been a number of model and frame configurations. The frame and the first node in the frame were tied together, forming a model. Each configuration was based on the frame type and the kind of node installed in the first slot. This led to an increasing number of possible prepackaged configurations as more nodes became available.

The introduction of a new Tall frame in 1998 was the first attempt to simplify the way frames and the nodes inside are configured. This new frame replaces the old frames. The most noticeable difference between the new and old frame is the power supply size. Also, the new Tall frame is shorter and deeper than the old Tall frame. With the new offering, IBM simplified the SP frame options by telecopying the imbedded node from the frame offering. Therefore, when you order a frame, all you receive is a frame with the power supply units and a power cord. All nodes, switches, and other auxiliary equipment are ordered separately.

All new designs are completely compatible with all valid SP configurations using older equipment. Also, all new nodes can be installed in any existing SP frame provided that the required power supply upgrades have been implemented in that frame.

Note: Tall frames and Short frames cannot be mixed in an SP system.

3.1.1 Tall Frames

The Tall model frame (model 550) and the Tall expansion frame (feature code #1550) each have eight drawers, which hold internal nodes and an optional switch board. Depending on the type of node selected, an SP Tall frame can contain up to a maximum of 16 Thin nodes, 8 Wide nodes or 4 High nodes. Node types may be mixed in a system and scaled up to 128 nodes (512 by special request).

3.1.2 Short Frames

The Short model frame (model 500) and the Short expansion frame (feature code #1500) each have four drawers, which hold internal nodes and an

optional switch board. Depending on the type of node selected, an SP Short frame can contain up to a maximum of 8 Thin nodes, 4 Wide nodes or 2 High nodes. Also, node types can be mixed and scaled up to only 8 nodes. Therefore, for a large configuration or high scalability, Tall frames are recommended.

Only the Short model frame can be equipped with a switch board. The Short expansion frame cannot hold a switch board, but nodes in the expansion frame can share unused switch ports in the model frame.

Figure 3 illustrates Short frame components from a front view.

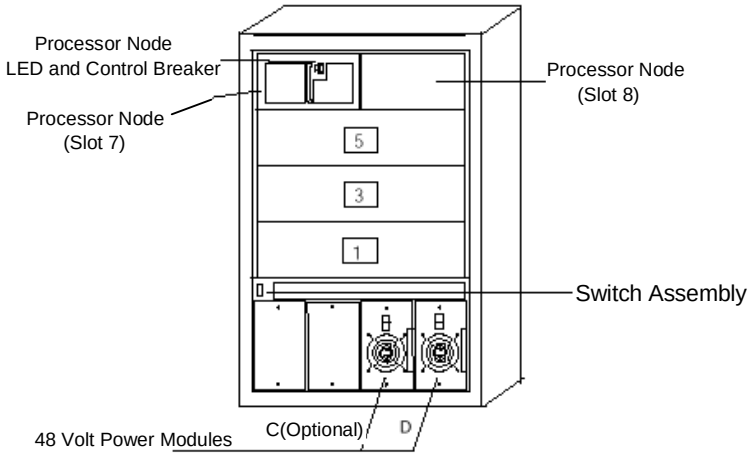


Figure 3. Front View of Short Components

3.1.3 SP Switch Frames

The SP Switch frame is defined as a base offering Tall frame equipped with either four or eight Intermediate Switch Boards (ISBs). This frame does not contain processor nodes. It is used to connect model frames and switched expansion frames that have maximized the capacity of their integral switch boards. Switch frames can only be connected within the local SP system.

The base level SP Switch frame (feature code #2031) contains four ISBs. An SP Switch frame with four ISBs will support up to 128 nodes. The base level SP Switch frame can also be configured into systems with fewer than 65 nodes. In this environment, the SP switch frame will greatly simplify future

system growth. Figure 4 on page 23 shows an SP Switch frame with eight ISBs.

Note: The SP Switch frame is required when the sixth SP Switch board is added to the system, and is a mandatory prerequisite for all large scale systems.

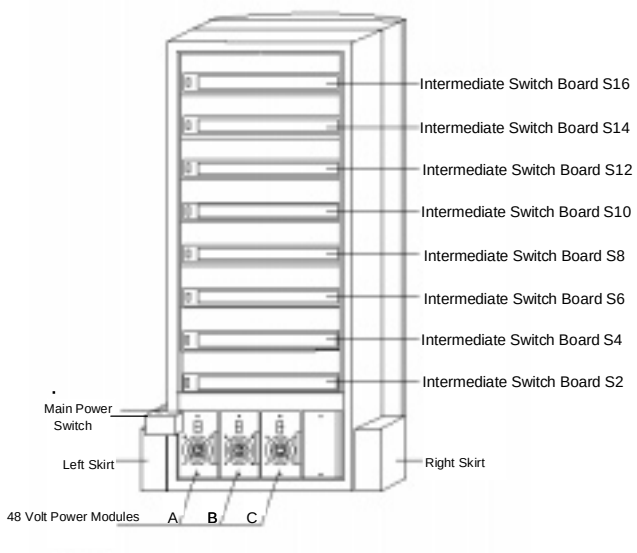


Figure 4. SP Switch Frame with 8 Intermediate Switch Boards (ISBs)

3.1.4 Power Supplies

Tall frames come equipped with redundant (N+1) power supplies; if one power supply fails, another takes over. Redundant power is an option on Short frames (feature code #1213). These power supplies are self-regulating units. Power units with the N+1 feature are designed for concurrent maintenance; if a power unit fails, it can be removed and repaired without interrupting running processes on the nodes.

A Tall frame has four power supplies. In a fully populated frame, the frame can operate with only three power supplies (N+1). Short frames come with two power supplies and a third, optional one, for N+1 support.

Figure 5 on page 24 illustrates Tall frame components from front and rear views.

The power consumption depends on the number of nodes installed in the frame. For details refer to *IBM RS/6000 SP: Planning Volume 1, Hardware and Physical Environment*, GA22-7280.

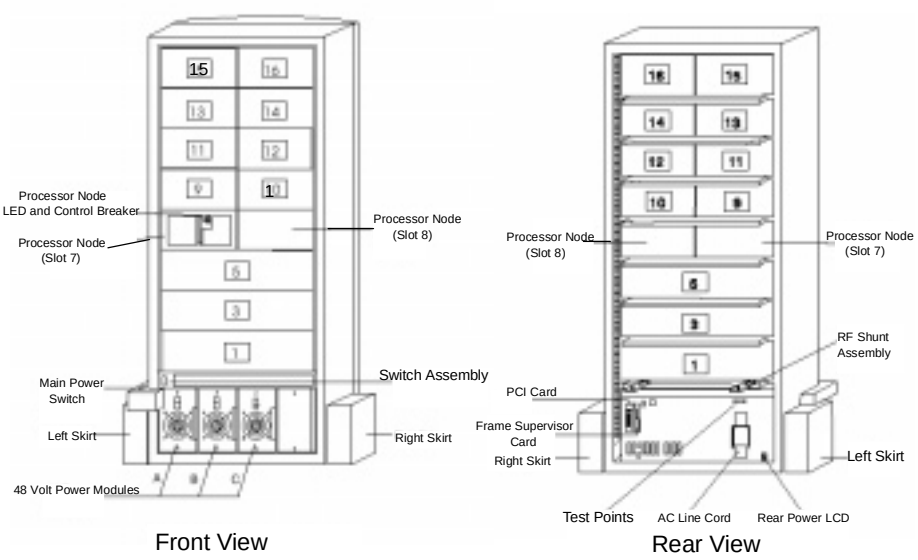


Figure 5. Front and Rear Views of Tall Frame Components

3.1.5 Hardware Control and Supervision

Each frame (Tall and Short) has a supervisor card. This supervisor card connects to the Control Workstation via a serial link as shown in Figure 6 on page 25.

The supervisor subsystem consists of the following components:

- Node supervisor card (one per processor node)
- Switch supervisor card (one per switch assembly)
- Internal cable (one per Thin processor node or switch assembly)
- Supervisor bus card (one per Thin processor node or switch assembly)
- Frame supervisor card
- Serial cable (RS-232)
- SAMI cable

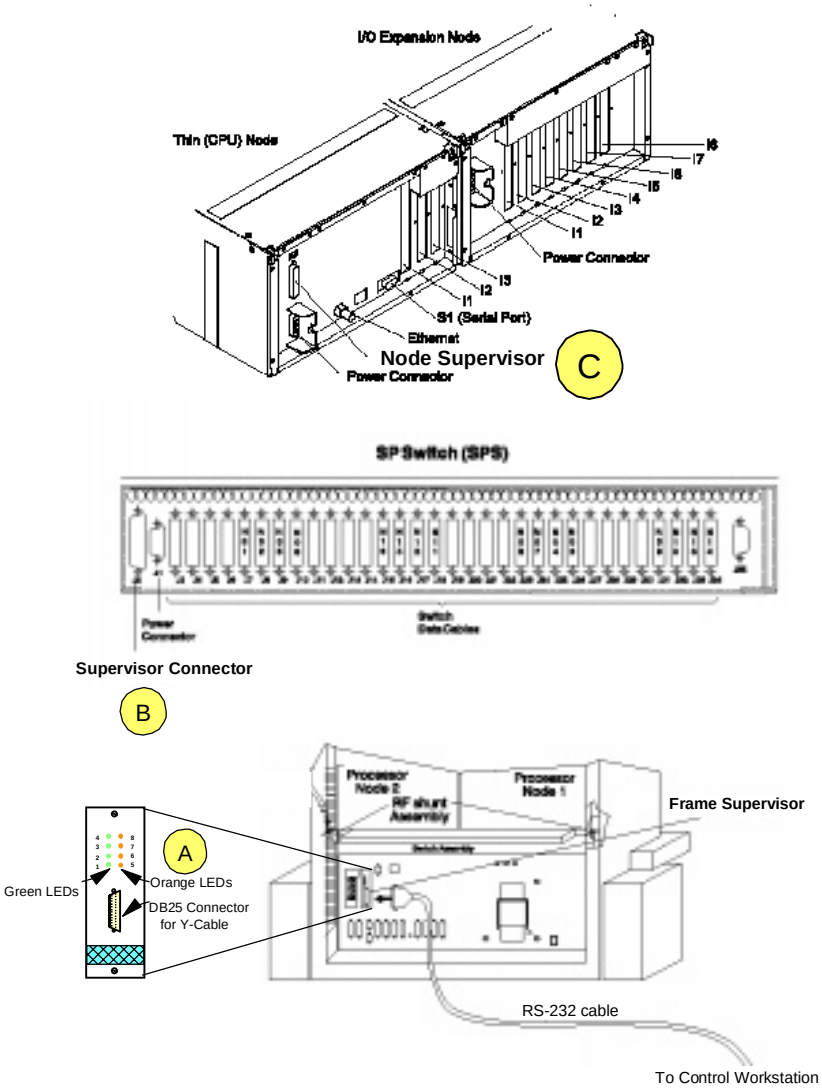


Figure 6. Frame Supervisor Attachment

There is a cable that connects from the frame supervisor card (position A) to the switch supervisor card (position B) on the SP Switch or the SP Switch-8 boards and to the node supervisor card (position C) of every node in the frame. Therefore, the control workstation can manage and monitor frames, switches and all in-frame nodes.

3.2 Standard Nodes

The basic RS/6000 SP building block is the server node or standard node. Each node is a complete server system, comprising processors, memory, internal disk drives, expansion slots, and its own copy of the AIX operating system. The basic technology is shared with standard RS/6000 workstations and servers, but differences exist that allow nodes to be centrally managed. There is no special version of AIX for each node. The same version runs on all RS/6000 systems.

Standard nodes can be classified as those which are inside the RS/6000 SP frame and those which are not.

3.2.1 Internal Nodes

Internal nodes can be classified, based on their physical size, as Thin, Wide and High nodes. A Thin node occupies one slot of an SP frame, while a Wide node occupies one full drawer of an SP frame, and a High node occupies two full drawers (four slots).

Since 1993, when IBM announced the RS/6000 SP, there have been 14 internal node types, excluding some special "on request" node types. There are five most current nodes: 160 MHz Thin P2SC node, 332 MHz SMP Thin node, 332 MHz SMP Wide node, POWER3 SMP Thin node and POWER3 SMP Wide node. Only the 160 MHz Thin P2SC node utilizes Micro Channel Architecture (MCA) bus architecture, while the others use PCI bus architecture.

160 MHz Thin P2SC Nodes

This node is based on the POWER2 Super Chip (P2SC) implementation of the POWER architecture. Each node contains a 160 MHz P2SC processor combining IBM RISC microprocessor technology and the IBM implementation of the UNIX operating system, AIX. The standard memory in each node is 64 MB, expandable to 1 GB maximum. The minimum internal disk storage in each node is 4.5 GB, expandable to 18.2 GB. Each node has two disk bays, four Micro Channel slots, and integrated SCSI-2 Fast/Wide and Ethernet (10 Mbps) adapters. This node is equivalent to the RS/6000 standalone model 7012-397.

332 MHz SMP Thin Nodes

This node is the first PCI architecture bus node of the RS/6000 SP. Each node has two or four PowerPC 604e processors running at 332 MHz clock

Ex.1010 / Page 50 of 572

cycle, two memory slots with 256 MB, expandable to 3 GB, of memory, and integrated Ethernet (10 Mbps) and SCSI-2 Fast/Wide I/O adapters to maximize the number of slots available for application use. This Thin node has two internal disk bays with a maximum of 18.2 GB (mirror), and two PCI I/O expansion slots (32-bit). The 332 MHz SMP Thin node can be upgraded to the 332 MHz SMP Wide node.

332 MHz SMP Wide Nodes

The 332 MHz SMP Wide node is a 332 MHz SMP Thin node combined with additional disk bays and PCI expansion slots. This Wide node has four internal disk bays with a maximum of 36.4 GB (mirror), and 10 PCI I/O expansion slots (three 64-bit, seven 32-bit). Both 332 MHz SMP Thin and Wide nodes are based on the same technology as the RS/6000 model H50 and have been known as the “Silver” nodes. Figure 7 shows a 332 MHz SMP node component diagram.

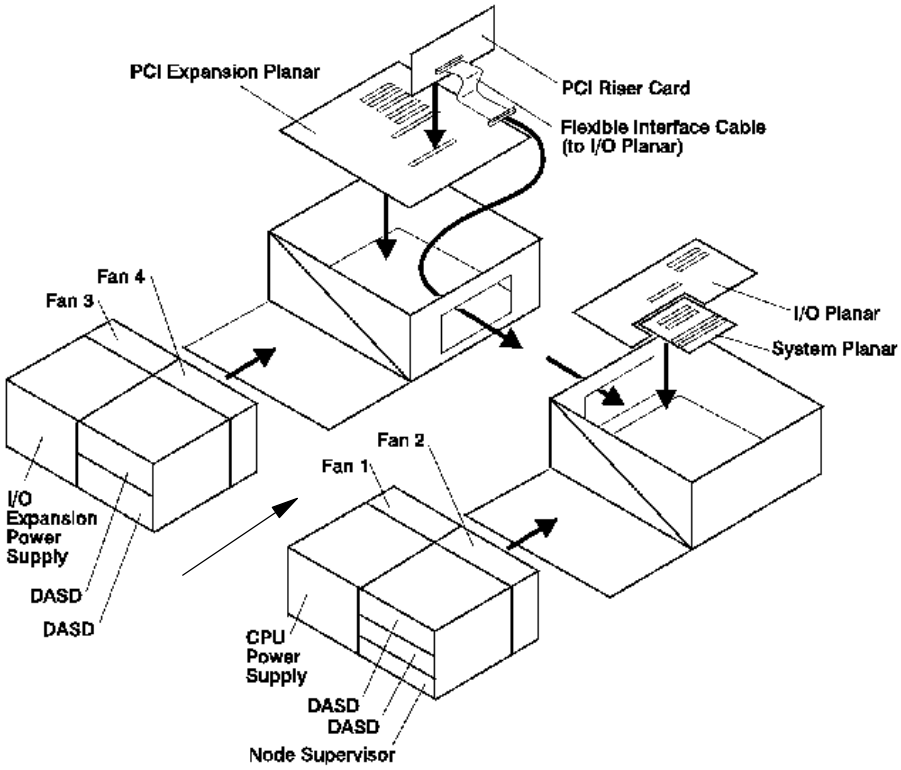


Figure 7. 332 MHz SMP Node Component Diagram

POWER3 SMP Thin Nodes

This node is the first 64-bit internal processor node of the RS/6000 SP. Each node has a 1- or 2-way (within two processor cards) configuration utilizing a 64-bit 200 MHz POWER3 processor, with a 4 MB Level 2 (L2) cache per processor. The standard ECC SDRAM memory in each node is 256 MB, expandable up to 4 GB (within two card slots). This new node is shipped with disk pairs as a standard feature to encourage the use of mirroring to significantly improve system availability. This Thin node has two internal disk bays for pairs of 4.5 GB, 9.1 GB and 18.2 GB Ultra SCSI disks. Each node has two 32-bit PCI slots and integrated 10/100 Ethernet and Ultra SCSI adapters. The POWER3 SMP Thin node can be upgraded to the POWER3 SMP Wide node.

POWER3 SMP Wide Nodes

The POWER3 SMP Wide node is a POWER3 SMP Thin node combined with additional disk bays and PCI expansion slots. This Wide node has four internal disk bays for pairs of 4.5 GB, 9.1 GB and 18.2 GB Ultra SCSI disks. Each node has 10 PCI slots (two 32-bit, eight 64-bit). Both POWER3 SMP Thin and Wide nodes are equivalent to the RS/6000 43P model 260. A diagram of the POWER3 SMP node is shown in Figure 8 on page 29. Notice that it uses docking connectors (position A) instead of flex cables as in the 332 MHz node.

Ex.1010 / Page 52 of 572

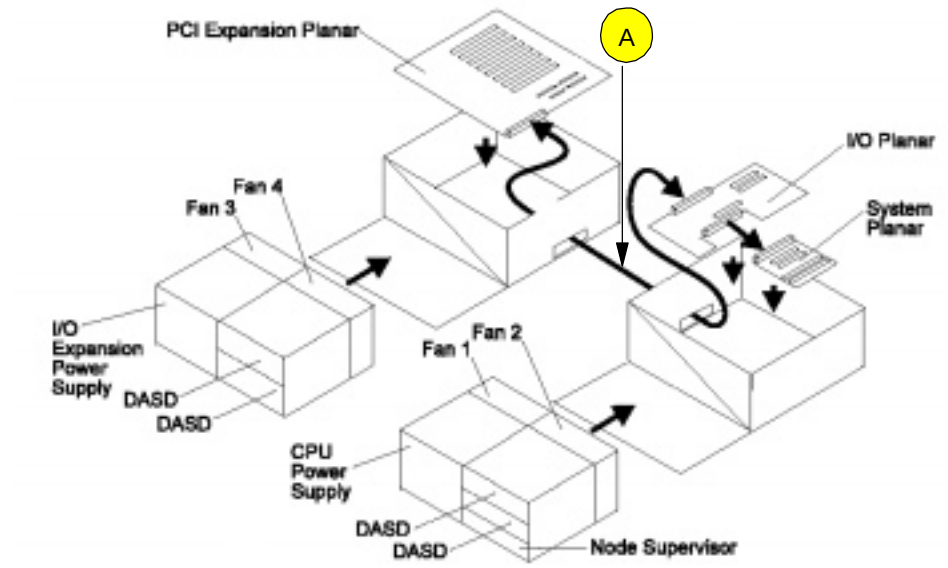


Figure 8. POWER3 SMP Node Component Diagram

The minimum software requirements for POWER3 SMP Thin and Wide nodes are AIX 4.3.2 and PSSP 3.1.

Table 2 shows a comparison of current nodes.

Table 2. Current Nodes Comparison

Node Type	**160 MHz Thin**	**332 MHz SMP Thin**	**332 MHz SMP Wide**	**POWER3 SMP Thin**	**POWER3 SMP Wide**
Processor	160 MHz P2SC	332 MHz 2- or 4-way PowerPC 604e		200 MHz 1- or 2-way POWER3	
L1 Cache (Instr./Data) per processor	32 KB/ 128 KB	32 KB/32 KB		32 KB/64 KB	
L2 Cache (per processor)	-	256 KB		4 MB	
Std. Memory	64 MB	256 MB		256 MB	
Max. Memory	1 GB	3 GB		4 GB	
Memory Slots	4	2		2	
Disk Bays	2	2	4	2	4

Node Type	**160 MHz Thin**	**332 MHz SMP Thin**	**332 MHz SMP Wide**	**POWER3 SMP Thin**	**POWER3 SMP Wide**
Min. Int. Disk	4.5 GB	None Required		None Required	
Max. Int. Disk	18.2 GB	36.4 GB or 18.2 GB (Mirror)	72.8 GB or 36.4 GB (Mirror)	36.4 GB or 18.2 GB (Mirror)	72.8 GB or 36.4 GB (Mirror)
Expansion Slots	4 MCA	2 PCI (32-bit)	10 PCI (3 64-bit, 7 32-bit)	2 PCI (32-bit)	10 PCI (8 64-bit, 2 32-bit)
Adapters	Integrated SCSI-2 F/W and Ethernet (10 Mbps)	Integrated SCSI-2 F/W and Ethernet (10 Mbps)		Integrated Ultra SCSI and Ethernet (10/100 Mbps)	

3.2.1.1 332 MHz SMP Node System Architecture

The 332 MHz SMP Thin and Wide nodes provide 2- or 4-way symmetric multiprocessing utilizing PowerPC technology, and extend the RS/6000 PCI I/O technology to the SP system. With their outstanding integer performance, these nodes are ideal for users who need mission-critical commercial computing solutions. The 332 MHz SMP node system structure is shown in Figure 9 on page 31.

Ex.1010 / Page 54 of 572

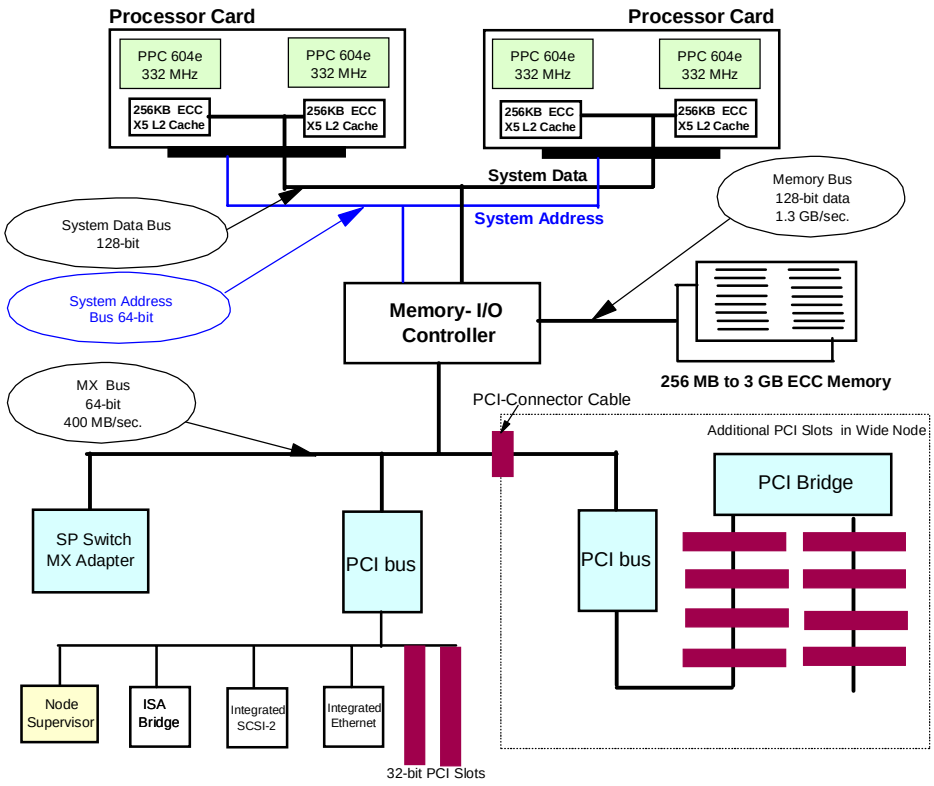


Figure 9. 332 MHz SMP Node System Architecture

Processor and Level 2 Cache Controller

The 332 MHz SMP node contains 2- or 4-way 332 MHz PowerPC 604e processors, each with its own 256 KB Level 2 cache. The X5 Level 2 cache controller incorporates several technological advancements in design providing greater performance over traditional cache designs. The cache controller implements an 8-way, dual-directory, set-associative cache using SDRAM. When instructions or data are stored in cache, they are grouped into sets of eight 64-byte lines. The X5 maintains an index to each of the eight sets. It also keeps track of the tags used internally to identify each cache line. Dual tag directories allow simultaneous processor requests and system bus snoops, thus reducing resource contention and speeding up access.

System Bus

The SMP system bus is optimized for high performance and multiprocessing applications. It has a separate 64-bit address bus and 128-bit data bus. These buses operate independently in the true split transaction mode and are aggressively pipelined. For example, new requests may be issued before previous requests are completed. There is no sequential ordering requirement. Each operation is tagged with an 8-bit tag, which allows a maximum of up to 256 transactions to be in progress in the system at any one time.

System Memory

The 332 MHz SMP node supports 256 MB to 3 GB of 10-nanosecond SDRAM. System memory is controlled by the memory-I/O chip, which is capable of providing a sustained memory bandwidth of over 1.3 GB per second. The memory controller supports up to two memory cards, with up to eight increments of SDRAM on each card.

I/O Subsystem

The memory-I/O controller implements a 64-bit, multiplexed address and data bus for attaching several PCI I/O buses and the SP Switch MX adapter. This bus runs concurrent with and independent from the system and memory buses. The peak bandwidth of this bus is 400 MB per second. Two 32-bit PCI slots are in the Thin node and three additional 64-bit PCI slots and five 32-bit PCI slots are in the Wide node.

3.2.1.2 POWER3 SMP Node System Architecture

The POWER3 SMP node has excellent performance for compute-intensive analysis applications. The heart of this node is the POWER3 microprocessor based on IBM PowerPC architecture and RS/6000 platform architecture. It provides a high bandwidth interface to a fast Level 2 (L2) cache, and a separate high bandwidth interface to memory and other system functions. The POWER3 microprocessor implements the 64-bit PowerPC architecture and is fully compatible with existing 32-bit applications.

The POWER3 SMP node system structure is shown in Figure 10 on page 33.

Ex.1010 / Page 56 of 572

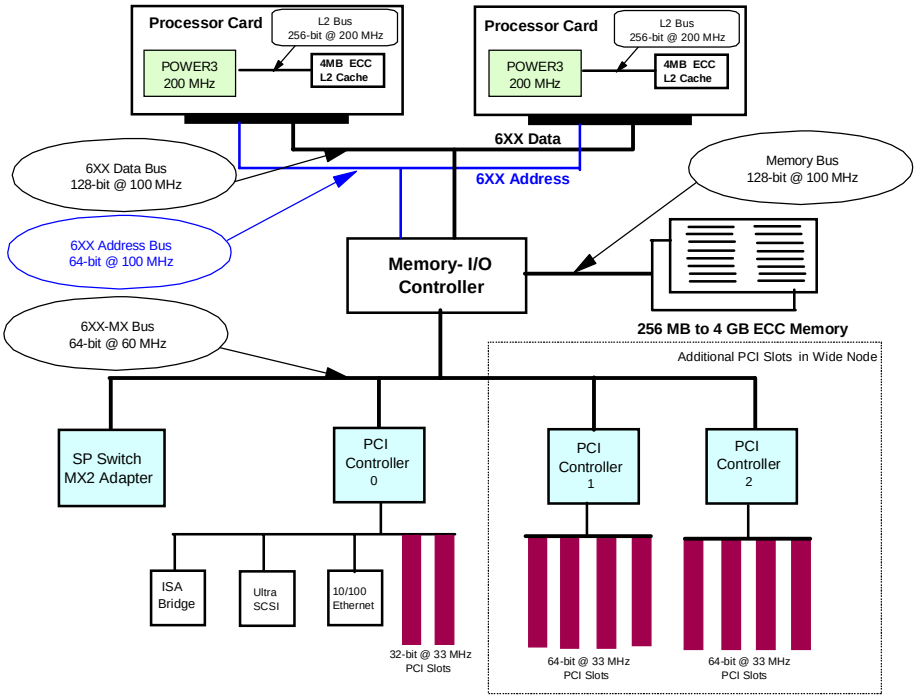


Figure 10. POWER3 SMP Node System Architecture

POWER3 Microprocessor

The POWER3 is a single chip implemented with 0.25 micron CMOS technology. It operates at 200 MHz clock cycle. The POWER3 design contains a superscalar core that is comprised of eight execution units, and allows concurrent operation of fixed-point, load/store, branch, and floating-point instructions. The processor can perform up to four floating-point operations per clock cycle. There is a 32 KB instruction and a 64 KB data Level 1 cache integrated within a single chip. Both instruction and data cache are parity protected. There is a 256-bit external interface to the 4 MB Level 2 cache, which operates at 200 MHz and is ECC protected.

System Bus

The system bus, referred to as the “6XX” bus, connects up to two POWER3 processors to the memory-I/O controller chip set. It provides 40 bits of real address and a separate 128-bit data bus. The address, data and tag buses are fully parity protected. The 6XX bus runs at a 100 MHz clock rate and peak data throughput is 1.6 GB/second.

System Memory

The POWER3 SMP node supports 256 MB to 4 GB of 10-nanosecond SDRAM. System memory is controlled by the memory-I/O chip set via the memory bus. The memory bus consists of a 128-bit data bus and operates at 100 MHz clock cycle. It is separated from the system bus (6XX bus), which allows for concurrent operations on these two buses. For example, cache-to-cache transfers can occur while a Direct Memory Access (DMA) operation is proceeding to an I/O device. There are two memory card slots, each supporting up to sixteen 128 MB memory DIMMs. Memory DIMMs must be used in pairs and at least one memory card with a minimum of 256 MB memory is required to be operational. System memory is protected with a Single Error Correction, Double Error Detection ECC code.

I/O Subsystem

The Memory-I/O controller chip set implements a 64-bit plus parity, multiplexed address and data bus (6XX-MX bus) for attaching three PCI controller chips and the SP Switch MX2 adapter. The 6XX-MX bus runs at 60 MHz clock cycle, the peak bandwidth of the 6XX-MX bus is 480 MB/second. The three PCI controllers attached to the 6XX-MX bus provide the interface for 10 PCI slots. Two 32-bit PCI slots are in the Thin node and eight additional 64-bit PCI slots are in the Wide node. One of the PCI controller chips (controller chip 0) provides support for integrated Ultra2 SCSI and 10Base2, 10/100BaseT Ethernet functions. The Ultra2 SCSI interface supports up to four internal disks. An ISA bridge chip is also attached to PCI controller chip 0 for supporting two serial ports and other internally used functions in the POWER3 SMP node.

Service Processor

The service processor function is integrated on the I/O planner board. This service processor performs system initialization, system error recovery and diagnostic functions that give the POWER3 SMP node a high level of availability. The service processor is designed to save the state of the system to 128 KB of nonvolatile memory (NVRAM) to support subsequent diagnostic and recovery actions taken by other system firmware and the AIX operating system.

3.2.2 External Nodes

A external node is a kind of processor node that cannot be housed in the frame due to its large size. The current supported external nodes are RS/6000 model S70 and RS/6000 model S70 Advanced (S7A). Both are large

Ex.1010 / Page 58 of 572

enterprise server class utilizing 64-bit symmetric multiprocessor (SMP) systems that support 32- and 64-bit applications concurrently. The bus architecture in these servers is PCI architecture. The differences between these models are the base processor (PowerPC RS64 125 MHz and PowerPC RS64 II 262 MHz respectively), standard memory and high availability I/O drawer on S70 Advanced Server. The external node is known as the SP-attached server.

These servers excel in capacity and scalability in On-line Transaction Processing (OLTP), Server Consolidation, Supply Chain Management, Enterprise Resource Planning (ERP) such as SAP, for which single large database servers are required.

3.2.2.1 SP-Attached Servers

The RS/6000 Enterprise Server Model S70 and Model S7A are packaged in two side-by-side units. The first unit is the Central Electronics Complex (CEC). The second unit is a standard 19-inch I/O rack. Up to three more I/O racks can be added to a system. Figure 11 on page 36 shows the RS/6000 model S70 and S7A scalability.

The Central Electronics Complex contains:

- Either 64-bit 125 MHz PowerPC RS64 processors (S70) or 262 MHz PowerPC RS64 II processors (S7A)
- Optional 4-way processor cards (the same processor) that scale configuration to 8-way or 12-way SMP processing
- 4 MB ECC L2 cache memory per 125 MHz processor or 8 MB per 262 MHz processor
- Standard 512 MB ECC SDRAM memory (S70) or 1 GB ECC SDRAM memory (S7A) expanded to 32 GB
- A high-speed multi-path switch, memory controller and two high-speed memory ports with a total collective memory bandwidth of up to 5.33 GB/sec (S70) and 5.6 GB/sec (S7A)

Each I/O rack accommodates up to two I/O drawers (maximum four drawers per system) with additional space for storage and communication subsystems. The base I/O drawer contains:

- A high-performance 4.5GB SCSI -2 Fast/Wide disk drive (S70) or 9.1 GB UltraSCSI disk drive (S7A)
- A 32X (Max) CD-ROM
- A 1.44 MB 3.5-inch diskette drive
- A service processor

- Fourteen PCI slots (nine 32-bit and five 64-bit) (eleven slots are available)
- Three media bays (Two available) for S70
- Two media bays (one available) for S7A
- Twelve hot-swapped disk drive bays (eleven available)

Each additional I/O drawer contains:

- Fourteen available PCI slots (nine 32-bit and five 64-bit) providing an aggregate data throughput of 500 MB per second to the I/O hub
- Three (S70) and two (S7A) available media bays
- Twelve available hot-swapped disk drive bays

When all four I/O drawers are installed, the S70 contains twelve media bays (Eight media bays for S7A), forty-eight hot-swapped disk drive bays, and fifty-six PCI slots per system.

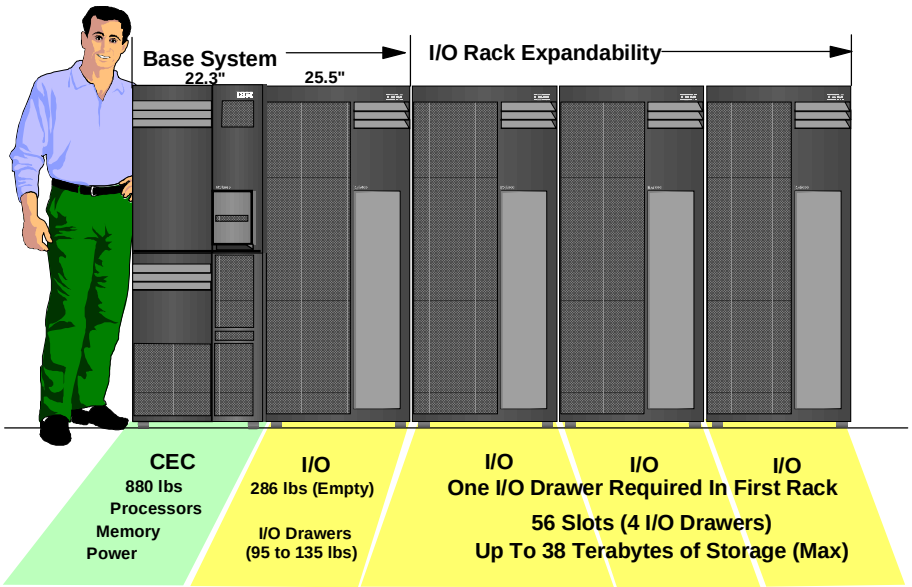


Figure 11. RS/6000 S70/S7A System Scalability

3.2.2.2 SP-Attached Server Attachment

It is important to note that the size of the S70 and S7A prohibit them from being physically mounted in the SP frame. Since the SP-attached server is mounted in its own rack, and is directly attached to the control workstation using two RS-232 cables, the SP system must view the SP-attached server

as a frame. Therefore, the SP system views the SP-attached server as an object with both frame and node characteristics.

The SP-attached server requires a minimum of four connections with the SP system in order to establish a functional and safe network. If your SP system is configured with an SP Switch, there will be five required connections as shown in Figure 12 on page 38. These connections are:

Three connections are required with the control workstation:

1. An Ethernet connection to the SP-LAN for system administration purposes
2. A RS-232 cable connecting the control workstation to the SAMI port of the SP-attached server (front panel)
3. A second RS-232 cable connecting the control workstation to the serial port of the SP-attached server (S1 port)

The fourth connection is a 10 meter frame-to-frame electrical ground cable.

The fifth connection is required if the SP system is switch configured.

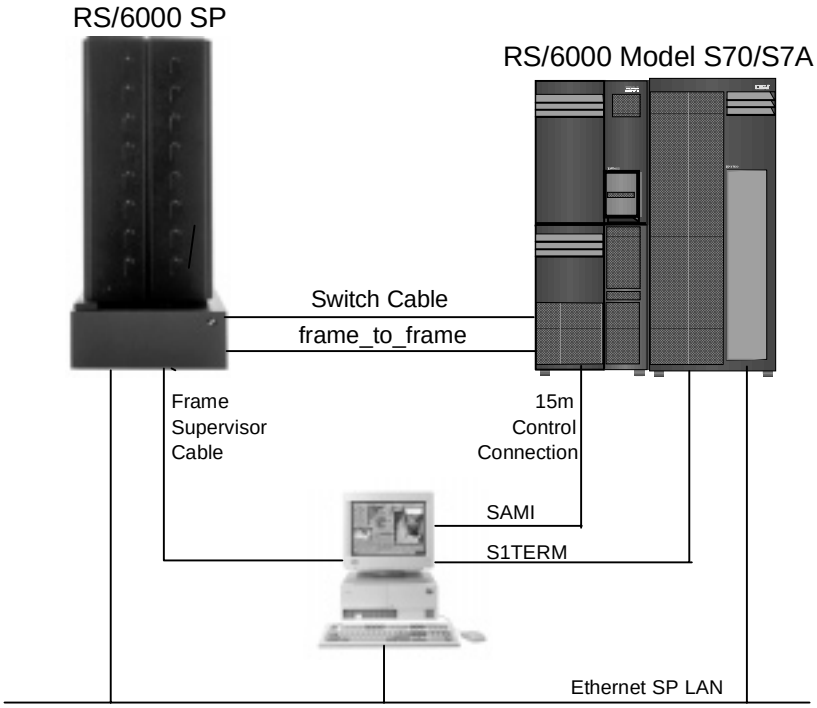


Figure 12. The SP-Attached Server Connection

3.3 Dependent Nodes

Dependent nodes are non-standard nodes that extend the SP system's capabilities, but that cannot be used in the same ways as standard SP processor nodes. 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. Typically, dependent nodes consist of four major components as follows:

1. A physical dependent node — The hardware device requiring SP processor node support.
2. A dependent node adapter — A communication card mounted in the physical dependent node. This card provides a mechanical interface for the cable connecting the physical dependent node to the SP system.
3. A logical dependent node — Made up of a valid, unused node slot and the corresponding unused SP switch port. The physical dependent node logically occupies the empty node slot by using the corresponding

SP switch port. The switch port provides a mechanical interface for the cable connecting the SP system to the physical dependent node.

4. A cable - To connect the dependent node adapter with the logical dependent node. It connects the extension node to the SP system.

3.3.1 SP Switch Router

The IBM 9077 SP Switch Router is a specific type of dependent node. The 9077 is a licensed version of the Ascend GRF (Goes Real Fast) switched IP router that has been enhanced for direct connection to the SP Switch. The SP Switch Router was known as the High Performance Gateway Node (HPGN) during the development of the adapter. These optional external devices can be used for high-speed network connections or system scaling using HIPPI backbones or other communications subsystems such as ATM or 10/100 Ethernet (see Figure 13 on page 40).

An SP Switch Router may have multiple logical dependent nodes, one for each dependent node adapter it contains. If an SP Switch Router contains more than one dependent node adapter, it can route data between SP systems or system partitions. For an SP Switch Router, this card is called a Switch Router Adapter (feature code #4021). Data transmission is accomplished by linking the dependent node adapters in the switch router with the logical dependent nodes located in different SP systems or system partitions.

In addition to the four major dependent node components, the SP Switch Router has a fifth optional category of components. These components are networking cards that fit into slots in the SP Switch Router. In the same way that the SP Switch Router Adapter connects the SP Switch Router directly to the SP Switch, these networking cards enable the SP Switch Router to be directly connected to an external network. The following networks can be connected to the RS/6000 SP Switch Router using available media cards:

- Ethernet 10/100 Base-T
- FDDI
- ATM OC-3c (single or multimode fiber)
- SONET OC-3c (single or multimode fiber)
- ATM OC-12c (single or multimode fiber)
- HiPPI
- HSSI

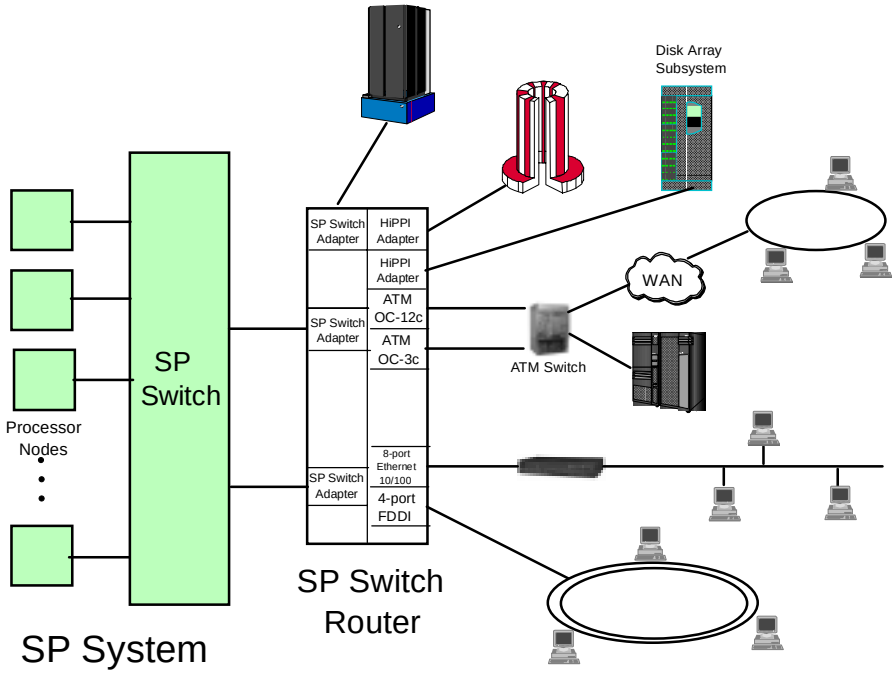


Figure 13. SP Switch Router

Although you can equip an SP node with a variety of network adapters and use the node to make your network connections, the SP Switch Router with the Switch Router Adapter and optional network media cards offers many advantages when connecting the SP to external networks, as follows:

- Each media card contains its own IP routing engine with separate memory containing a full route table of up to 150,000 routes. Direct access provides much faster lookup times compared to software-driven lookups.
- Media cards route IP packets independently at rates of 60,000 to 130,000 IP packets per second. With independent routing available from each media card, the SP Switch Router gives your SP system excellent scalability characteristics.
- The SP Switch Router has a dynamic network configuration to bypass failed network paths using standard IP protocols.
- Using multiple Switch Router Adapters in the same SP Switch Router, you can provide high performance connections between system partitions in a single SP system or between multiple SP systems.

Ex.1010 / Page 64 of 572

- A single SP system can also have more than one SP Switch Router attached to it, further insuring network availability.
- Media cards are hot swappable for uninterrupted SP Switch Router operations.
- Each SP Switch Router has redundant (N+1) hot swappable power supplies.

Two versions of the RS/6000 SP Switch Router can be used with the SP Switch. The Model 04S (GRF 400) offers four media card slots and the Model 16S (GRF 1600) offers 16 media card slots. Except for the additional traffic capacity of the Model 16S, both units offer similar performance and network availability, as shown in Figure 14.

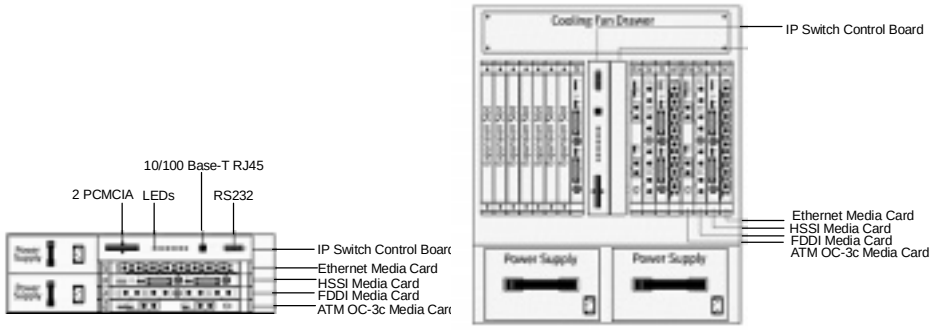


Figure 14. GRF Model 400 and Model 1600

3.3.2 SP Switch Router Attachment

The SP Switch Router requires a minimum of three connections with your SP system in order to establish a functional and safe network. These connections are:

1. A network connection with the control workstation — The SP Switch Router must be connected to the control workstation for system administration purposes. This connection may be either:
   - A direct Ethernet connection between the SP Switch Router and the control workstation
   - An Ethernet connection from the SP Switch Router to an external network which then connects to the control workstation

2. A connection between an SP Switch Router Adapter and the SP Switch — The SP Switch Router transfers information into and out of the processor nodes of your SP system. The link between the SP Switch Router and the SP processor nodes is implemented by:
   - An SP Switch Router adapter
   - A switch cable connecting the SP Switch Router adapter to a valid switch port on the SP Switch
3. A frame-to-frame electrical ground — The SP Switch Router frame must be connected to the SP frame with a grounding cable. This frame-to-frame ground is required in addition to the SP Switch Router electrical ground. The purpose of the frame-to-frame ground is to maintain the SP and SP Switch Router systems at the same electrical potential.

For more information refer to *IBM 9077 SP Switch Router: Get Connected to the SP Switch, SG24-5157*.

3.4 Control Workstation

The RS/6000 SP system requires an RS/6000 workstation. The control workstation serves as a point of control for managing, monitoring, and maintaining the RS/6000 SP frames and individual processor nodes. It connects to each frame via an RS232 line to provide hardware control functions. It also connects to each external node or SP-attached server with two RS232 cables, but hardware control is minimal because SP-attached 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 in the RS/6000 SP system. In addition, the control workstation can be set up as an authentication server using Kerberos. It can be the Kerberos primary server, with the master database and administration service, as well as the ticket-granting service. As an alternative, the control workstation can be set up as a Kerberos secondary server, with a backup database, to perform ticket-granting services.

An optional High Availability Control Workstation (HACWS) option allows a backup control workstation to be connected to an SP system. The second control workstation provides backup when the primary workstation requires update services or fails.

Ex.1010 / Page 66 of 572

3.4.1 Supported Control Workstations

There are two basic types of control workstations:

- MCA-based
- PCI-based

Both types of control workstations must be connected to each frame through an RS-232 cable and the SP Ethernet.

MCA-based Control Workstations:

- RS/6000 7012 Models 37T, 370, 375, 380, 39H, 390, 397, G30, and G40
- RS/6000 7013 Models 570, 58H, 580, 59H, 590, 591, 595, J30, J40, and J50 (See note 1.)
- RS/6000 7015 Models 97B, 970, 98B, 980, 990, R30, R40, and R50 (See note 1.)
- RS/6000 7030 Models 3AT, 3BT and 3CT

Note:

1. Requires a 7010 Model 150 X-Station and display. Other models and manufacturers that meet or exceed this model can be used. An ASCII terminal is required as the console.

PCI-based Control Workstations:

- RS/6000 7024 Models E20 and E30 (See following note 1.)
- RS/6000 7025 Model F30 (See following notes 1 and 2.)
- RS/6000 7025 Models F40 and F50 (See note 3.)
- RS/6000 7026 Models H10 and H50 (See note 3.)
- RS/6000 7043 43P Models 140 and 240 (See notes 3, 4 and 5.)

Notes:

1. Supported by PSSP 2.2 and later.
2. On systems introduced since PSSP 2.4, either the 8-port (feature code #2493) or 128-port (feature code #2944) PCI bus asynchronous adapter should be used for frame controller connections. IBM strongly suggests you use the support processor option (feature code #1001). If you use this option, the frames must be connected to a serial port on an asynchronous adapter and not to the serial port on the control workstation planar board.

3. The native RS232 ports on the system planar can not be used as tty ports for the hardware controller interface. The 8-port asynchronous adapter EIA-232/ RS-422, PCI bus (feature code #2943) or the 128-port Asynchronous Controller (feature code #2944) are the only RS232 adapters that are supported. These adapters require AIX 4.2.1 or AIX 4.3 on the control workstation.
4. The 7043 can only be used on SP systems with up to four frames. This limitation applies to the number of frames and not the number of nodes. This number includes expansion frames.
5. The 7043-43P is *not* supported as a control workstation whenever an S70/S7A is attached to the SP. The limitation is due to the load that the extra daemons place on the control workstation.

3.4.2 Control Workstation Minimum Hardware Requirements

The minimum hardware requirements for the control workstation are:

- At least 128 MB of main memory. For SP systems with more than 80 nodes, 256 MB is required; 512 MB of memory is recommended.
- 4 GB of disk storage plus 1 GB for each AIX release and modification level in your SP system. Double the number of physical disks you plan on using rootvg mirroring.
- Physically installed with the RS232 cable to each SP frame.
- Physically installed with two RS232 cables to each external node SP-attached server, such as an RS/6000 Enterprise Server Model S70 or S70 Advanced.
- Equipped with the following I/O devices and adapters:
  - A 3.5 inch diskette drive.
  - Four- or 8-millimeter (or equivalent) tape drive.
  - SCSI CD-ROM drive.
  - One RS232 port for each SP frame.
  - Keyboard and mouse.
  - Color graphics adapter and color monitor. (An X-station model 150 and display are required if an RS/6000 that does not support a color graphics adapter is used.)
  - SP Ethernet adapters for connection to the SP Ethernet (see 3.5, “Administrative Ethernet” on page 47 for details).

Ex.1010 / Page 68 of 572

3.4.3 High Availability Control Workstation

The design of the SP High Availability Control Workstation (HACWS) is modeled on the High Availability Cluster Multi-Processing for RS/6000 (HACMP) licensed program product. HACWS utilizes HACMP running on two RS/6000 control workstations in a two-node rotating configuration. HACWS utilizes an external disk that is accessed nonconcurrently between the two control workstations for storage of SP related data. There is also a Y-cable connected from the SP frame supervisor card to each control workstation. This HACWS configuration provides automated detection, notification, and recovery of control workstation failures. Figure 15 on page 46 shows the logical view of HACWS attachment.

Ex.1010 / Page 69 of 572

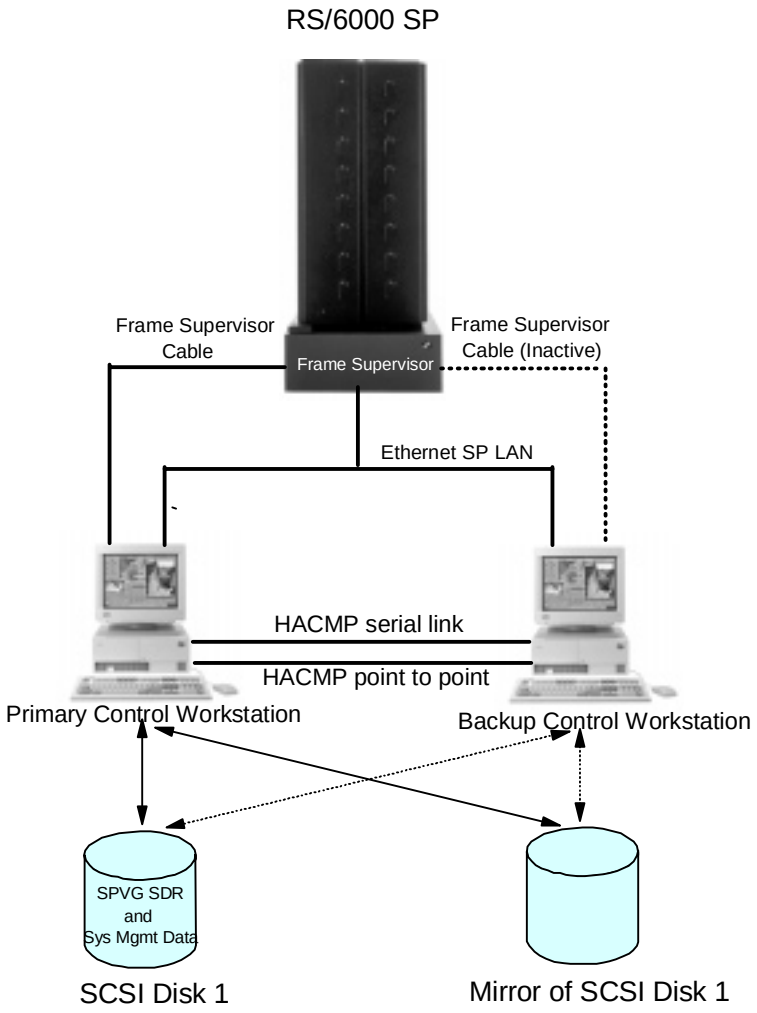


Figure 15. High Availability Control Workstation (HACWS) Attachment

The primary and backup control workstations are also connected on a private point-to-point network and a serial TTY link or target mode SCSI. The backup control workstation assumes the IP address, IP aliases, and hardware address of the primary control workstation. This lets client applications run without changes. The client application, however, must initiate reconnects when a network connection fails.

The HACWS has the following limitations and restrictions:

- You cannot split the load across a primary and backup control workstation. Either the primary or the backup provides all the functions at one time.
- The primary and backup control workstations must each be an RS/6000. You cannot use a node at your SP as a backup control workstation.
- The backup control workstation cannot be used as the control workstation for another SP system.
- The backup control workstation cannot be a shared backup of two primary control workstations.
- There is a one-to-one relationship of primary to backup control workstations; a single primary and backup control workstation combination can be used to control only one SP system.
- If a primary control workstation is an SP authentication server, the backup control workstation must be a secondary authentication server.
- The S70 and S70 Advanced SP-attached servers are directly attached to the control workstation through two RS232 serial connections. There is no dual RS232 hardware support for these connections like there is for SP frames. These servers can only be attached to one control workstation at a time. Therefore, when a control workstation fails or scheduled downtime occurs, and the backup control workstation becomes active, you will lose hardware monitoring and control and serial terminal support for your SP-attached servers. The SP-attached servers will have the SP Ethernet connection from the backup control workstation, so PSSP components requiring this connection will still work correctly. This includes components such as the availability subsystems, user management, logging, authentication, the SDR, file collections, accounting and others.

3.5 Administrative Ethernet

The SP requires an Ethernet connection between the control workstation and all nodes, which is used for network installation of the nodes and for system management. This section describes the setup of that administrative Ethernet, which is often called the “SP LAN.”

3.5.1 Frame and Node Cabling

SP frames include coaxial Ethernet cabling for the SP LAN, also known as "thin-wire" Ethernet or 10BASE-2. All nodes in a frame can be connected to that medium through the BNC connector of either their integrated 10 Mbps Ethernet or a suitable 10 Mbps Ethernet adapter, using T-connectors. Access to the medium is shared among all connected stations, and controlled by Carrier Sense, Multiple Access/Collision Detect (CSMA/CD). 10BASE-2 only

supports half duplex (HDX). There is a hard limit of 30 stations on a single 10BASE-2 segment, and the total cable length must not exceed 185 meters. However, it is not advisable to connect more than 16 to 24 nodes to a single segment. Normally, there is one segment per frame and one end of the coaxial cable is terminated in the frame. Depending on the network topology, the other end connects the frame to either the control workstation or to a boot/install server in that segment, and is terminated there. In the latter case, the boot/install server and control workstation are connected through an additional Ethernet segment, so the boot/install server needs two Ethernet adapters.

It is also possible to use customer-provided Unshielded Twisted Pair (UTP) cabling of category 3, 4 or 5. A UTP cable can be directly connected to the RJ-45 Twisted Pair (TP) connector of the Ethernet adapter if one is available, or through a transceiver/media converter to either the AUI or BNC connector. Twisted Pair connections are always point-to-point connections. So all nodes have to be connected to a customer-provided repeater or Ethernet switch, which is normally located outside the SP frame and is also connected to the control workstation. Consequently, using UTP involves much more cabling. On the other hand, fault isolation will be much easier with UTP than with thin-wire Ethernet, and there are more opportunities for performance improvements. Twisted Pair connections at 10 Mbps are called 10BASE-T, those operating at 100 Mbps are called 100BASE-TX.

In order to use Twisted Pair in full duplex mode, there must be a native RJ-45 TP connector at the node (no transceiver), and an Ethernet switch like the IBM 8274 must be used. A repeater always works in half duplex mode, and will send all IP packets to all ports (like in the 10BASE-2 LAN environment). We therefore recommend that you always use an Ethernet switch with native UTP connections.

The POWER3 SMP nodes (made available in 1999) have an integrated 10/100 Mbps Ethernet adapter. They may still be connected and installed at 10 Mbps, using 10BASE-T or 10BASE-2 and a transceiver. However, to fully utilize the adapter at 100 Mbps, category 5 UTP wiring to a 100 Mbps repeater or Ethernet switch is required (100BASE-TX). As previously mentioned, we recommend the use of an Ethernet switch since this allows you to utilize the full duplex mode and avoids collisions. The control workstation also needs a fast connection to this Ethernet switch.

A list of the Ethernet adapters supported by PSSP 3.1 can be found in Appendix A., “Currently Supported Adapters” on page 517.

Ex.1010 / Page 72 of 572

3.5.2 SP LAN Topologies

The network topology for the SP LAN mainly depends on the size of the system, and should be planned on an individual basis. We strongly recommend that additional network connectivity be provided (through the SP Switch or additional Ethernet, Token Ring, FDDI or ATM networks) if the applications on the SP perform significant communication among nodes. To avoid overloading the SP LAN by application traffic, it should be used only for SP node installations and system management, and applications should use these additional networks.

In the following, only the SP LAN is considered. We show some typical network topologies, their advantages, and limitations.

3.5.2.1 Shared 10BASE-2 Network

In relatively small SP configurations like single frame systems, the control workstation and nodes typically share a single thin-wire Ethernet. Figure 16 shows this setup.

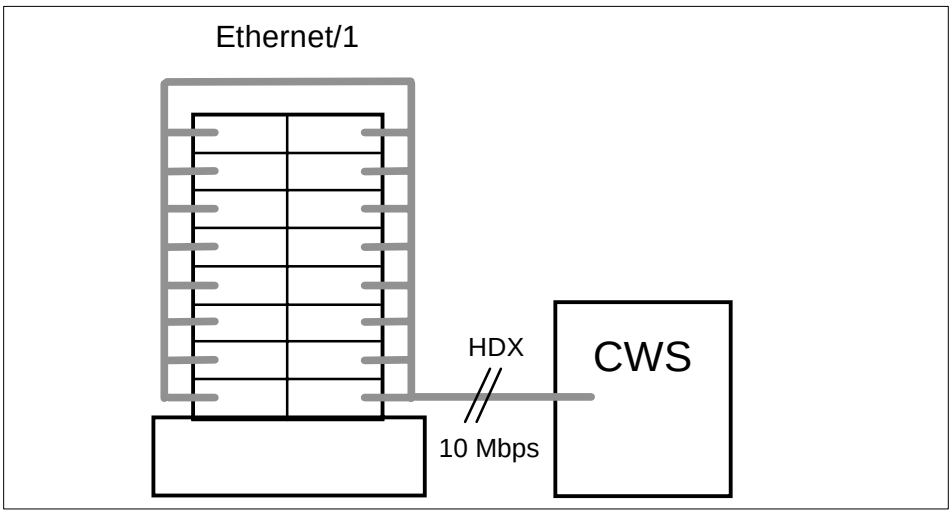


Figure 16. Shared 10BASE-2 SP Network

This configuration is characterized by the following properties:

- No routing is required, since the control workstation and all nodes share one subnet.
- Consequently, the whole SP LAN is a single *broadcast domain* as well as a single *collision domain*.

- The control workstation acts as boot/install server for all nodes.
- Performance is limited to one 10-Mbps HDX connection at a time.
- Only six to eight network installs of SP nodes from the control workstation NIM server could be performed simultaneously.

Even if this performance limitation is accepted, this setup is limited by the maximum number of 30 stations on a 10BASE-2 segment. In practice, not more than 16 to 24 stations should be connected to a single 10BASE-2 Ethernet segment.

3.5.2.2 Segmented 10BASE-2 Network

A widely used approach to overcome the limitations of a single shared Ethernet is segmentation. The control workstation is equipped with additional Ethernet adapters, and each one is connected to a different shared 10BASE-2 Ethernet subnet. This is shown in Figure 17.

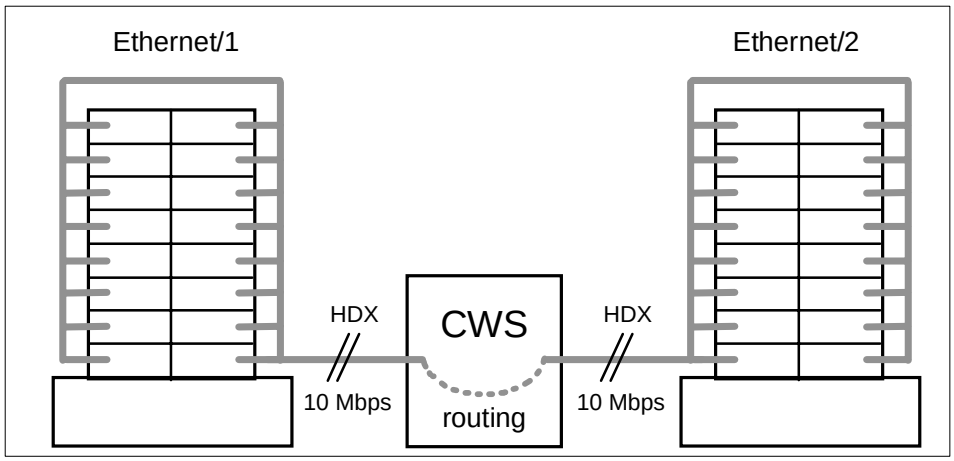


Figure 17. Segmented 10BASE-2 SP Network with Two Subnets

For a configuration with N separate subnets (and consequently N Ethernet cards in the control workstation), the following holds:

- Nodes in one subnet need static routes to the (N-1) other subnets through the control workstation, and routing (or IP forwarding) must be enabled on the control workstation.
- The SP LAN is split into N broadcast domains.
- The control workstation acts as boot/install server for all nodes, since it is a member of all N subnets.

Ex.1010 / Page 74 of 572

- Aggregate performance is limited to a maximum of N times 10 Mbps HDX. However, this is only achievable if the control workstation communicates with one node in each of the subnets simultaneously.
- Only six to eight network installs per subnet should be performed simultaneously, increasing the maximum to 6N to 8N simultaneous installs.

Attention

This approach is limited primarily by the number of available adapter slots in the control workstation, but also by the ability of the control workstation to simultaneously handle the traffic among these subnets or to serve 6N to 8N simultaneous network installations. In practice, more than four subnets should not be used.

3.5.2.3 Segmented 10BASE-2 Networks with Boot/Install Servers

For very large systems where this model of segmentation would require more 10-Mbps Ethernet adapters in the control workstation than possible, a more complex network setup can be deployed which uses additional boot/install servers. This is shown in Figure 18 on page 52.

The control workstation is directly connected to only one Ethernet segment, which is attached to the 10-Mbps Ethernet adapter en0 of a set of N boot/install server (BIS) nodes, typically the first node in each frame. We call this Ethernet subnet the Install Ethernet, since it is the network through which the control workstation installs the boot/install server nodes.

The remaining nodes are grouped into N additional Ethernet segments (typically one per frame), which are not directly connected to the control workstation. Instead, each of these subnets is connected to one of the boot/install servers through a second 10-Mbps Ethernet adapter in the boot/install servers.

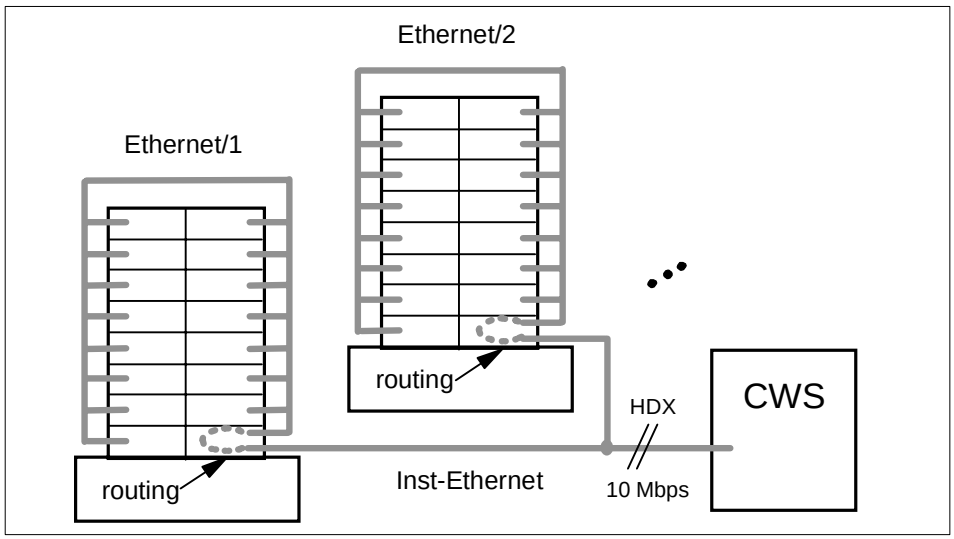


Figure 18. Segmented SP Network with Boot/Install Server Hierarchy

Note the following with such a network configuration.

- Routing is complicated:
  - Non-BIS nodes in a segment have routes to all other segments through their BIS node.
  - BIS nodes have routes to the (N-1) other nodes' segments through the BIS nodes attached to these segments.
  - The control workstation has routes to the N nodes' segments through the BIS nodes in these segments.
- The SP LAN is split into (N+1) broadcast domains.
- The boot/install servers are installed from the NIM server on the control workstation. After this, all non-BIS nodes are installed by the boot/install servers. Note that some NIM resources, such as the LPPSOURCE, are only served by the control workstation.
- The maximum bandwidth in the Install Ethernet segment (including the control workstation) is 10 Mbps HDX.
- Only six to eight BIS nodes can be installed simultaneously from the control workstation in a first installation phase. In a second phase, each BIS node can install six to eight nodes in its segment simultaneously.

Apart from the complex setup, this configuration suffers from several problems. Communication between regular nodes in different subnets requires routing through two boot/install server nodes. All this traffic, and all communications with the control workstation (routed through one BIS node), have to compete for bandwidth on the single shared 10-Mbps half duplex Install Ethernet.

The situation can be improved by adding a dedicated router. Connecting all the nodes' segments to this router removes the routing traffic from the BIS nodes, and using a fast "uplink" connection to the control workstation provides an alternative, high bandwidth path to the control workstation. The BIS nodes in each segment are still required, because the network installation process requires that the NIM server and the client are in the same broadcast domain. Figure 19 shows such a configuration. Nodes in the frames now have a route to the control workstation and the other frames' networks through the router, which offloads network traffic from the BIS nodes.

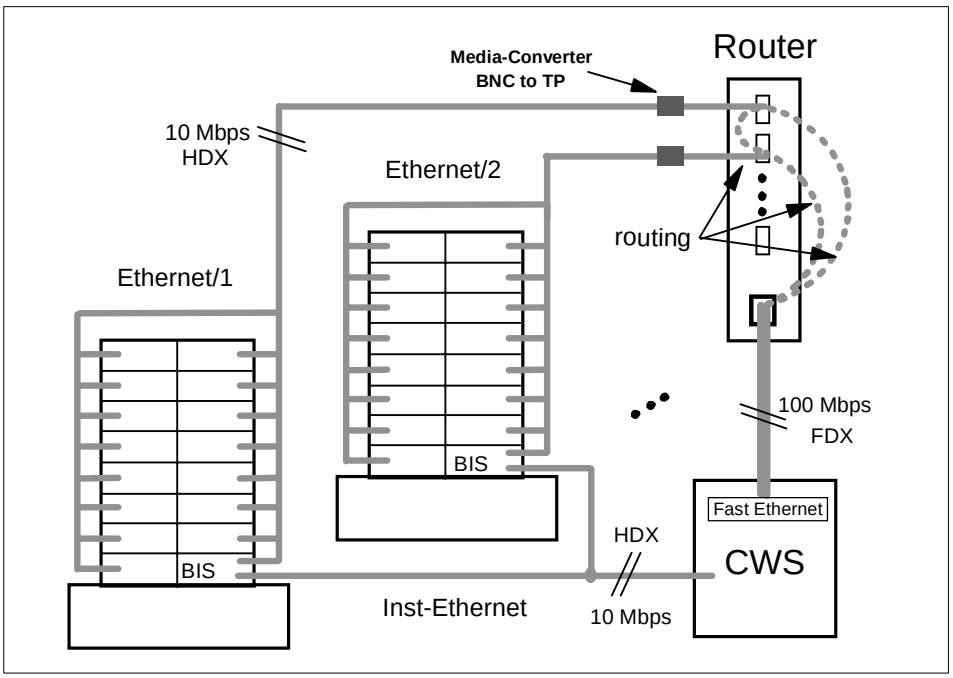


Figure 19. Boot/Install Server Hierarchy with Additional Router

Even when a router is added, the solution presented in the following section is normally preferable to a segmented network with boot/install servers, both

from a performance viewpoint and from a management and complexity viewpoint.

3.5.2.4 Switched 10BASE-2 Network

An emerging technology to overcome performance limitations in shared or segmented Ethernet networks is Ethernet Switching, sometimes called micro-segmentation. An SP example is shown in Figure 20.

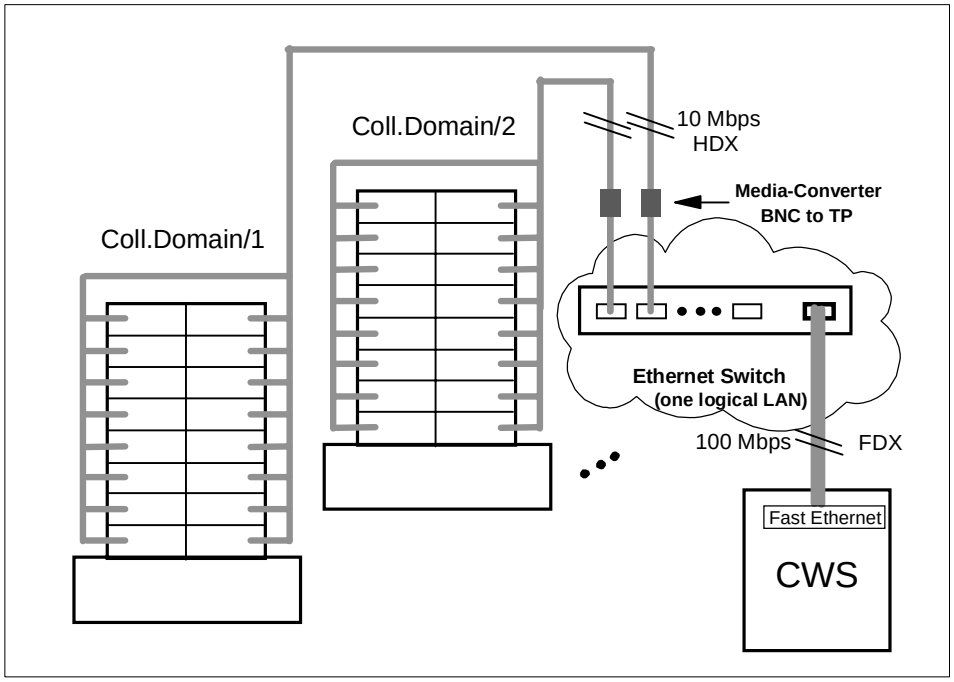


Figure 20. Switched 10BASE-2 SP Network with Fast Uplink

This configuration has the following properties:

- No routing is required. All Ethernet segments are transparently combined to one big LAN by the Ethernet switch.
- Of course, node-to-node connections within a single Ethernet segment still have to share that 10BASE-2 medium in half duplex mode. But many communications between different ports can be switched simultaneously by the Ethernet switch. The uplink to the control workstation can be operated in 100 Mbps full duplex mode.
- The control workstation can act as the boot/install server for all nodes, since the Ethernet switch combines the control workstation and nodes into one network (or broadcast domain).

Ex.1010 / Page 78 of 572

This setup eliminates the routing overhead for communications between nodes or a node and the control workstation. With a 100-Mbps, full duplex Ethernet uplink to the control workstation, there should also be no bottleneck in the connection to the control workstation, at least if the number of 10BASE-2 segments is not much larger than 10.

Considering only the network topology, the control workstation should be able to install 6 to 8 nodes in each Ethernet segment (port on the Ethernet switch) simultaneously, since each Ethernet segment is a separate *collision domain*. Rather than the network bandwidth, the limiting factor most likely is the ability of the control workstation itself to serve a very large number of NIM clients simultaneously, for example answering UPD bootp requests or acting as the NFS server for the mksysb images. To quickly install a large SP system, it may therefore still be useful to set up boot/install server nodes, but the network topology itself does not require boot/install servers. For an installation of all nodes of a large SP system, we advocate the following:

1. Using the `spbootins` command, set up approximately as many boot/install server nodes as can be simultaneously installed from the control workstation.
2. Install the BIS nodes from the control workstation.
3. Install the non-BIS nodes from their respective BIS nodes. This provides the desired scalability for the installation of a whole, large SP system.
4. Using the `spbootins` command, change the non-BIS nodes' configuration so that the control workstation becomes their boot/install server. Do not forget to run `setup_server` to make these changes effective.
5. Reinstall the original BIS nodes. This removes all previous NIM data from them, since no other node is configured to use them as boot/install server.

Using this scheme, the advantages of both a hierarchy of boot/install servers (scalable, fast installation of the whole SP system) and a flat network with only the control workstation acting as a NIM server (less complexity, less disk space for BIS nodes) are combined. Future reinstallations of individual nodes (for example, after a disk crash in the root volume group) can be served from the control workstation. Note that the control workstation will be the only file collection server if the BIS nodes are removed, but this should not cause performance problems.

The configuration in Figure 20 scales well to about 128 nodes. For larger systems, the fact that all the switched Ethernet segments form a single broadcast domain can cause network problems if operating system services or applications frequently issue broadcast messages. Such events may cause "broadcast storms", which can overload the network. For example, Topology

Services from the RS/6000 Cluster Technology use broadcast messages when the group leader sends PROCLAIM messages to attract new members.

Attention

ARP cache tuning: Be aware that for SP systems with very large networks (and/or routes to many external networks), the default AIX settings for the ARP cache size might not be adequate. The Address Resolution Protocol (ARP) is used to translate IP addresses to Media Access Control (MAC) addresses, and vice versa. Insufficient APR cache settings can severely degrade your network's performance, in particular when many broadcast messages are sent. Refer to /usr/lpp/ssp/README/ssp.css.README and the redbook *RS/6000 SP Performance Tuning*, SG24-5340, for more information about ARP cache tuning.

In order to avoid problems with broadcast traffic, no more than 128 nodes should be connected to a single switched Ethernet subnet. Larger systems should be set up with a suitable number of switched subnets. To be able to network boot and install from the control workstation, each of these switched LANs must have a dedicated connection to the control workstation. This can be accomplished either through multiple uplinks between one Ethernet switch and the control workstation, or through multiple switches which each have a single uplink to the control workstation.

3.5.2.5 Shared or Switched 100BASE-TX Network

With the introduction of the POWER3 SMP nodes in 1999, it has become possible to operate nodes on the SP LAN at 100 Mbps, including network installation. This requires UTP cabling as outlined in 3.5.1, "Frame and Node Cabling" on page 47.

One possible configuration would be to use a repeater capable of sustaining 100 Mbps, and a fast Ethernet adapter in the control workstation. This would boost the available bandwidth up to 100 Mbps, but it would be shared among all stations, and connections are only half duplex. Although the bandwidth would be higher by a factor of 10 compared to a 10BASE-2 SP Ethernet, we recommend to use an Ethernet switch, which supports full duplex connections at 100 Mbps, instead of a repeater. Many node-to-node and node-to-control workstation connections can be processed by the Ethernet switch simultaneously, rather than the shared access through a repeater. This configuration is shown in Figure 21 on page 57. As discussed in the previous section, the limiting factor for the number of simultaneous network installations of nodes will probably be the processing power of the control workstation, not the network bandwidth.

Ex.1010 / Page 80 of 572

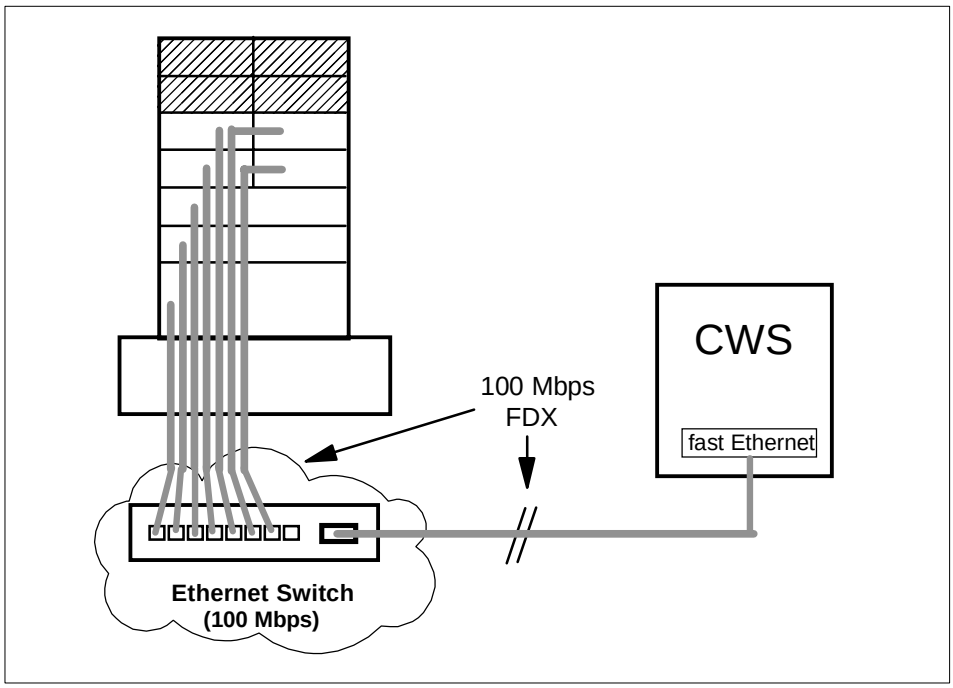


Figure 21. Simple 100BASE-TX SP Network

For larger SP configurations, the cabling required to establish point-to-point connections from all nodes to the Ethernet switch can be impressive. An IBM 8274 Nways LAN RouteSwitch could be used to provide the required switching capacities; models with 3, 5 or 9 switching modules are available.

3.5.2.6 Heterogeneous 10/100 Mbps Network

In many cases, an existing SP system will be upgraded by new nodes which have fast Ethernet connections, but older or less lightly loaded nodes should continue to run with 10 Mbps SP LAN connections. A typical scenario with connections at both 10 Mbps and 100 Mbps is shown in Figure 22 on page 58.

Ex.1010 / Page 81 of 572

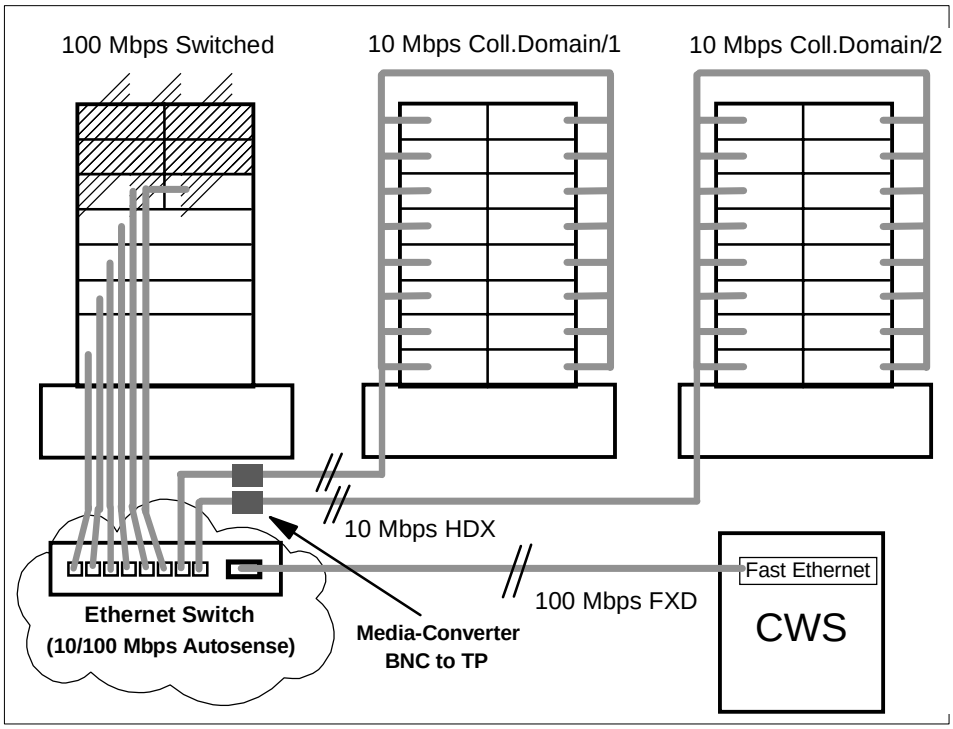


Figure 22. Heterogeneous 10/100 Mbps SP Network

In this configuration, an Ethernet switch like the IBM 8274 is again used to provide a single LAN, and connects to the control workstation at 100 Mbps FDX. One frame has new nodes with a 100-Mbps Ethernet, which are individually cabled by 100BASE-TX Twisted Pair to ports of the Ethernet Switch, and operate in full duplex mode as in the previous example. Two frames with older nodes and 10BASE-2 cabling are connected to ports of the same Ethernet switch, using media converters as in the configuration shown in Figure 20 on page 54. Ideally, a switching module with autosensing ports is used, which automatically detects the communication speed.

3.6 SP Switch Communication Network

During the initial development of the SP system, a high-speed interconnection network was required to enable communication between the nodes that make up the SP complex. The initial requirement was to support the demands of parallel applications that utilize the distributed memory MIMD programming model. More recently, the SP Switch network has been extended to a variety of purposes:

Ex.1010 / Page 82 of 572

- Primary network access for users external to the SP complex (when used with SP Switch Router)
- Used by ADSM for node backup and recovery
- Used for high-speed internal communications between various components of third-party application software (for example, SAP's R/3 suite of applications)

All of these applications are able to take advantage of the sustained and scalable performance provided by the SP Switch. The SP Switch provides the message passing network that connects all of the processors together in a way that allows them to send and receive messages simultaneously.

Two networking topologies can be used to connect parallel machines: direct and indirect.

In direct networks, each switching element connects directly to a processor node. Each communication hop carries information from the switch of one processor node to another.

Indirect networks, on the other hand, are constructed such that some intermediate switch elements connect only to other switch elements. Messages sent between processor nodes traverse one or more of these intermediate switch elements to reach their destination. The advantages of the SP Switch network are:

- Bisectional bandwidth scales linearly with the number of processor nodes in the system.

  Bisectional bandwidth is the most common measure of total bandwidth for parallel machines. Consider all possible planes that divide a network into two sets with an equal number of nodes in each. Consider the peak bandwidth available for message traffic across each of these planes. The bisectional bandwidth of the network is defined as the minimum of these bandwidths.
- The network can support an arbitrarily large interconnection network while maintaining a fixed number of ports per switch.
- There are typically at least four shortest-path routes between any two processor nodes. Therefore, deadlock will not occur as long as the packet travels along any shortest-path route.
- The network allows packets that are associated with different messages to be spread across multiple paths, thus reducing the occurrence of hot spots.

The hardware component that supports this communication network consists of two basic components: the SP Switch adapter and the SP Switch board. There is one SP Switch adapter per processor node and generally one SP Switch board per frame. This setup provides connections to other processor nodes. Also, the SP system allows switch boards-only frames that provide switch-to-switch connections and greatly increase scalability.

3.6.1 SP Switch Hardware Components

This section discusses the hardware components that make up the SP Switch network: the Switch Link, the Switch Port, the Switch Chip, the Switch Adapter and the Switch Board. The Switch Link itself is the physical cable connecting two Switch Ports. The Switch Ports are hardware subcomponents that can reside on a Switch Adapter that is installed in a node or on a Switch Chip that is part of a Switch Board.

3.6.1.1 SP Switch Board

An SP Switch board contains eight SP Switch chips that provide connection points for each of the nodes to the SP Switch network, as well as for each of the SP Switch Boards to the other SP Switch Boards. The SP Switch chips each have a total of 8 Switch Ports, which are used for data transmission. The Switch Ports are connected to other Switch ports via a physical Switch Link.

In summary, there are 32 external SP Switch Ports in total. Of these, 16 are available for connection to nodes, and the other 16 to other SP Switch Boards. The SP Switch board is mounted in the base of the SP Frame, above the power supplies.

A schematic diagram of the SP Switch board is shown on Figure 23 on page 61.

Ex.1010 / Page 84 of 572

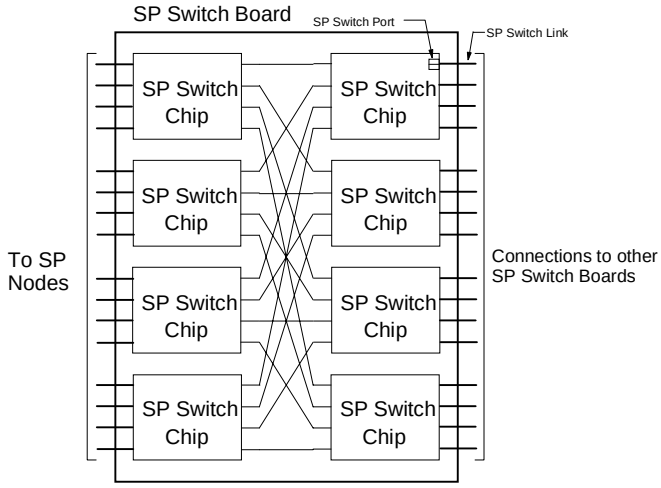


Figure 23. SP Switch Board

3.6.1.2 SP Switch Link

An SP Switch Link connects two switch network devices. It contains two channels carrying packets in opposite directions. Each channel includes:

- Data (8 bits)
- Data validation (1 bit)
- Token signal (1 bit)

The first two elements here are driven by the transmitting element of the link, while the last element is driven by the receiving element of the link.

3.6.1.3 SP Switch Port

An SP Switch Port is part of a network device (either the SP Switch adapter or SP Switch chip) and is connected to other SP Switch Ports through the SP Switch Link. The SP Switch Port includes two ports (input and output) for full duplex communication.

The relationship between the SP Switch chip link and the SP Switch chip port is shown in Figure 24 on page 62.

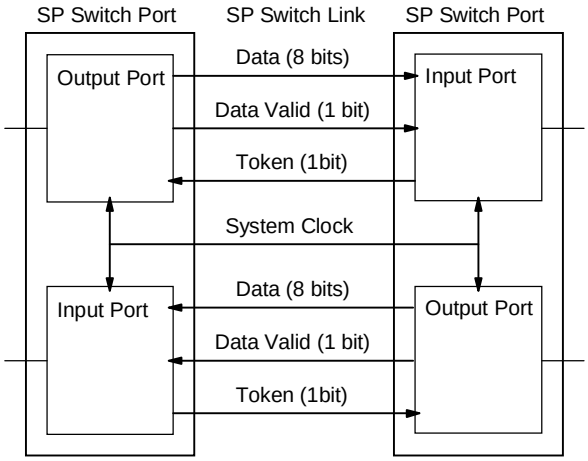


Figure 24. Relationship Between Switch Chip Link and Switch Chip Port

3.6.1.4 SP Switch Chip

An SP Switch chip contains eight SP Switch Ports, a central queue and an unbuffered crossbar, which allows packets to pass directly from receiving ports to transmitting ports. These crossbar paths allow packets to pass through the SP Switch (directly from the receivers to the transmitters) with low latency, whenever there is no contention for the output port. As soon as a receiver decodes the routing information carried by an incoming packet, it asserts a crossbar request to the appropriate transmitter. If the crossbar request is not granted, it is dropped (and hence the packet will go to the central queue). Each transmitter arbitrates crossbar requests on a least recently served basis. A transmitter will honor no crossbar request if it is already transmitting a packet or if it has packet chunks stored in the central queue. Minimum latency is achieved for packets that use the crossbar.

A schematic diagram of the SP Switch chip is shown in Figure 25 on page 63.

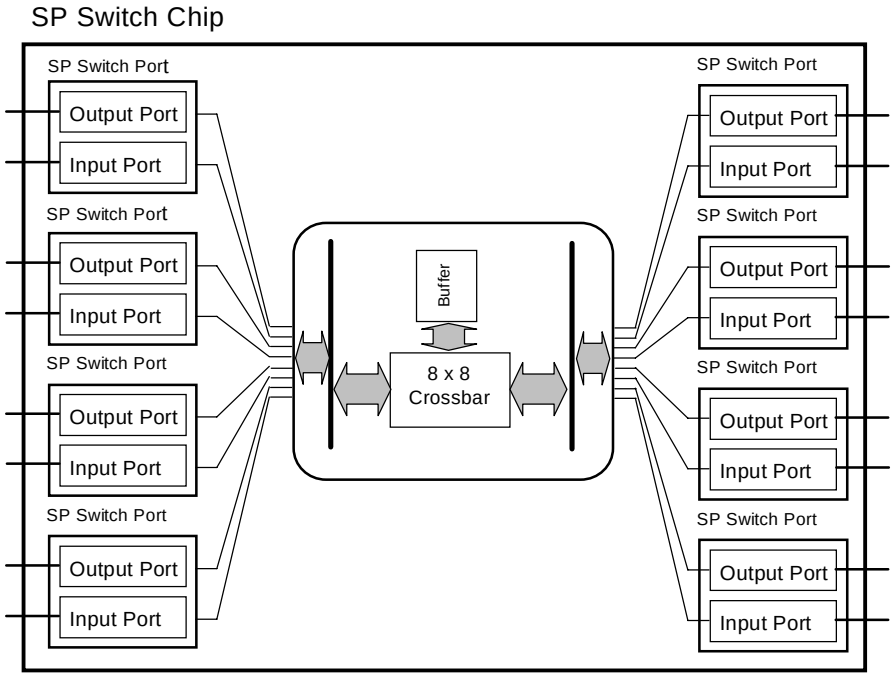


Figure 25. SP Switch Chip Diagram

3.6.1.5 SP Switch Adapter

Another network device that uses an SP Switch Port is the SP Switch adapter. An SP Switch adapter includes one SP Switch Port that is connected to an SP Switch board. The SP Switch adapter is installed in an SP node.

Nodes based on RS/6000s that use the MCA bus use the MCA-based Switch Adapter (#4020). The same adapter is used in uniprocessor Thin, Wide and SMP High nodes.

New nodes based on PCI bus architecture (332 MHz SMP Thin and Wide Nodes, the 200 MHz POWER3 SMP Thin and Wide Nodes) must use the newer MX-based Switch Adapters (#4022 and #4023, respectively) since these are installed on the MX bus in the node. The so-called mezzanine or MX bus allows the SP Switch adapter to be connected directly onto the processor bus, providing faster performance than adapters installed on the I/O bus. The newer (POWER3) nodes use an improved adapter based on a faster mezzanine (MX2) bus.

External nodes such as the 7017-S70 and 7017-S7A are based on standard PCI bus architecture. If these nodes are to be included as part of an SP switch network, then the switch adapter installed in these nodes is a PCI-based adapter (#8396).

Figure 26 shows a schematic diagram of the SP Switch adapter.

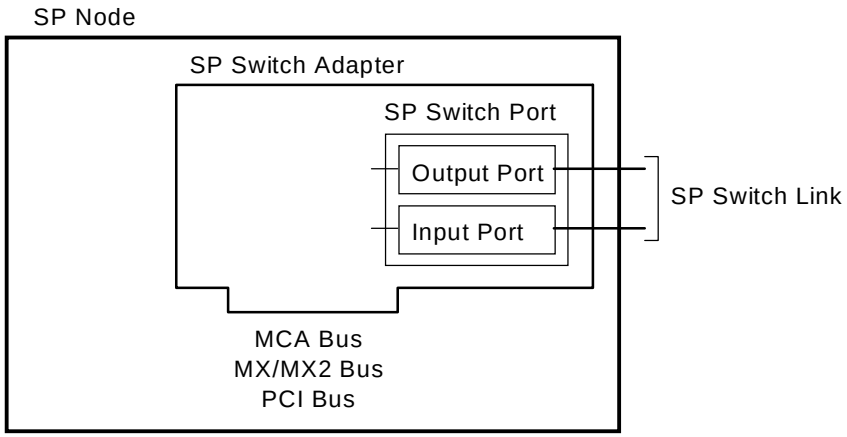


Figure 26. SP Switch Adapter

3.6.1.6 SP Switch System

The SP Switch system in a single frame of an SP is illustrated in Figure 27 on page 65. In one SP frame, there are 16 nodes (maximum) equipped with SP Switch Adapters, and one SP Switch board. Sixteen node SP Switch Adapters are connected to 16 of 32 SP Switch Ports in the SP Switch board. The remaining 16 SP Switch ports are available for connection to other SP Switch Boards.

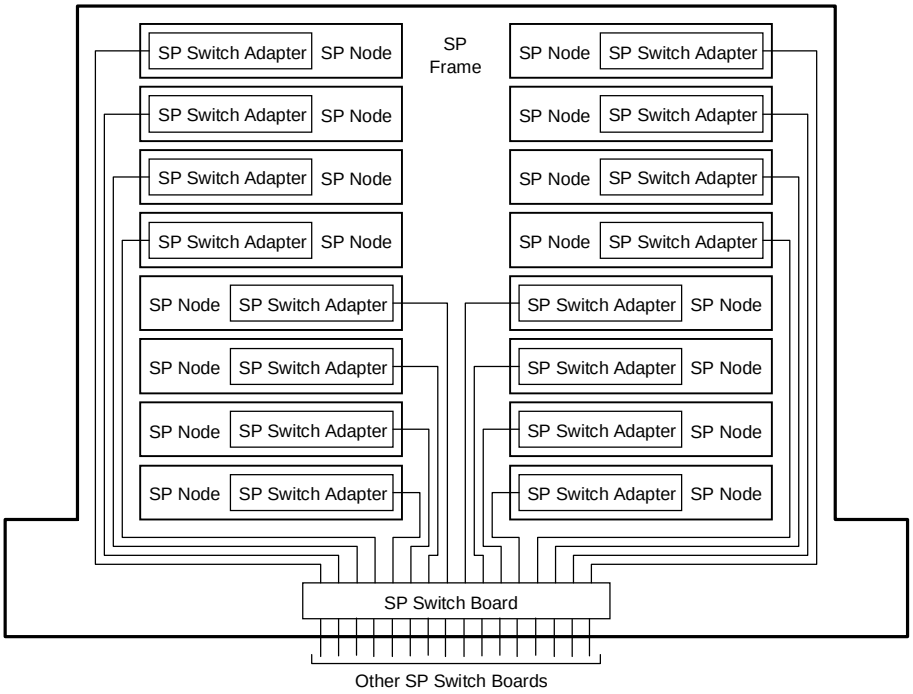


Figure 27. SP Switch System

3.6.2 SP Switch Networking Fundamentals

When considering the network topology of the SP Switch network, nodes are logically ordered into groups of 16 that are connected to one side of the SP Switch Boards. A 16-node SP system containing one SP Switch board is schematically presented in Figure 28 on page 66. This SP Switch board, which connects nodes, is called a node switch board (NSB). This figure also illustrates the possible shortest-path routes for packets sent from node A to two destinations. Node A can communicate with node B by traversing a single SP Switch chip, and with node C by traversing three SP Switch chips.

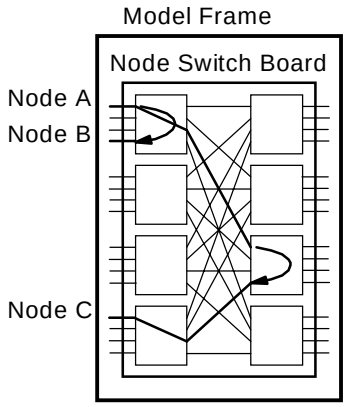


Figure 28. 16-Node SP System

The 16 unused SP Switch ports on the right side of the node switch board are used for creating larger networks. There are two ways to do this:

- For an SP system containing up to 80 nodes, these SP Switch ports connect directly to the SP Switch ports on the right side of other node switch boards.
- For an SP system containing more than 80 nodes, these SP Switch ports connect to additional stages of switch boards. These additional SP Switch boards are known as intermediate switch boards (ISBs).

The strategy for building an SP system of up to 80 nodes is shown in Figure 29 on page 67. The direct connection (made up of 16 links) between two NSBs forms a 32-node SP system. Example routes from node A to node B, C, and D are shown. Just as for a 16-node SP system, packets traverse one or three SP Switch chips when the source and destination pair are attached to the same node switch board. When the source and destination pair are attached to different node switch boards, the shortest path routes contain four SP Switch chips. For any pair of nodes connected to separate SP Switch boards in a 32-node SP system, there are four potential paths, providing a high level of redundancy.

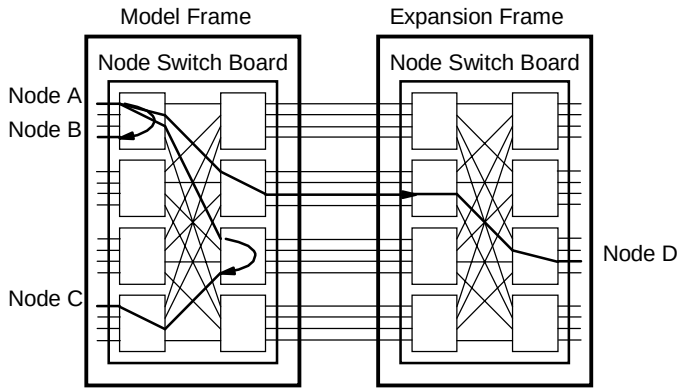


Figure 29. 32-Node SP System

If we now consider an SP system made up of three frames of Thin nodes (48 nodes in total, see Figure 30), we observe that the number of direct connections between frames has now decreased to eight. (Note that for the sake of clarity, not all the individual connections between Switch ports of the NSBs have been shown; instead, a single point-to-point line in the diagram has been used to represent the eight real connections between frames. This simplifying representation will be used in the next few diagrams.) Even so, there are still four potential paths between any pair of nodes that are connected to separate NSBs.

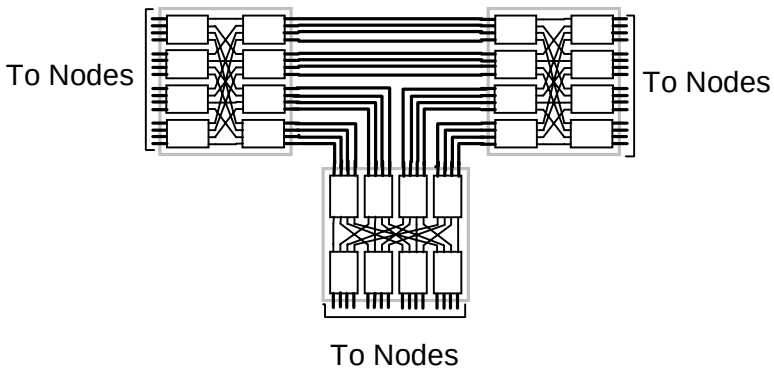


Figure 30. SP 48-Way System Interconnection

Adding another frame to this existing SP complex further reduces the number of direct connections between frames. The 4-frame, 64-way schematic diagram is shown in Figure 31 on page 68. Here, there are at least five connections between each frame, and note that there are six connections

between frames 1 and 2 and between frames 3 and 4. Again, there are still four potential paths between any pair of nodes that are connected to separate NSBs.

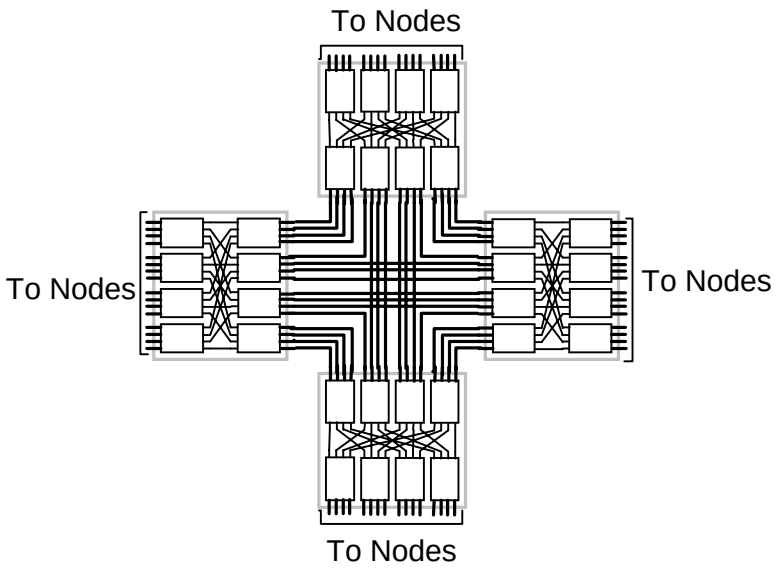


Figure 31. 64-Way System Interconnection

If we extend this 4-frame SP complex by adding another frame, the connections between frames are reduced again (see Figure 32 on page 69); at this level of frame expansion, there are only four connections between each pair of frames. However, there are still four potential paths between any pair of nodes that are connected to separate NSBs.

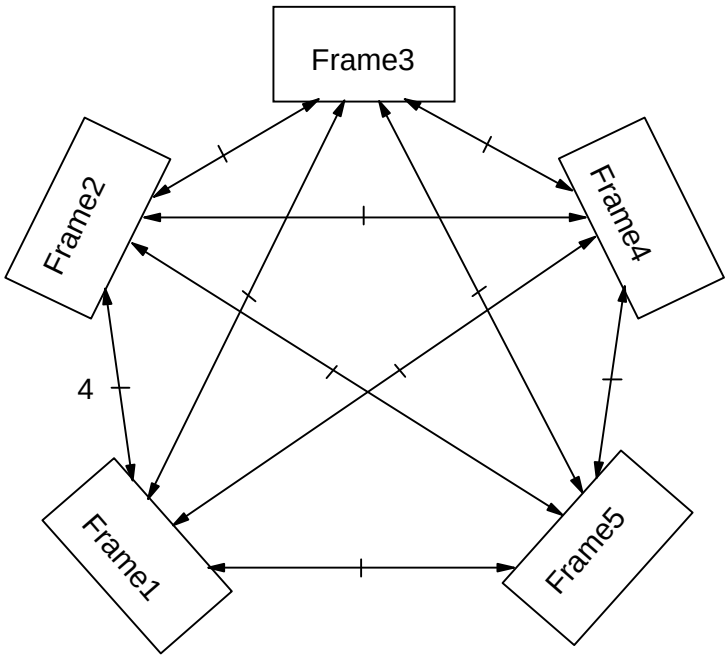


Figure 32. SP 80-Way System Interconnection

The addition of a sixth frame to this configuration would reduce the number of direct connections between each pair of frames to below four. In this hypothetical case, each frame would have three connections to four other frames and four connections to the fifth frame, for a total of 16 connections per frame. This configuration, however, would result in increased latency and reduced switch network bandwidth. Therefore, when more than 80 nodes are required for a configuration, an (ISB) frame is used to provide 16 paths between any pair of frames.

The correct representation of an SP complex made up of six frames with 96 Thin nodes is shown in Figure 33 on page 70. Here we see that all interframe cabling is between each frame's NSB and the switches within the ISB. This cabling arrangement provides for 16 paths between any pair of frames, increasing network redundancy, and allowing the network bandwidth to scale linearly.

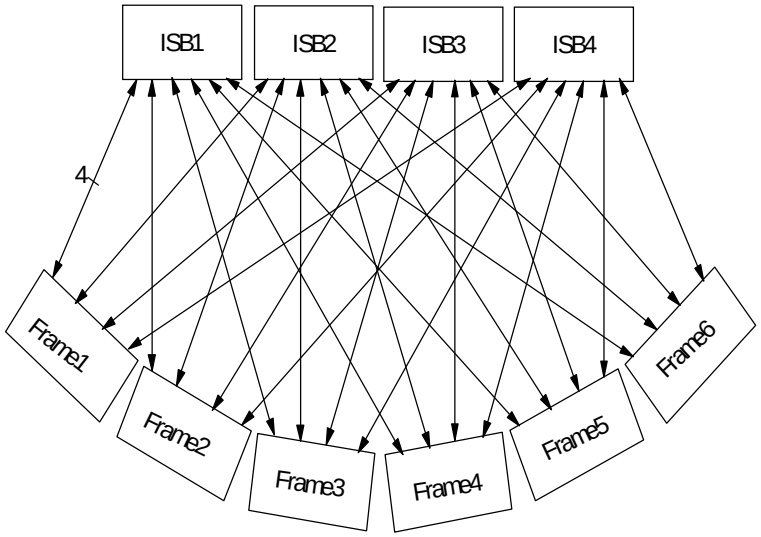


Figure 33. SP 96-way System Interconnection

3.6.3 SP Switch Network Products

Since the original RS/6000 SP product was made available in 1993, there have been two evolutionary cycles in switch technology. The original switch, known as the High Performance Switch (HiPS, feature code #4010), was last supported in Parallel System Support Programs (PSSP) Version 2.4. The latest version of PSSP software (Version 3.1) does not provide support for the HiPS switch. Switch adapters and switches (both 16-port and 8-port) based on the old HiPS technology are no longer available.

3.6.3.1 SP Switch

The operation of the SP Switch (feature code #4011) has been described in the preceding discussion. When configured in an SP order, internal cables are provided to support expansion to 16 nodes within a single frame. In multi-switch configurations, switch-to-switch cables are provided to enable the physical connectivity between separate SP switch boards. The required SP switch adapter connects each SP node to the SP Switch board.

3.6.3.2 SP Switch-8

To meet some customer requirements, eight port switches provide a low cost alternative to the full size 16-port switches. The 8-port SP Switch-8 (SPS-8, feature code #4008) provides switch functions for an 8-node SP system. The

SP Switch-8 is compatible with High nodes. The SP Switch-8 is the only low-cost switch available for new systems.

The SP Switch-8 has two active switch chip entry points. Therefore, the ability to configure system partitions is restricted with this switch. With the maximum eight nodes attached to the switch, there are two possible system configurations:

- A single partition containing all eight nodes
- Two system partitions containing four nodes each

If a switch is configured in an SP system, an appropriate switch adapter is required to connect each RS/6000 SP node to the switch subsystem. Table 3 on page 71 summarizes the switch adapter requirements for particular node types. We have also included here the switch adapter that would be installed in the SP Switch Router. An overview of this dependent node, along with installation and configuration information, can be found in *IBM 9077 SP Switch Router: Get Connected to the SP Switch*, SG24-5157.

Table 3. Supported Switch Adapters

SP Node Type	Supported SP Switch Adapter
160 Mhz Thin, 135 Mhz Wide, or 200 Mhz High	#4020 SP Switch adapter
332 Mhz SMP Thin or Wide Node	#4022 SP Switch MX Adapter
200 Mhz POWER3 SMP Thin or Wide	#4023 SP Switch MX2 Adapter
External, S70 or S7A	#8396 SP System Attachment Adapter
External, SP Switch Router	#4021 SP Switch Router Adapter

The 332 Mhz and 200 Mhz SMP PCI-based nodes listed here have a unique internal bus architecture which allows the SP Switch adapters installed in these nodes to see increased performance compared with previous node types. A conceptual diagram illustrating this internal bus architecture is shown in Figure 34 on page 72.

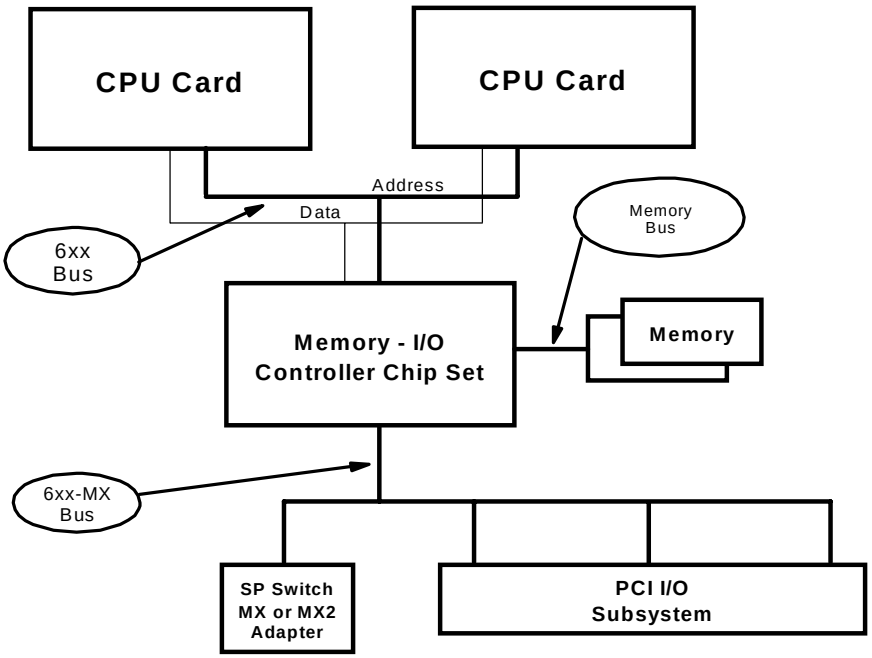


Figure 34. Internal Bus Architecture for PCI-Based SMP Nodes

These nodes implement the PowerPC MP System Bus (6xx bus). In addition, the memory-I/O controller chip set includes an independent, separately clocked "mezzanine" bus (6xx-MX) to which three PCI bridge chips and the SP Switch MX or MX2 Adapter are attached. The major difference between these node types is the clocking rates for the internal buses. The SP Switch Adapters in these nodes plug directly into the MX bus - they do not use a PCI slot. The PCI slots in these nodes are clocked at 33 Mhz. In contrast, the MX bus is clocked at 50 Mhz in the 332 Mhz SMP nodes, and at 60 Mhz in the 200 Mhz POWER3 SMP nodes. Thus, substantial improvements in the performance of applications using the Switch can be achieved.

3.7 Peripheral Devices

The attachment of peripheral devices, such as disk subsystems, tape drives, CD-ROMs, and printers, is very straightforward on the SP. There are no SP-specific peripherals; since the SP uses mainstream RS/6000 node technology, it simply inherits the array of peripheral devices available to the RS/6000 family. The SP's shared-nothing architecture gives rise to two key concepts when attaching peripherals:

1. Each node has I/O slots. Think of each node as a standalone machine when attaching peripherals: it can attach virtually any peripheral available to the RS/6000 family, such as SCSI and SSA disk subsystems, MagStar tape drives, and so on. Peripheral device attachment is very flexible, as each node can have its own mix of peripherals, or none at all.
2. From an overall system viewpoint, as nodes are added, I/O slots are added, thus the scalability of I/O device attachment is tremendous. A 512-node High node system would have several thousand I/O slots!

Each node must have internal disks to hold its copy of the operating system. Since multiple internal disks can be installed in a node, and software loading is provided by the control workstation or boot/install server across the SP administrative Ethernet, it is common to have nodes without any peripheral devices.

When you attach a disk subsystem to one node, is it automatically visible to all the other nodes? The answer is no, but the SP provides a number of techniques and products to allow access to a disk subsystem from other nodes.

There are some general considerations for peripheral device attachment:

- Since the nodes are housed in frames, cable lengths from the I/O adapter in a node slot to the actual peripheral must be long enough.
- Devices such as CD-ROMs and bootable tape drives may be attached directly to SP nodes, but IBM does not support their use as installation devices. Nodes must be network-installed by the control workstation or a boot/install server.
- Graphics adapters for attachment of displays are not supported.

The currently supported adapters for MCA nodes and PCI nodes are listed in Appendix A, “Currently Supported Adapters” on page 517.

3.8 Configuration Rules

The RS/6000 SP system has extremely wide scalability. For a standard configuration, the RS/6000 SP system mounts up to 128 processor nodes. This section provides you with information on how to expand your SP system and what kind of configuration meets your requirements. We also provide a set of rules and sample configurations to facilitate the design of more complex SP configurations. You may use these configuration rules as a check list when you configure your SP system.

This section uses the following node, frame, switch and switch adapter types to configure SP systems:

Nodes

- 160 MHz Thin node (feature code #2022)
- 332 MHz SMP Thin node (feature code #2050)
- 332 MHz SMP Wide node (feature code #2051)
- POWER3 SMP Thin node (feature code #2052)
- POWER3 SMP Wide node (feature code #2053)
- 200 MHz SMP High node (feature code #2009) (This node has been withdrawn from marketing.)
- RS/6000 Server Attached node (feature code #9122)

Frames

- Short model frame (model 500)
- Tall model frame (model 550)
- Short expansion frame (feature code #1500)
- Tall expansion frame (feature code #1550)
- SP Switch frame (feature code #2031)
- RS/6000 server frame (feature code #9123)

Switches

- SP Switch-8 (8-port switch, feature code #4008)
- SP Switch (16-port switch, feature code #4011)

Switch Adapter

- SP Switch adapter (feature code #4020)
- SP Switch MX adapter (feature code #4022)
- SP Switch MX2 adapter (feature code #4023)
- SP System attachment adapter (feature code #8396)

Although the RS/6000 SP configurations are very flexible, there are some basic configuration rules that apply.

Configuration Rule 1

The Tall frame and Short frame cannot be mixed within an SP system.

All frames in an SP configuration must either be Tall frames or Short frames, but not a mixture of both. The SP Switch frame is classified as a Tall frame. You can use an SP Switch frame with Tall frame configurations.

Configuration Rule 2

If there is a single PCI Thin node in a drawer, it must be installed in the odd slot position (left side of the drawer).

With the announcement of the POWER3 SMP nodes in 1999, a single PCI Thin node is allowed to be mounted in a drawer. In this case it must be installed in the odd slot position (left side). This is because the lower slot number is what counts when a drawer is not fully populated. Moreover, different PCI Thin nodes can be mounted in the same drawer, so that you can install a POWER3 SMP Thin node in the left side of a drawer and a 332 MHz Thin node in the right side of the same drawer.

Based on configuration rule 1, the rest of this section is separated into two major parts. The first part provides the configuration rules for using Short frames, and the second part provides the rules for using Tall frames.

3.8.1 Short Frame Configuration

Short frames can be developed into two kinds of configurations: nonswitched and switched. The supported switch for Short frame configurations is SP Switch-8. Only one to eight internal nodes can be mounted in Short frame configurations. The SP-attached servers are not supported in Short frame configurations. Additionally to configuration rule 2, a single PCI Thin node must be the last node in a Short frame.

Configuration Rule 3

A Short model frame must be completely full before a Short expansion frame can mount nodes. You are not allowed any imbedded empty drawers.

3.8.1.1 Nonswitched Short Frame Configuration

This configuration does not have a switch, and it mounts 1 to 8 nodes. A minimum configuration is formed by one Short model frame and one PCI Thin node, or one Wide node, or one High node, or one pair of MCA Thin nodes as shown in Figure 35.

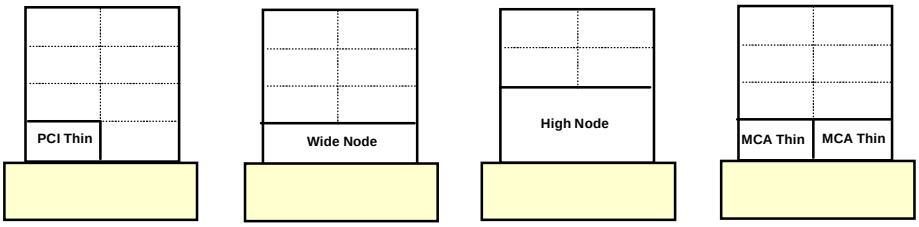


Figure 35. Minimum Nonswitched Short Frame Configurations

The Short model frame must be completely full before the Short expansion frame can mount nodes as shown in Figure 36.

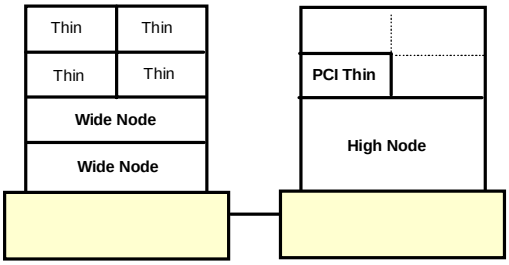


Figure 36. Example of Nonswitched Short Frame Configuration

3.8.1.2 SP Switch-8 Short Frame Configuration

This configuration mounts 1 to 8 nodes and connects through a single SP Switch-8. These nodes are mounted in one required Short model frame containing the SP Switch-8 and an additional nonswitched Short expansion frame. Each node requires a supported SP Switch adapter. Nodes in the nonswitched Short expansion frames share unused switch ports in the Short model frame. Figure 37 on page 77 shows an example of maximum SP Switch-8 Short frame configurations.

Configuration Rule 4

A Short frame supports only a single SP Switch-8 board.

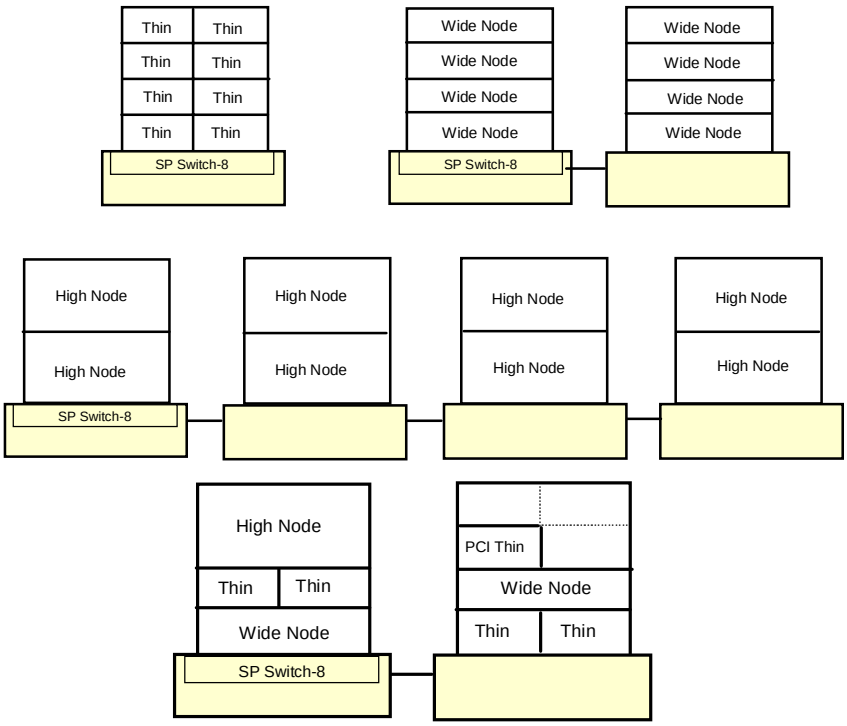


Figure 37. Maximum SP Switch-8 Short Frame Configurations

3.8.2 Tall Frame Configuration

The Tall frame offers several configurations and is more flexible than the Short frame. The SP-attached servers are supported in Tall frame configurations. There are four kinds of Tall frame configurations based on the switch type.

- Nonswitched configuration
- SP Switch-8 configuration
- Single stage SP Switch configuration
- Two-stage SP Switch configuration

Configuration Rule 5

Tall frames support SP-attached servers.

3.8.2.1 Nonswitched Tall Frame Configuration

This configuration does not have a switch. A minimum configuration is formed by one Tall model frame and single PCI Thin node, or one Wide node, or one High node, or one pair of MCA Thin nodes. In contrast to the Short frame configuration, the Tall expansion frame can mount nodes even when the model frame has some empty drawers. It provides more flexibility for adding more nodes in the future.

3.8.2.2 SP Switch-8 Tall Frame Configuration

This configuration mounts 1 to 8 nodes and connects through a single SP Switch-8. A minimum configuration is formed by one Tall model frame equipped with an SP-Switch-8 and a single PCI Thin node, or one Wide node, or one High node, or one pair of MCA Thin nodes. Each node requires a supported SP Switch adapter. A nonswitched Tall expansion frame may be added. Nodes in an expansion frame share unused switch ports in the model frame. You are not allowed any imbedded empty drawers. Again, if there is a single PCI Thin node in a drawer it must be placed at the last node in a frame. Figure 38 shows examples of SP Switch-8 Tall frame configurations.

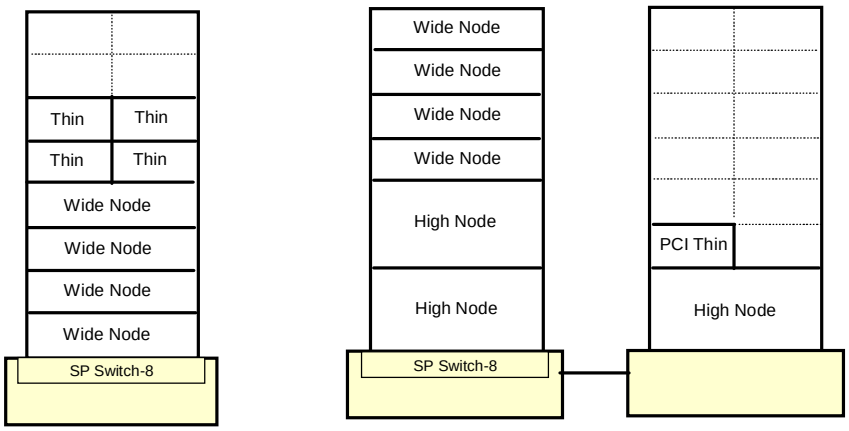


Figure 38. Example of SP Switch-8 Tall Frame Configurations

Ex.1010 / Page 102 of 572

3.8.2.3 Single Stage SP Switch Configuration

This is probably the most common SP configuration. It provides both scalability and flexibility. This configuration can mount 1 to 80 processor nodes in one required Tall model frame with an SP Switch and additional switched and/or nonswitched expansion frames. A minimum configuration is formed by one Tall model frame equipped with an SP Switch and single PCI Thin node, or one Wide node, or one High node, or one pair of MCA Thin node. Each node requires a supported SP Switch adapter. Empty drawers are allowed in this configuration.

Single Stage SP Switch with Single SP-Switch Configuration

If your SP system has no more than 16 nodes, a single SP Switch is enough to fulfill the system requirements. In this circumstance, nonswitched expansion may be added depending on the number of nodes and node locations (see 3.9.4, “The Switch Port Numbering Rule” on page 86 and Figure 44 on page 88).

Figure 39 on page 81 shows several examples of single stage SP Switch configurations with no more than 16 nodes.

In configuration (a), four Wide nodes and eight Thin nodes are mounted in a Tall model frame equipped with an SP Switch. There are 4 available switch ports which you can use to attach SP-attached servers or SP Switch routers. Expansion frames are not supported in this configuration because there are Thin nodes on the right side of the model frame.

Configuration Rule 6

If a model frame on switched expansion frame has Thin nodes on the right side, it cannot support nonswitched expansion frames.

In configuration (b), six Wide nodes and two PCI Thin nodes are mounted in a Tall model frame equipped with an SP Switch. There also are a High node, two Wide nodes, and four PCI Thin nodes mounted in a nonswitched expansion frame. Note that all PCI Thin nodes on the model frame must be placed on the left side to comply with configuration rule 6. All Thin nodes on an expansion frame must also be placed on the left side to comply with the switch port numbering rule. There is one available switch port which you can use to attach SP-attached servers or SP Switch routers.

In configuration (c), there are eight Wide nodes mounted in a Tall model frame equipped with an SP Switch and four High nodes mounted in a nonswitched expansion frame (frame 2). The second nonswitched expansion

Ex.1010 / Page 103 of 572

frame (frame 3) houses a High node, two Wide nodes and one PCI Thin node. This configuration occupies all 16 switch ports in the model fame. Note that the Wide nodes and the PCI Thin node in frame 3 have to be placed on High node locations.

Now, you can try to describe configuration (d). If you want to add two POWER3 Thin nodes, what would be the locations?

A maximum of three nonswitched expansion frames can be attached to each model frame and switched expansion frame.

Ex.1010 / Page 104 of 572

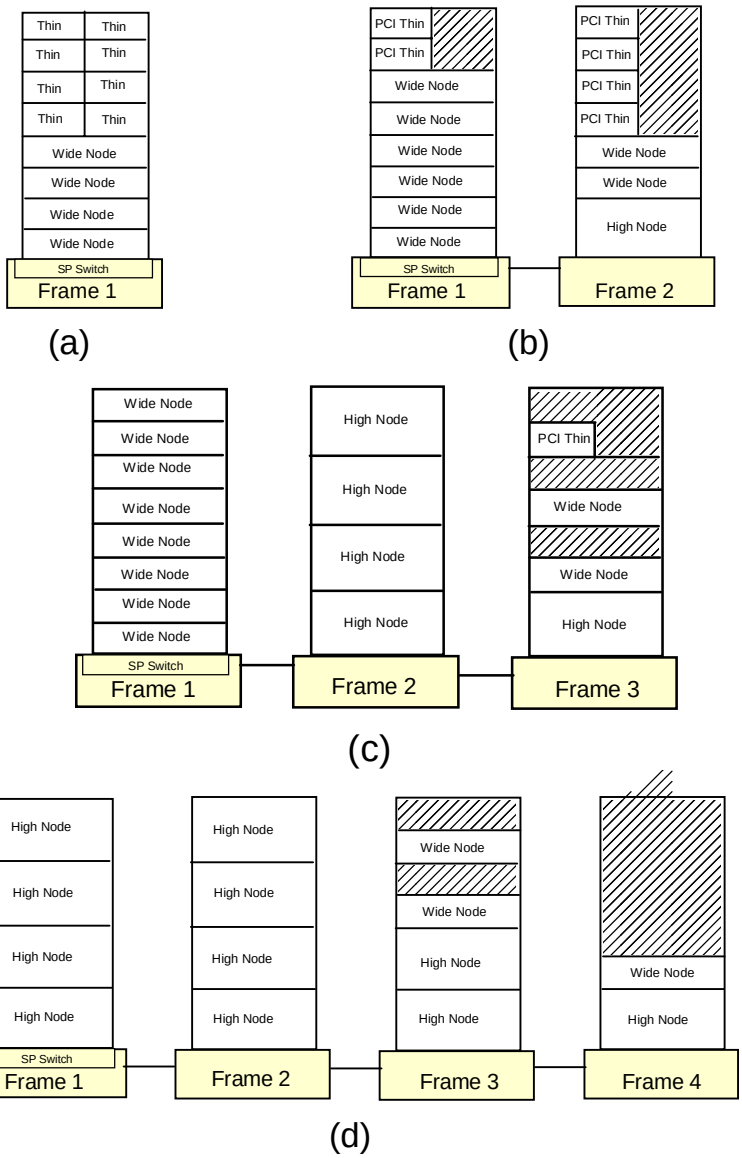


Figure 39. Example of Single SP-Switch Configurations

Single Stage with Multiple SP-Switches Configuration
If your SP system has 17 to 80 nodes, switched expansion frames are required. You can add switched expansion frames and nonswitched expansion frames. Nodes in the nonswitched expansion frame share unused switch ports that may exist in the model frame and in the switched expansion frames. Figure 40 shows an example of a Single Stage SP Switch with both switched and nonswitched expansion frame configurations. There are four SP Switches, each can support up to 16 processor nodes. Therefore, this example configuration can mount a maximum of 64 nodes.

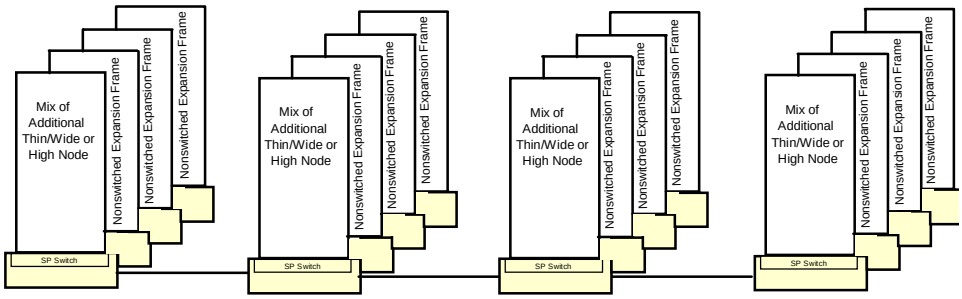


Figure 40. Example of Multiple SP-Switches Configuration

3.8.2.4 Two-Stage SP Switch Configuration

This configuration shown in Figure 41 on page 83, requires an SP Switch frame which forms the second switching layer. A minimum of 24 processor nodes is required to make this configuration work. It supports up to 128 nodes. Each node requires a supported SP Switch adapter. These nodes are mounted in one required Tall model frame equipped with an SP Switch and at least one switched expansion frames. The SP Switch in these frames forms the first switching layer. The SP Switch frame is also required if you want more than 80 nodes or more than four switched expansion frames. This configuration can utilize both switched and nonswitched expansion frames as well. Nodes in the nonswitched expansion frame share unused switch ports that may exist in the model frame.

Ex.1010 / Page 106 of 572

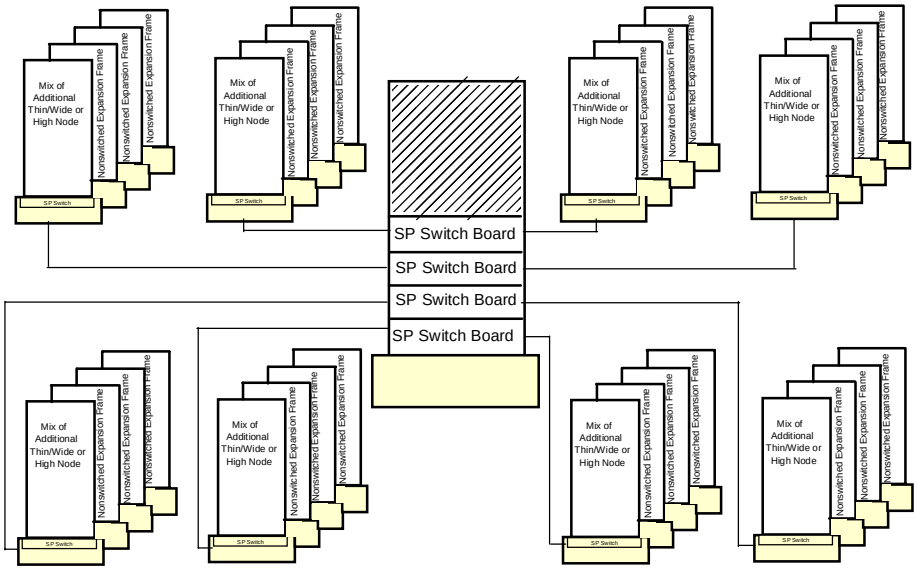


Figure 41. Example of Two-Stage SP Switch Configuration

3.9 Numbering Rules

In order to place nodes in an SP system, you need to know the following numbering rules:

- The frame numbering rule
- The slot number rule
- The node numbering rule
- The SP Switch port numbering rule

3.9.1 The Frame Numbering Rule

The administrator establishes the frame numbers when the system is installed. Each frame is referenced by the tty port to which the frame supervisor is attached, and is assigned a numeric identifier. The order in which the frames are numbered determines the sequence in which they are examined during the configuration process. This order is used to assign global identifiers to the switch ports and nodes. This is also the order used to determine which frames share a switch.

If you have an SP Switch frame, you must configure it as the last frame in your SP system. Assign a high frame number to an SP Switch frame to allow for future expansion.

3.9.2 The Slot Numbering Rule

A Tall frame contains eight drawers which have two slots each for a total of 16 slots. A Short frame has only four drawers and eight slots. When viewing a Tall frame from the front, the 16 slots are numbered sequentially from bottom left to top right.

The position of a node in an SP system is sensed by the hardware. That position is the slot to which it is wired. That slot is the slot number of the node.

- A Thin node occupies a single slot in a drawer and its slot number is the corresponding slot.
- A Wide node occupies two slots and its slot number is the odd-numbered slot.
- A High node occupies four consecutive slots in a frame. Its slot number is the first (lowest number) of these slots.

Figure 42 on page 85 shows slot numbering for Tall frame and Short frame.

An SP-attached server is managed by the PSSP components, as it is in a frame of its own. However, it does not enter into the determination of the frame and switch configuration of your SP system. It has the following additional characteristics:

- It is the only node in its frame. It occupies slot number 1 but uses the full 16 slot numbers. Therefore, 16 is added to the node number of the SP-attached server to get the node number of the next node.
- It cannot be the first frame.
- It connects to a switch port of a model frame or a switched expansion frame.
- It cannot be inserted between a switched frame and any nonswitched expansion frame using that switch.

Ex.1010 / Page 108 of 572

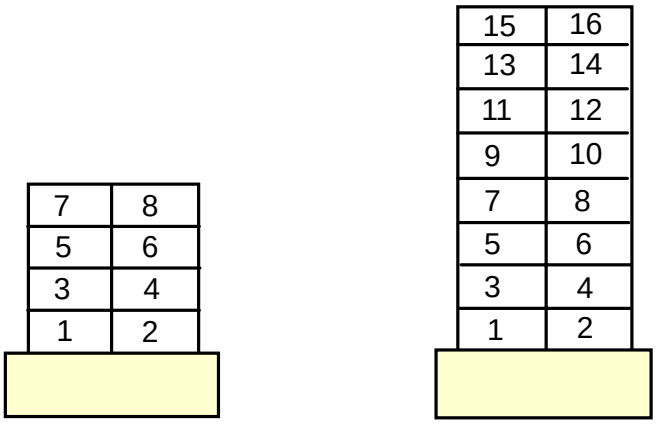


Figure 42. Slot Numbering for Short Frame and Tall Frame

3.9.3 The Node Numbering Rule

A *node number* is a global ID assigned to a node. It is the primary means by which an administrator can reference a specific node in the system. Node numbers are assigned for all nodes, including SP-attached servers, regardless of node or frame type by the following formula:

node_number = ((*frame_number* - 1) x 16) + *slot_number*

where *slot_number* is the lowest slot number occupied by the node. Each type (size) of node occupies a consecutive sequence of slots. For each node, there is an integer n such that a Thin node occupies slot *n*, a Wide node occupies slots *n, n+1* and a High node occupies *n, n+1, n+2, n+3*. For Wide and High nodes, n must be odd.

Node numbers are assigned independent of whether the frame is fully populated. Figure 43 on page 86 demonstrates node numbering. Frame 4 represents an SP-attached server in a position where it does not interrupt the switched frame and companion nonswitched expansion frame configuration. It can use a switch port on frame 2, which is left available by the High nodes in frame 3. Its node number is determined by using the previous formula.

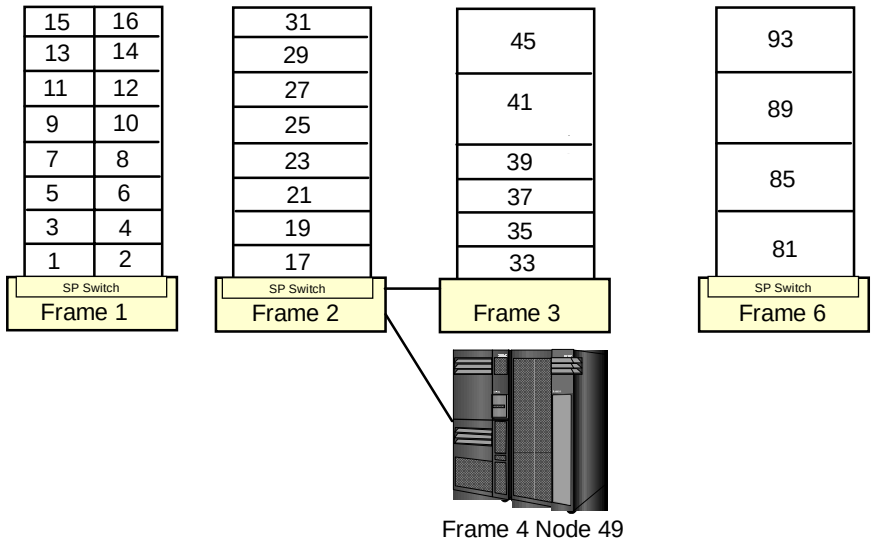


Figure 43. Node Numbering for an SP System

3.9.4 The Switch Port Numbering Rule

In a switched system, the switch boards are attached to each other to form a larger communication fabric. Each switch provides some number of ports to which a node can connect (16 ports for an SP Switch, and 8 ports for the SP Switch-8.) In larger systems, additional switch boards (intermediate switch boards) in the SP Switch frame are provided for switch board connectivity; such boards do not provide node switch ports.

Switch boards are numbered sequentially starting with 1 from the frame with the lowest frame number to that with the highest frame number. Each full switch board contains a range of 16 switch port numbers (also known as switch node numbers) that can be assigned. These ranges are also in sequential order with their switch board number. For example, switch board 1 contains switch port numbers 0 through 15.

Switch port numbers are used internally in PSSP software as a direct index into the switch topology and to determine routes between switch nodes.

Switch Port Numbering for an SP Switch

The SP Switch has 16 ports. Whether a node is connected to a switch within its frame or to a switch outside of its frame, you can evaluate the following formula to determine the switch port number to which a node is attached:

switch_port_number = ((*switch_number* - 1) x 16) + *switch_port_assigned*

where *switch_number* is the number of the switch board to which the node is connected and *switch_port_assigned* is the number assigned to the port on the switch board (0 to 15) to which the node is connected.

Figure 44 on page 88 shows the frame and switch configurations that are supported and the switch port number assignments in each node. Let us describe more details on each configuration.

In configuration 1, the switched frame has an SP Switch that uses all 16 of its switch ports. Since all switch ports are used, the frame does not support nonswitched expansion frames.

If the switched frame has only Wide nodes, it could use at most eight switch ports, and so has eight switch ports to share with nonswitched expansion frames. These expansion frames are allowed to be configured as in configuration 2 or configuration 3.

In configuration 4, four High nodes are mounted in the switched frame. Therefore, its switch can support 12 additional nodes in nonswitched expansion frames. Each of these nonswitched frames can house a maximum of four High nodes. If Wide nodes are used, they must be placed in the High node slot positions.

A single PCI Thin node is allowed to be mounted in a drawer. Therefore, it can be mounted in nonswitched expansion frames. In this case, it must be installed in the Wide node slot positions (configuration 2) or High node slot positions (configurations 3 and 4).

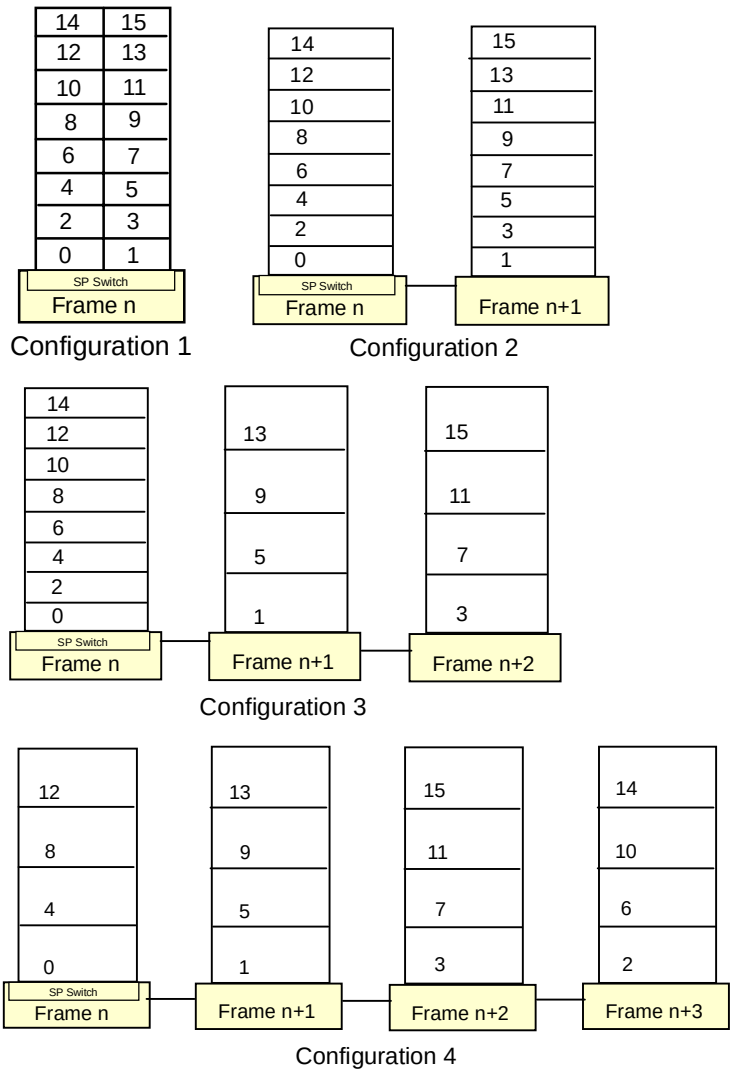


Figure 44. Switch Port Numbering for an SP Switch

Switch Port Numbering for an SP Switch-8

An SP system with SP Switch-8 contains only switch port numbers 0 through 7. The following algorithm is used to assign nodes their switch port numbers in systems with eight port switches:

1. Assign the node in slot 1 to switch_port_number = 0. Increment switch_port_number by 1.
2. Check the next slot. If there is a node in the slot, assign it the current switch_port_number, then increment the number by 1.

   Repeat until you reach the last slot in the frame or switch port number 7, whichever comes first.

Figure 45 shows sample switch port numbers for a system with a Short frame and an SP Switch-8.

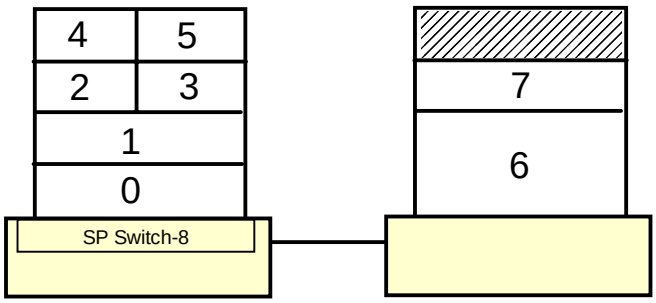


Figure 45. Example of Switch Port Numbering for an SP Switch-8

Chapter 4. System Data Repository and System Partitioning

The System Data Repository (SDR) is a database that stores SP system configuration information. It is a central repository of data that only resides on the control workstation. The SDR makes this data available to the nodes in an SP complex by using a client/server interface.

The System Data Repository daemon *sdrd* is an SRC-managed subsystem that manages the SP system configuration and some operational information such as:

- Information about frames, nodes and switches and how they are configured
- Information about partitions, and their configurations
- VSD information (if applicable)
- The current values of host_responds and switch_responds, two indicators of the health of a node

System partitioning is a method for organizing an SP system into non-overlapping groups of nodes that can be dedicated to specific purposes, such as separate test and production environments. A group of such nodes (not including the control workstation) is called a *system partition*.

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. Isolation of these logical SP systems is achieved by partitioning the switch in such a way that switch communication paths for different system partitions do not cross each other. Of course, normal TCP/IP-based communication between nodes is still possible using the SP Ethernet or other external networks.

The reason these two topics are presented together in this chapter is because an SP system is, by default, partitioned, albeit in most installations there is only one partition. This fact is reflected in the data structures that make up the SDR, and in the way that nodes communicate with the control workstation to retrieve data from the SDR, since they all are in at least one partition.

The following topics are covered in this chapter.

- The SDR Data Model
- The SDR Daemon
- User Interface to the SDR
- Managing the SDR (log files, backup and recovery)

© Copyright IBM Corp. 1999

4.1 SDR Data Model

The PSSP installation procedures create the directory structure and the data files that make up the SDR, and populate it with an initial set of data that verifies its proper installation. The process of configuring and customizing your site-specific environment and node information adds entries to the SDR that are critical for the correct installation and management of the nodes that make up the SP complex.

The SDR data model consists of classes, objects and attributes. Classes contain objects. Objects are made up of attributes. Objects have no unique ID, but their combined attributes must be unique among other objects. Attributes can be one of three types: strings, floating-point numbers or integers.

A class can represent a type of physical device or feature (for example, an adapter, a frame, a node), and it can also represent a logical set of information relevant to the SP configuration (for example, how system partitions may be assigned, or how volume groups have been set up for specific nodes).

The attributes are the details about a class. For example, some of the attributes of the Adapter class are: node_number, adapter_type, netaddr.

An object is made up of a specific set of attributes within a class. For example, within the Adapters class, all the information (the attributes) that belongs to the entry for the node whose node_number is 3 make up that object. 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.

Two examples of specific classes within the SDR (the Adapter class and the SP Class) are compared in Figure 46 on page 93.

Ex.1010 / Page 116 of 572

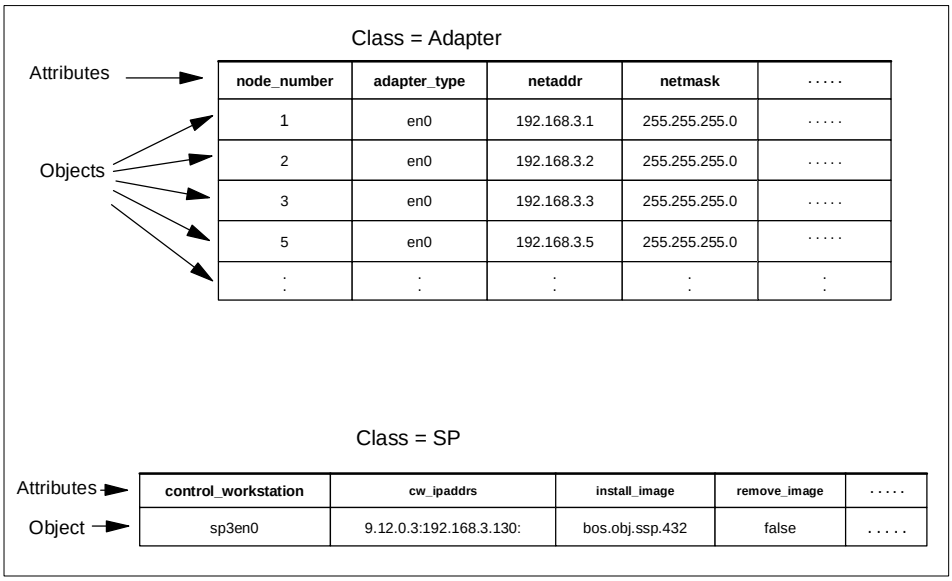


Figure 46. Sample SDR Classes

From these examples, we observe that the Adapter class has many objects stored within it, whereas the SP class has only one object. As we have previously mentioned, the attributes that belong to each object have a datatype associated with them, and the datatypes can have one of three values:

1. String. For example, the control_workstation attribute within the SP class is a string consisting of the hostname of the control workstation.
2. Integer. The node_number attribute of the Adapter class has an integer value.
3. Floating Point. There are currently no attributes of this datatype in the SDR.

The SDR stores its data within the spdata/sys1/sdr directory on the control workstation. The complete SDR directory structure is presented in Figure 47 on page 94.

Ex.1010 / Page 117 of 572

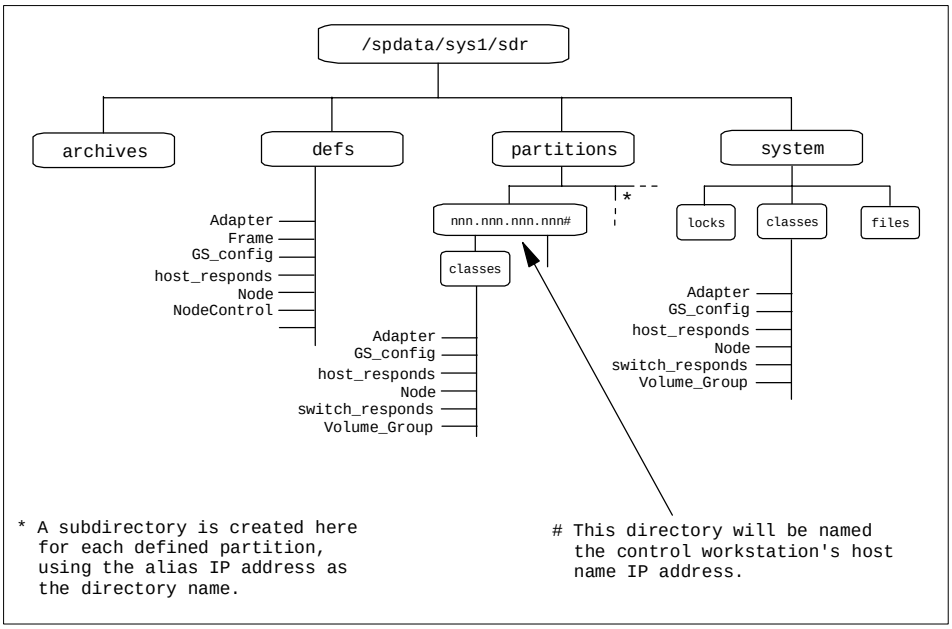


Figure 47. SDR Directory Structure

The SP system (more specifically, the PSSP software) 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. Regardless of whether you use system partitioning or not, there is always one system partition defined in an SP system, the *default* partition that consists of all the nodes in the SP complex.

Therefore, there are two types of SDR classes: system and partitioned.

- System classes contain objects that are common to all system partitions.
- Partitioned classes contain objects that are specific to a partition. The *definition* of a partitioned class is common to all system partitions, but each system partition contains different objects for the class.

The SDR directory structure shows us there are only eight classes that are common to all system partitions (the existence of some of these classes depends on your specific SP configuration). The remaining classes are duplicated for each partition that is defined. The subdirectory name for the default partition is the IP address associated with the en0 adapter of the control workstation. Any additional partitions defined will use the *alias* IP address of the en0 adapter of the control workstation.

The contents of the SDR directory structure are:

/spdata/sys1/sdr/archives
Contains archived copies of the SDR for backup purposes. A snapshot of the contents of the SDR directories are saved in a tar file, and this file can be used to restore the SDR to a known working state.

/spdata/sys1/sdr/defs
Contains the header files for all the object classes (system and partitioned). Each file describes the fields and attributes that are used to define the object of the class.

/spdata/sys1/sdr/partition
An /spdata/sys1/sdr/partition/*<syspar_ip_addr>* subdirectory is maintained for each partition. There will always be at least one subdirectory here, corresponding to the default partition. Object classes are replicated for each partition and each class keeps information on the objects specific to the partition.

/spdata/sys1/sdr/system
This subdirectory contains classes and files that are common to all partitions within an SP system. Some classes exist in this subdirectory only if explicitly created; for example, the SysNodeGroup Class will only be present if node groups are defined.

Refer to Appendix G, *IBM Parallel System Support Program for AIX: Administration Guide,* SA22-7348 for more details on classes, objects, and attributes.

4.2 Partitioning the SP System

A system partition is a static entity that contains a specified subset of SP nodes with a consistent software environment. System partitioning lets you divide system resources to make your SP system more efficient and more tailored to your needs, while keeping system management simple and efficient. At the most elementary level, a system partition is a group of nodes working together.

Systems with switches are assumed to be used in performance-critical parallel computing. One major objective in partitioning a system with a switch is to keep the switch communication traffic in one switch partition from interfering with that of another. In order to ensure this, each switch chip is placed completely in one system partition.

Any link which joins switch chips in different partitions is disabled, so traffic in one partition cannot enter the physical bounds of another partition through the SP Switch.

The control workstation does not belong to any system partition. Instead, it provides a central control mechanism for all partitions that exist on the system.

4.2.1 Partitioning Rules

There are basic rules that must be followed when you set up partitions. These rules impose some limitations on the configurations that can be selected for the partitions.

- Nodes in a partition must not cross the switch chip boundaries - all the nodes connected to a switch chip must be in the same partition. The nodes have predetermined fixed connectivity with each switch board. For example, in a single-frame, single-switch system, with two high nodes in slots 1 and 5 (so they are connected to the same switch chip), it would not be possible to have these two nodes in different partitions.
- HACMP nodes must be in the same partition if they are running HACMP 4.2.1 or earlier. Nodes running HACMP 4.3 are not bound by this restriction.
- Virtual Shared Disks (VSDs) cannot span partitions.

System partitioning can be achieved by applying one of a fixed set of supplied configurations. The ssp.top fileset 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 fileset makes available the System Partitioning Aid, a tool for creating customized system partitions to generate other configurations.

More information about the implications of SP System Partitioning and the SP Switch is found in 9.10, “System Partitioning and the SP Switch” on page 249.

4.2.2 Partitions and Switchless SP Systems

Partitioning is also applicable to switchless systems. If you have a switchless system, and later add a switch, you might have to rethink your system partition choice. In fact you might want to reinstall ssp.top so that any special switchless configurations you have constructed are removed from the system.

If you choose one of the supplied layouts, your partitioning choice is "switch smart": your layout will still be usable when the switch arrives. This is

because the predefined layouts are constrained to be usable in a system with a switch. Such a layout might be unsatisfactory, however, for your switchless environment, in which case you can use the System Partitioning Aid to build your own layout.

4.2.3 Why Partition

When partitioning was first made available in PSSP 2.1, the advantages that were offered to system administrators included:

- The ability to run all types of work on a set of nodes without interfering with any work on another set, regardless of application or node failures.
- The ability to separate a test area for application development from your production area.
- The ability to install and test new releases and migration plans without affecting current work.
- The ability to isolate switch traffic and prevent it from affecting switch traffic in another system partition.
- The ability to have one operator manage, at a system level, more than one logical system.

With the introduction of coexistence support in PSSP 2.2, and the availability of the SP Switch in 1996, which allowed the isolation of switch faults to the failing node alone, the need to consider partitioning as a possible solution to these operational requirements has been reduced.

Under PSSP 3.1, coexistence is supported in the same system partition or a single default system partition (the entire SP system) for nodes running any combination of:

- PSSP v3.1 and AIX 4.3.2
- PSSP v2.4 and AIX 4.2.1 or 4.3
- PSSP v2.3 and AIX 4.2.1 or 4.3
- PSSP v2.2 and AIX 4.1.5 or 4.2.1

Attention

The control workstation must always be at the highest level of PSSP and AIX that is used by the nodes.

4.3 SDR Daemon

The daemon that handles the SDR queries and updates is called sdrd. This daemon is controlled by System Resource Controller (SRC), and so has respawn protection: if the daemon dies, it is automatically restarted by SRC.

There is one SDR daemon for each partition. Each daemon is uniquely identified by an IP address. In the case of the default partition, this is the IP address associated with the *hostname interface* of the control workstation. For other partitions, this address is the alias IP address that was established for this same interface. The startup process passes the IP address associated with the partition to the respective daemon.

Multiple SDR daemons are started by use the group option with the `startsrc` command.

4.3.1 Client/Server Communication

A review of the /etc/services file shows that SDR daemons are listening on TCP port 5712. Each node can connect to this port by specifying the IP address associated with its partition (the IP address alias or the hostname of the control workstation), and making a socket connection. In this way, each SDR daemon on the control workstation is able to communicate only with the nodes that have been defined in its partition.

When a node is initially booted, the /etc/rc.sp script, started from inittab, looks for the environment variable SP_NAME to retrieve the IP address of the partition of which it is a member. The SP_NAME variable is not set by default, and if the system administrator has a requirement to execute partition-sensitive commands, then this variable has to be explicitly set and exported. If it is not set, the /etc/SDR_dest_info file is referenced.

This file contains the IP address and name for the *default* and *primary* partitions. The contents of this file extracted from a node are shown.

```
[sp4n01:/]# cat /etc/SDR_dest_info
default:192.168.4.140
primary:192.168.4.140
nameofdefault:sp4en0
nameofprimary:sp4en0
[sp4n01:/]#
```

The default stanza is set to the IP address associated with the default system partition (the hostname of the control workstation). The primary stanza is set

to the IP address given to the partition that this node belongs to. The default stanza in the /etc/SDR_dest_info file serves two purposes.

1. Whenever the node is rebooted, the /etc/rc.sp script that runs from inittab performs the following steps:
   - It retrieves the default IP address from the /etc/SDR_dest_info file on that node.
   - It contacts the SDR (using the default stanza for the control workstation IP address) and queries the Syspar_map class to find out whether the primary IP address value in its own /etc/SDR_dest_info file for the primary partition is correct. The Syspar_map is a *system-wide* class that describes the mapping of nodes onto partitions. It contains the following attributes (with some sample entries):

```
syspar_name  syspar_addr  node_number  switch_node_number used node_type
sp4en0       192.168.4.140        1             0            1 standard
sp4en0       192.168.4.140        2             1            0 standard
sp4en0       192.168.4.140        3             2            0 standard
sp4en0       192.168.4.140        4             3            0 standard
sp4en0       192.168.4.140        5             4            1 standard
sp4en0       192.168.4.140        6             5            1 standard
sp4en0       192.168.4.140        7             6            1 standard
sp4en0       192.168.4.140        8             7            1 standard
```

   - If the value of the syspar_addr attribute corresponding to that node is different from the IP address entry retrieved from the /etc/SDR_dest_info file, then the node updates the /etc/SDR_dest_info file with the value obtained from the Syspar_map object.
   - This is a critical step in getting the partitioning to work and is one of the reasons why the affected nodes should be shut down. The reboot of the node ensures that the /etc/SDR_dest_info file is updated with the new IP address that corresponds to the node's partition.
2. If the partition to which a node is assigned is deleted by the restoration of an archive of the SDR or by the application of a new partitioning scheme, then the node can still reach the SDR daemon on the control workstation by using this IP address (after failing to connect to the primary).

An example of how a node in the default partition interacts with the SDR daemon sdrd to get access to partition information is shown in Figure 48 on page 100.

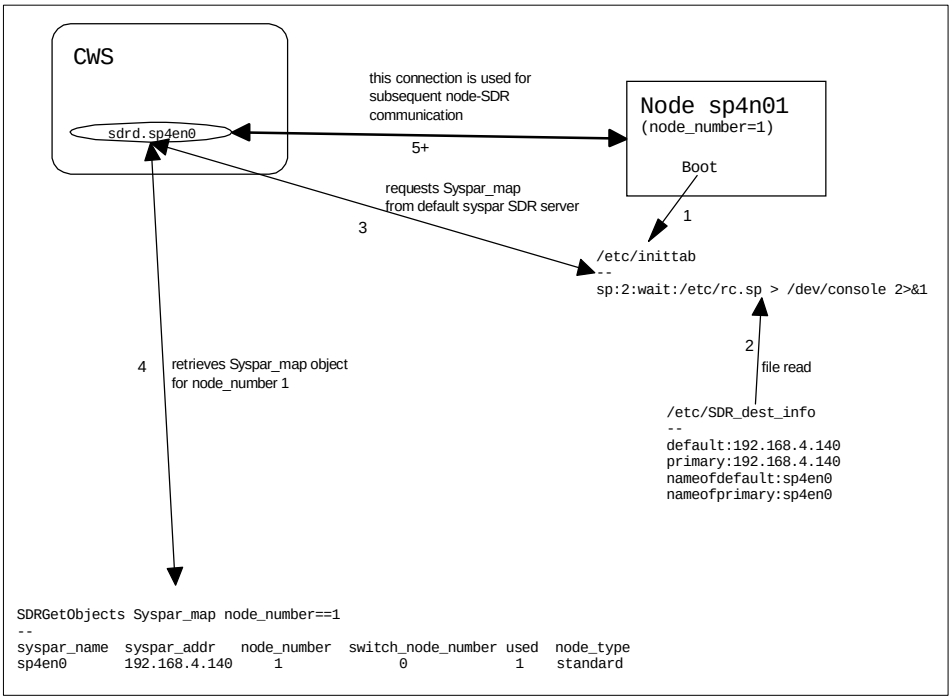


Figure 48. Node<->sdrd Communication Process (Non-Partitioned Example)

A reboot (or an initial boot) of the node results in the following (high-level) steps being carried out.

1. During /etc/inittab processing, the /etc/rc.sp script is executed. Among other actions, this script reads the contents of the /etc/SDR_dest_info that resides on the node.
2. The SP_NAME environment variable is set to the default address that it finds in this file.
3. Using an SDR API call, the node makes a request to the sdrd daemon on the control workstation for the Syspar_map object corresponding to its node_number.
4. The value of the primary field of the node's /etc/SDR_dest_info file is compared with the value of syspar_addr attributes returned from the Syspar_map class in the SDR. If the values differ, then the primary field is updated with the syspar_addr value.

   This step is repeated for the default, *nameofprimary* and *nameofdefault* stanzas. The latter two stanzas's fields are compared with the value of the syspar_name attribute.

5. In this case, none of the fields in the node's /etc/SDR_dest_info file differ from the values returned from the SDR on the control workstation - the node is in its correct partition, the default partition.

We can use a similar logic flow to consider the case where an SP system has been partitioned on the control workstation. To simplify this example, we assume that an alias IP address has been set up for the control workstation, and that the hostname corresponding to the alias has been added to the /etc/hosts file. The system has been partitioned in two: one partition consists of nodes in slots 1 to 8, and the second partition consists of nodes in slots 9 to 16.

When the node in the second partition is being rebooted, the logic flow is as shown in Figure 49.

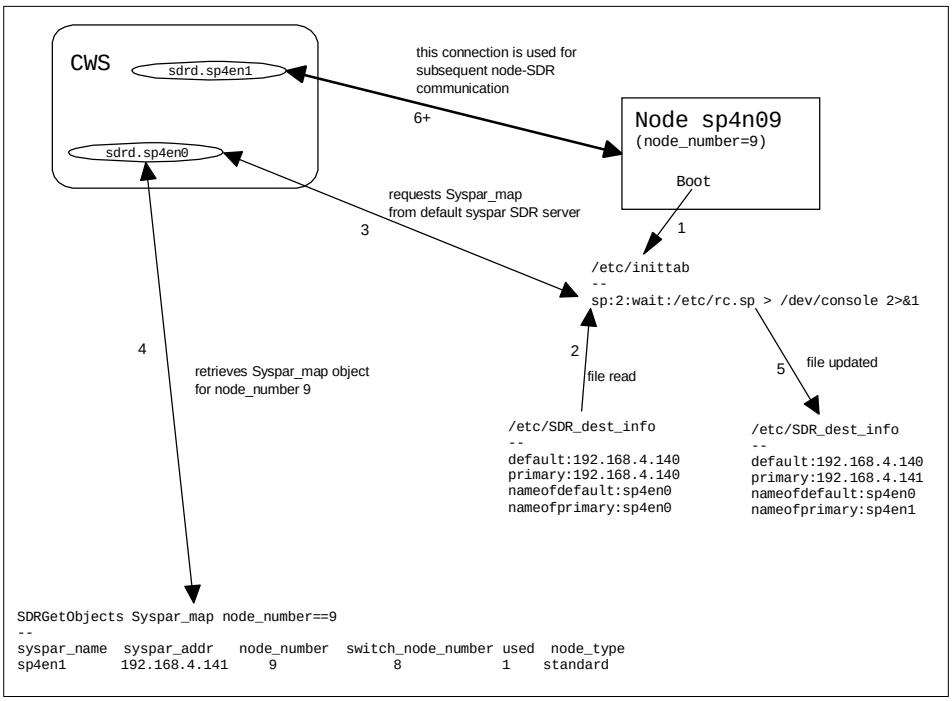


Figure 49. Node<->sdrd Communication Process (Partitioned Example)

In this example, we see that the node recognizes that it is now in a new partition. It updates the /etc/SDR_dest_info file (the primary and nameofprimary stanzas) with the address and hostname of the partition that it now belongs to. It uses these fields to establish communication with the appropriate sdrd daemon on the control workstation.

Ex.1010 / Page 125 of 572

4.3.2 Locking

Now that there is the potential to have multiple SDR daemons accessing the same system-wide or global object classes, there is the requirement to have a persistent locking mechanism to prevent simultaneous updates of object classes. The lock files are created in the /spdata/sys1/sdr/system/locks directory. The file that is created contains details of the node updating the SDR and the partition to which it belongs.

Whenever the SDR daemons are started up, they check for the presence of any lock files in case they were left behind when an update of the SDR ended abnormally. If any are found at this point, they are removed. This is another reason why it is a good idea to recycle the SDR daemons when there are apparent SDR problems.

4.3.2.1 Holding Locks

SDR clients (the SDR command-line routines and so on) can lock an entire class to get exclusive write access. While the client holds the lock, other clients can read the data as it was before the lock, but cannot write to the class. If the client that holds the lock fails to unlock the class before it exits, the server releases the lock without making any changes to the class. The only cases where a class may be locked and the server can release the lock are the following:

- The client that has the class locked is stuck in an infinite loop.
- The node which the client is running crashes or is powered down. The SDR server is not notified of the broken connection until the connection times out. This occurs after the time indicated in the *tcp_keepidle* variable of the no command output. The default is 14,400 half-second increments, or two hours.

Only when an administrator is sure that one of the previously mentioned conditions has happened, should he use the `SDRClearLock` command. The `SDRWhoHasLock` command can be used to determine if a class is locked and by whom.

4.4 User Interface to SDR

There are three ways users can get access to the SDR information:

SMIT

SMIT is by far the most commonly used method for interacting with the SDR. Once PSSP is successfully installed, SMIT menus under the “RS/6000 SP

Ex.1010 / Page 126 of 572

System Management" entry will be added to let you manipulate or query SDR database information. The fastpath to these menus is `smit config_mgmt`.

Command line interface
Sometimes it is quicker and more convenient to query SDR information directly. For example, the `SDRGetObjects` command with appropriate arguments can be used to query the SDR database directly.

Through SP commands
The SDR can also be accessed through high-level PSSP commands. One such command is `splstdata`, which (among other things) can be used to query node, frame, and boot/install server information from the SDR.

4.4.1 Shadow Files

As we have observed, the SDR classes are stored in subdirectories under directory /spdata/sys1/sdr. The system-wide classes are stored in subdirectory system/classes and the partition-sensitive classes are stored in subdirectory partitions/<syspar_ip_addr>/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.

4.4.2 Manipulating SDR Data

Before making any changes to the SDR, it should be backed up with the `SDRArchive` command. This enables you to recover from any catastrophic problems such as a power loss. The `SDRRestore` command can be used to restore the SDR to a known working state.

Attention

You must be careful when making changes to the SDR, either directly through the SDR commands or through the SMIT dialogs. If not, your actions may result in a corrupted SDR with unpredictable consequences.

Making changes to an Object

Under normal system operation, you should not need to directly use the commands that manipulate the SDR. The commands are normally used by PSSP to manipulate the SDR data.

When using the SDR commands, the syntax outlined in *IBM PSSP for AIX Command and Technical Reference Vol 2*, SA22-7351 must be used.

When making changes to attributes of existing objects in a class, you can use the `SDRChangeAttrValues` command. With this command, you query the attribute that needs changes and assign it a new value. This command (and others) uses a specify syntax to confirm the value of an attribute before it is changed. The syntax of this command is:

```
SDRChangeAttrValues class_name [ attr==value ... ] attr=value ...
```

The double equal signs (==) signify comparison, and if the value of the attribute in the SDR is not the same as that specified in the command, then the change will not be made.

Adding AIX Files to the SDR

Another characteristic of the SDR is its ability to store and distribute other files. Commands such as `SDRCreateFile, SDRCreateSystemFile, SDRRetreiveFile, SDRReplaceFile`, and `SDRDeleteFile` are available to manipulate these files stored in the SDR.

Some of the subsystems that use this SDR facility are:

- The Switch, to distribute the topology files
- Topology services, to distribute configuration files
- General Parallel File System, to distribute configuration files

Since these commands require the user to be *root*, the SDR offers some degree of security.

For example, let us suppose that some sensitive configuration files are stored in /etc/test_file.conf on the control workstation, and you would like the nodes to securely update this file on the nodes. You can store this file in the SDR using:

Ex.1010 / Page 128 of 572

```
[sp4en0:/]# SDRCreateSystemFile /etc/test_file.conf TestFile
```

This command creates an SDR file called TestFile that can be retrieved from any system partition. (The SDRCreateFile command creates a partitioned SDR file from the AIX file, that is, it can only be retrieved from a node that is a member of the partition it was created in.) To retrieve this file on any node in the SP complex, you would enter:

```
[sp4n06:/]# SDRRetrieve TestFile /etc/test_file.conf
```

4.4.3 Useful SDR Commands

We list here some useful SDR commands for quick reference. For details of the syntax of all SDR commands, refer to *IBM PSSP for AIX Command and Technical Reference Vol 2*, SA22-7351.

SDR_test
This command verifies that the installation and configuration of the SDR completed successfully. The test clears out a class name, creates the class, performs some tasks against the class, and then removes it.

SDRGetObjects
This command lists the contents of attributes in the specified object. Some useful objects to remember include:

- host_responds
- switch_responds
- Syspar
- Syspar_map
- Node
- SP
- Adapter

You can filter the output by specifying the required attributes as arguments. For example, to query the SDR Adapter class for the node number and network address of all switch adapters, enter:

```
[sp4en0:/]# SDRGetObjects Adapter adapter_type==css0 node_number netaddr
node_number  netaddr
          1 192.168.14.1
          5 192.168.14.5
          6 192.168.14.6
          7 192.168.14.7
          8 192.168.14.8
          9 192.168.14.9
         10 192.168.14.10
         11 192.168.14.11
```

```
13 192.168.14.13
15 192.168.14.15
```

Remember, the double equal signs signify comparison, and because we have specified additional attributes as arguments, only those attributes will be listed in the output.

SDRChangeAttrValues
This command changes attribute values of one or more objects. Remember, this is a “last resort” action and should never be performed if there is another method, through command-level actions, to perform the same function. For example, the SP Switch will not start if an Oncoming Primary node is fenced, but we cannot use the `Eunfence` command to unfence the node, since the switch is not started. So, in this case, we have to change this attribute in the SDR using the SDRChangeAttrValues command. The command with the relevant arguments is:

```
[sp4en0:/]# SDRChangeAttrValues switch_responds node_number==7 isolated=0
[sp4en0:/]#
```

SDRListClasses
This command outputs all of the class names (system and partitioned) currently defined in the SDR to standard output. It has no flags or arguments.

SDRWhoHasLock
This command returns the transaction ID of a lock on a specified class. The form of the transaction ID is host_name:pid:session, where host_name is the long name of the machine running the process with the lock, pid is the process ID of the process that owns the lock, and session is the number of the client’s session with the SDR.

SDRClearLock
This command unlocks an SDR class.

Attention

Use this command carefully and only when a process that obtained a lock ended abnormally without releasing the lock. Using it at any other time could corrupt the SDR database.

4.5 The SDR Log File

Serious SDR problems are reported in a log file located on the control workstation. This SDR log file has the following filename format:

Ex.1010 / Page 130 of 572

```
/var/adm/SPlogs/sdr/sdrlog.<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. Based on the number of partitions, you may have multiple sdrd daemons running. Often, the log file contains little more than a message showing a return code of 110, indicating that a client exited without closing its session with the SDR. These messages have no adverse effect on the SDR daemon and can be ignored.

A new log file is created every time the SDR daemon is started. Once the contents are checked, the files can be either archived or discarded, except for the file that corresponds to the daemon that is currently running. To find the current log file, find the process ID of the SDR daemon using `lssrc -g sdr`. The current log file has that process ID as its extension.

The logs contain the data and time the process started along with any problems the daemon may have run into.

4.6 Backup and Restore of SDR

To insure against the loss of SDR data, it is recommended to back up the SDR regularly using the `SDRArchive` command. This commands takes a snapshot of the SDR contents and saves it in a tar file. It is a good idea to use this facility before making any system configuration changes.

The backup filename format is:

```
backup.JULIANdate.HHMM.append_string
```

where `JULIANdate.HHMM` is a number or string uniquely identifying the date and time of the archive and `append_string` is an optional string argument passed to the command.

If you need to restore the SDR to a previous state, you must use the PSSP command `SDRRestore`.

This command will remove the contents of the SDR and retrieve the archived contents of the backup file. The backup file must be in the /spdata/sys1/sdr/archives directory. Any new SDR daemons that represent partitions in the restored SDR are then started and any SDR daemons (from old partitions) that are not in the new SDR are stopped.

Attention

Each new PSSP release may introduce new SDR classes and attributes. Use caution when using `SDRArchive` and `SDRRestore` in order to avoid overwriting new SDR classes and attributes.

We suggest that after migration you do not execute `SDRRestore` from a back level system since it will overwrite any new SDR classes and attributes.

Ex.1010 / Page 132 of 572

Chapter 5. Hardware Control Subsystem

One main task of the control workstation for the RS/6000 SP system is to provide a centralized hardware control mechanism. In a large SP environment, the control workstation makes it easier to monitor and control the system. The administrator can power a node on or off, update supervisor microcodes, open a console for a node and so on, without having to be physically near the node. This chapter describes how the hardware control subsystem works and where it is applied.

5.1 Components That Can Be Monitored And Controlled

An RS/6000 SP system consists of the following hardware components:

- Frame
- Node
- Switch
- Control workstation

The control workstation, being the center of hardware control, is able to monitor and control the frames, nodes and switches.

Frame
Hardware components in a frame that can be *monitored* include:

- Power LEDs
- Environment LEDs
- Switch
- Temperature
- Voltage
- State of the power supply

Hardware components in a frame that can be *controlled* include:

- Power on/off

Node
Hardware components in a node that can be *monitored* include:

- Three-digit or LCD display
- Power LEDs
- Environment LEDs
- Temperature
- Voltage
- Fan

© Copyright IBM Corp. 1999

Attention

Not all node types will display all of this information.

Hardware components in a node that can be *controlled* include:

- Power on/off
- Key Mode Switch
- Reset

Switch

Hardware components in a switch that can be *monitored* include:

- Temperature
- Voltage
- Fan
- Power LEDs
- Environment LEDs
- MUX

Hardware components in a switch that can be *controlled* include:

- Power on/off
- Change multiplexor clock setting

Attention

For SP Switch, `Eclock` must be used to change the multiplexor clock setting. This command runs only on the control workstation and must not be run unless absolutely necessary, since it brings down the switch network.

5.2 How It Works

Now that we know what hardware components can be monitored and controlled, we can describe how the hardware control subsystem works.

A daemon, called *hardmon,* runs on the Monitor and Control Node (MACN), an alias of the control workstation where hardmon is concerned, which has the RS-232 serial cables connected to the frames. It constantly polls the frames at a default rate of 5 seconds.

Attention

Do not change the value of this default poll rate as it can cause unpredictable results in some client commands.

This daemon is invoked during the boot process by the /etc/inittab file and is controlled by the System Resource Controller (SRC) subsystem. Unless specifically called by the `stopsrc` command, the SRC subsystem will restart this daemon on termination.

The hardmon daemon first obtains configuration information from the System Data Repository (SDR). The SP_ports object class provides the port number this daemon uses to accept TCP/IP connections from client commands. The Frame object class provides the frame number, tty port, MACN name (also backup_MACN name in a HACWS environment), hardware_protocol, s1_tty and so on. Once this information is acquired, the hardmon daemon opens up the tty device for each frame and begins polling for state information over the RS-232 serial line, using SLIP protocol. The first thing this daemon checks for is the Frame ID. Any discrepancy will disable any control over the frame. However, monitoring of the frame is still allowed. To correct Frame ID mismatch, the `hmcmds` command is used.

In order to determine the boundary conditions for state variables like temperature, voltage and amperage, the hardmon daemon reads the /spdata/sys1/spmon/hmthresholds file. This file provides for software thresholding in conjunction with hardware thresholding.

While the hardmon daemon provides the underlying layer for gathering hardware state information, client commands `spmon`, `hmmon`, `hmcmds`, `s1term`, and `nodecond` interact with it to monitor and control the hardware.

The hardmon daemon reports variables representing the state of the frame, node and switch hardware. These state variables can be queried with the `spmon` and `hmmon` commands. For a full list of available state variables, use the following command:

```
# hmmon -V
```

The *splogd* daemon runs alongside the hardmon daemon. It provides error logging capability for all errors detected by hardmon. To define what logging is to be done and what user exits should be called, syslogd uses the /spdata/sys1/spmon/hwevents file.

Figure 50 shows the relationship between all the components of the hardware monitor.

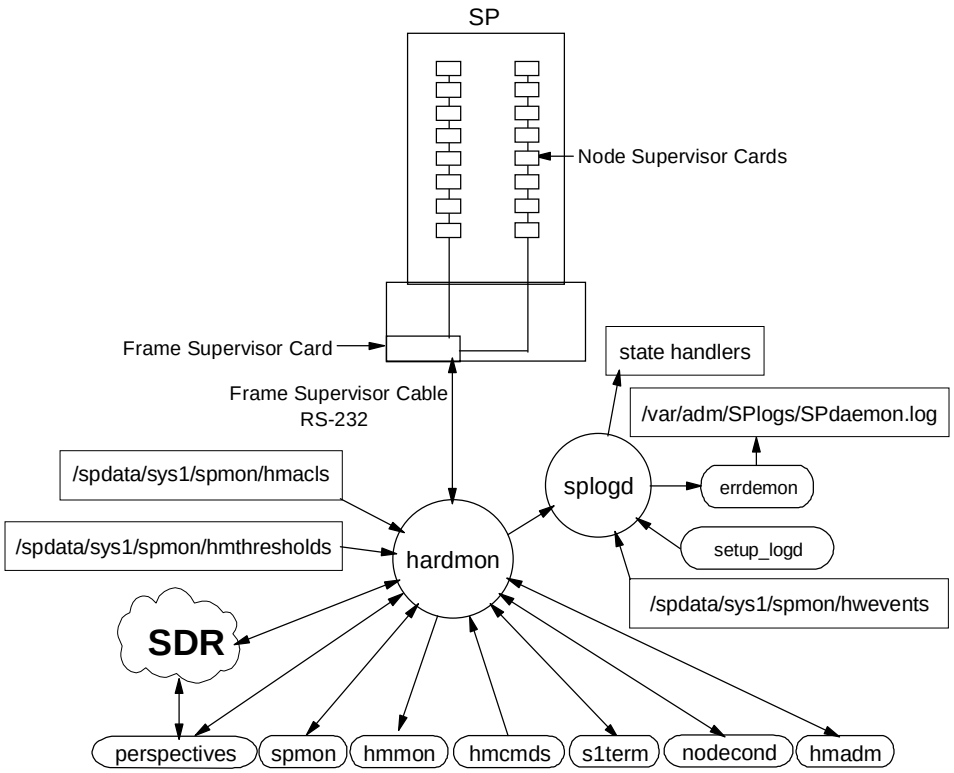


Figure 50. Hardware Monitor Components and Relationship

5.2.1 SP-Attached Servers

Hardware monitoring for the SP-attached servers, like the S70, S7A and Netfinity servers, is slightly different from that of the normal frames and nodes. These servers do not have supervisor cards. A new mechanism was designed in PSSP 3.1 to help monitor and control them in an SP environment.

When the hardmon daemon starts, one of the things it checks for is the hardware_protocol field in the SDR's Frame class. If this field is SP, it treats the frame as a regular SP frame and begins its communications to the supervisor cards. If the field indicates SAMI, it recognizes the attached server as an external S70/S7A and begins communicating with the S70/S7A's Service Processor. If the field specifies SLIM, the daemon recognizes the

Ex.1010 / Page 136 of 572

attached server as a Netfinity server and communicates with the Service Processor on that server. To display the Frame class, the `SDRGetObjects` command is used as follows:

```
# SDRGetObjects Frame
```

However, the output format may be somewhat distorted for reading.

Frame information is stored in the /spdata/sys1/sdr/system/classes/Frame file. The definition for each field in the Frame class is located in the /spdata/sys1/sdr/defs/Frame file. The contents of these files are as follows:

- Contents of /spdata/sys1/sdr/system/classes/Frame

  ```
  1=1 2=/dev/tty0 3=switch 4=sp3en0 6=16 7=1 8=0 9=0 10=SP
  1=2 2=/dev/tty1 4=sp3en0 6=1 10=SAMI 11=/dev/tty2
  1=3 2=/dev/tty3 3=non-node 4=sp3en0 6=1 10=SLIM
  ```

- Contents of /spdata/sys1/sdr/defs/Frame

  ```
  gI1=frame_number S2=tty S3=frame_type S4=MACN S5=backup_MACN I6=slots
  I7=frame_in_config I8=snn_index I9=switch_config S10=hardware_protocol
  S11=s1_tty
  ```

Although you can access these files (as well as other SDR files) directly, do not modify them manually. All changes must be made through the proper installation and customization procedures.

Attention

Support for SP-attached Netfinity Servers does not come with the default PSSP codes. It can be purchased as a separate product and is available only on a Programming Request for Price Quotation (PRPQ) from IBM. The PRPQ number is P88717.

The hardmon daemon recognizes the existence of SP-attached servers, and starts the relevant daemons to support them. The two special daemons which support SP-attached servers are: *s70d* daemon and *nfd* daemon.

For the S70/S7A servers, the s70d daemon is started by the hardmon daemon. For each server attached, one daemon is started. Figure 51 on page 114 shows the hardware monitor components and relationships for the S70/S7A attachment.

The s70d daemon provides the interface for hardmon to communicate with the S70/S7A hardware. Since S70/S7A's SAMI Manufacturing Interface uses SAMI protocol, this daemon has the function of translating hardmon

commands to the SAMI language. Likewise, information sent back by the service processor gets translated into supervisor data packets for hardmon to digest. The link through which the SAMI information flows is the SAMI RS-232 serial line. Through this line, the daemon polls the servers for hardware states and sends them to hardmon. Another RS-232 line is also required for the `s1term` command to use. This line provides the channel for the `s1term` command to open up a virtual console for the server.

The s70d daemon consists of two interfaces: the frame supervisor interface and the node supervisor interface. The frame supervisor interface is responsible for keeping the state data in the frame's packet current and formatting the frame packet for return to the hardmon daemon. The node supervisor interface is responsible for keeping the state data in the node's packet current and translating the commands received from the frame supervisor interface into SAMI protocol before sending them to the service processor on the servers.

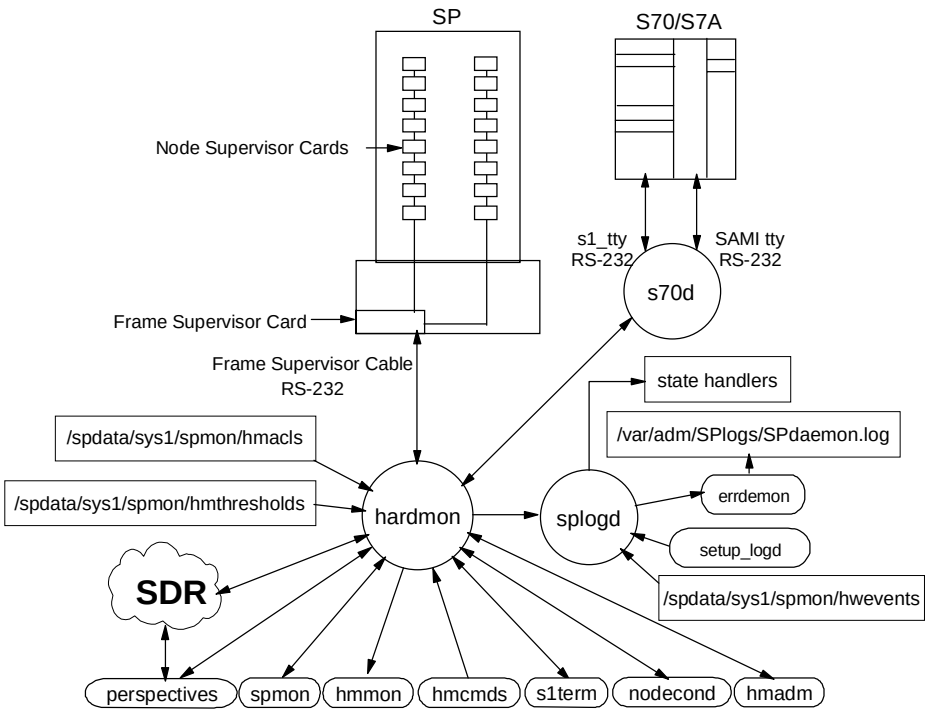


Figure 51. Hardware Monitoring For SP-Attached S70/S7A Servers

For the Netfinity servers, the nfd daemon is started by the hardmon daemon. Like the s70d daemon, one nfd daemon is started for each server attached. Figure 52 shows how the nfd daemon integrates with hardmon in the hardware monitoring flow.

The nfd daemon provides the interface for hardmon to communicate with the Netfinity server's hardware. It emulates the frame supervisor card as well as the node supervisor card. This daemon translates hardmon commands to SLIM language for the service processor on the server. Information sent back by the service processor gets translated into supervisor data packets for hardmon to digest. An RS-232 serial line provides the channel for hardware data flow. Through this line, the daemon polls the servers for hardware states and sends them to hardmon. Unlike the S70/S7A attachment, where another serial line allows the opening of a virtual console, the Netfinity server runs Windows NT or OS/2 and thus does not support virtual console.

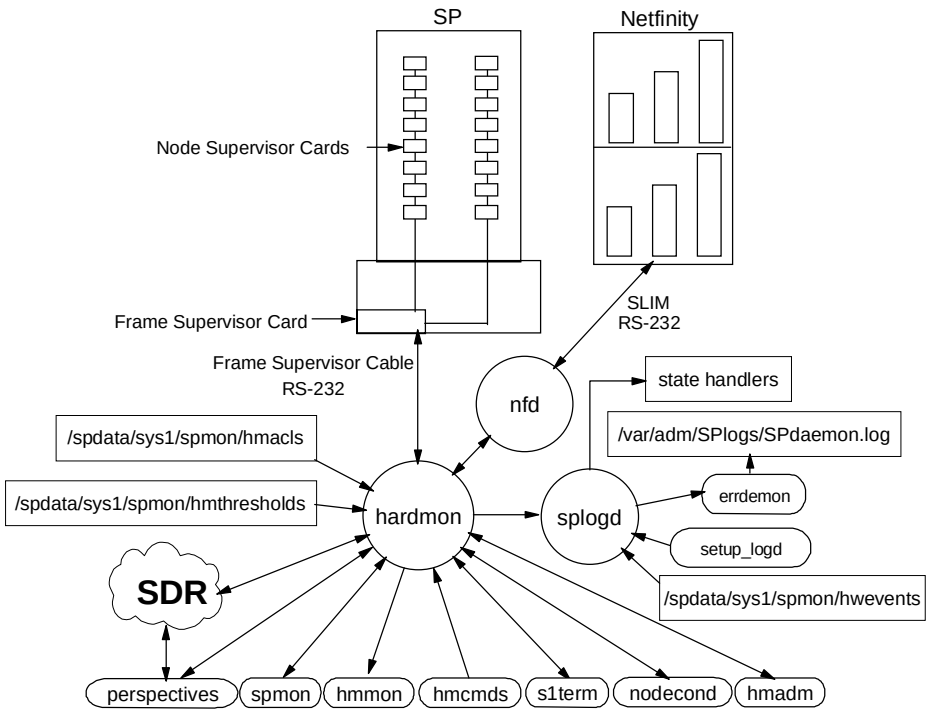


Figure 52. Hardware Monitoring For SP-Attached Netfinity Servers

5.2.2 Security Aspect

For full control of hardmon services, the Kerberos root principal is required. However, a normal user can have access to these services with a certain amount of control. To do so, the user must be registered as a principal in the SP Kerberos V4 authentication server. With this in place, the /spdata/sys1/spmon/hmacls file has to be edited to include this principal user, specifying what authority level the user has to access the hardmon services. The hardmon daemon will read the hmacls file when the `hmadm setacls` command is issued or when it is restarted. For more information on the security aspect of hardmon, refer to Chapter 7, “SP Security” on page 139.

5.3 User Interfaces

Two methods are available to interface with the hardmon daemon, to access information from and control the hardware components. They are the command line interface and the graphical user interface SP Perspectives. For instructions on how to use hardmon client commands, refer to *PSSP: Command and Technical Reference Volume 1 and 2*, SA22-7351. The following sections outline some common usages of these interfaces.

5.3.1 Command Line

This section discusses commands used to interface with the hardmon daemon to obtain the various hardware states and at the same time provide control. Command line execution is seldom used. However, administrators can incorporate such commands into scripts to help them monitor the states of the hardware and notify them upon detecting any abnormal condition. Unless necessary, refrain from executing such commands to control hardware. Wrong usage can cause undesirable results. Commands used to monitor hardware states are generally safe to use but will require a bit of in-depth knowledge in order to interpret the output.

Attention

Examples listed here are based on the current setup in our laboratory. Variables used may not be applicable in your hardware environment. Check against the listed variables in *PSSP: Administration Guide*, SA22-7348 for your hardware configuration.

hmmon

The `hmmon` command monitors the state of the SP hardware in one or more SP frames. For a list of state variables which this command reports, refer to

Appendix E "System Monitor Variables" in *PSSP Administration Guide*, SA22-7348. This same list can be generated by the following command:

```
# hmmon -V
```

To list all the hardware states at a particular point in time, the following command is used:

```
# hmmon -G -Q -s
```

This command provides information about the frames, switches and nodes. Without the -G flag, only nodes information in the current partition will be listed. To monitor for any changes in states, leave the -Q flag out. This will continuously check for state changes. Any condition that changed will be reflected on the screen where the command was executed.

A common usage for this command is to detect whether the LED on a node is flashing. Nodes that crash with flashing 888 LED codes can be detected by this command.

For example, to detect if the LED on Frame 1 Node 5 is flashing, issue the following command:

```
# hmmon -G -Q -v 7segChanged 1:5
```

If the LED is flashing, the value returned will be TRUE. The output will be as follows:

```
frame 001, slot 05:
  7 segment display flashing     TRUE
```

As another example, to detect if the power switch is off for a frame (maybe due to a power trip) on power module A, use the following command:

```
# hmmon -G -Q -v frPowerOff_A 1:0
```

If nothing is wrong with this power module, the value returned is FALSE as shown in the following:

```
frame 001, slot 00:
  AC-DC section A power off     FALSE
```

spmon

The `spmon` command can be used for both monitoring and controlling the system. There is a whole list of tasks this command can perform. Included here are examples showing the common usage of this command.

Using the diagnostics mode of this command, a series of tests is run. It checks if the hardmon process is running and tries to open a connection to the server. Next, it queries and checks the frames, nodes and switches. The following command shows an example of entering this mode:

```
# spmon -G -d
```

The output is as follows:

```
1.  Checking server process
    Process 42910 has accumulated 87 minutes and 40 seconds.
    Check ok

2.  Opening connection to server
    Connection opened
    Check ok

3.  Querying frame(s)
    1 frame(s)
    Check ok

4.  Checking frames

        Controller   Slot 17  Switch   Switch     Power supplies
Frame   Responds     Switch   Power    Clocking   A   B   C   D
-------------------------------------------------------------------
  1       yes         yes      on         0       on  on  on  on

5.  Checking nodes
--------------------------------- Frame 1 -----------------------------------
Frame  Node    Node         Host/Switch  Key     Env   Front Panel   LCD/LED is
Slot   Number  Type  Power   Responds    Switch  Fail    LCD/LED      Flashing
-------------------------------------------------------------------------------
  1       1    high    on   yes  autojn  normal   no   LCDs are blank   no
  5       5    thin    on   yes  yes     normal   no   LEDs are blank   no
  6       6    thin    on   yes  yes     normal   no   LEDs are blank   no
  7       7    thin    on   yes  yes     normal   no   LEDs are blank   no
  8       8    thin    on   yes  yes     normal   no   LEDs are blank   no
  9       9    thin    on   yes  yes     normal   no   LEDs are blank   no
 10      10    thin    on   yes  yes     normal   no   LEDs are blank   no
 11      11    thin    on   yes  yes     normal   no   LEDs are blank   no
 12      12    thin    on   yes  yes     normal   no   LEDs are blank   no
 13      13    thin    on   yes  yes     normal   no   LEDs are blank   no
 14      14    thin    on   yes  yes     normal   no   LEDs are blank   no
 15      15    wide    on   yes  yes     normal   no   LEDs are blank   no
```

The next example shows how you can power off a node. Use the following command with caution, since as you can accidentally power off a wrong node if you specify the wrong node number. This command also does not issue a shutdown sequence prior to powering it off.

```
# spmon -p off node5
```

This command powers off node 5. No message output will be given.

Ex.1010 / Page 142 of 572

hmcmds

The hmcmds command controls the state of the SP hardware via the Virtual Front Operator Panel (VFOP). Like the spmon command, it can be quite risky issuing this command from the command line. Normally, it is not necessary to run this command. All controls can be accessed via either the SMIT menu or from SP Perspectives.

An example of using this command to turn the key mode of all nodes in Frame 1 to Service is as follow:

```
# hmcmds service 1:1-16
```

s1term

The s1term command enables an administrator to open up a virtual console on a node. The default read-only option allows an administrator to monitor the processes going on in a node. The read/write option allows the administrator to enter commands to administer the node. The following command opens a console with read/write option on Frame 1 Node 15.

```
# s1term -w 1 15
```

Attention

There can only be one read/write console at a time for a particular node. If there is already a read/write console opened, the nodecond command will fail, since it will attempt to open a read/write console.

nodecond

The nodecond command is used for conditioning a node, where it is used to obtain the Ethernet hardware address or initiate a network boot. This command is discussed in further detail in Chapter 10, “Installation and Configuration” on page 257.

5.3.2 SP Perspectives

The graphical version of spmon is no longer available in PSSP 3.1. Instead, use the sphardware command monitor and control the SP hardware from SP Perspectives. Alternatively, the perspectives command can be invoked, which will start off the SP Perspectives Launch Pad. An icon indicating Hardware Perspective will lead you to the same screen; this is illustrated in Figure 53 on page 120. For information relating to SP Perspectives, refer to Chapter 11, “System Monitoring” on page 293.

Ex.1010 / Page 143 of 572



Figure 53. SP Perspectives Launch Pad

The Hardware Perspective window is shown in Figure 54 on page 121. By default, two panes are shown: CWS, System and Syspars, and Nodes. Additional panes can be added to monitor various conditions; we describe how to do this in page 125. The items in the panes, by default, do not monitor any condition. You can customize those items to monitor the conditions you consider important.

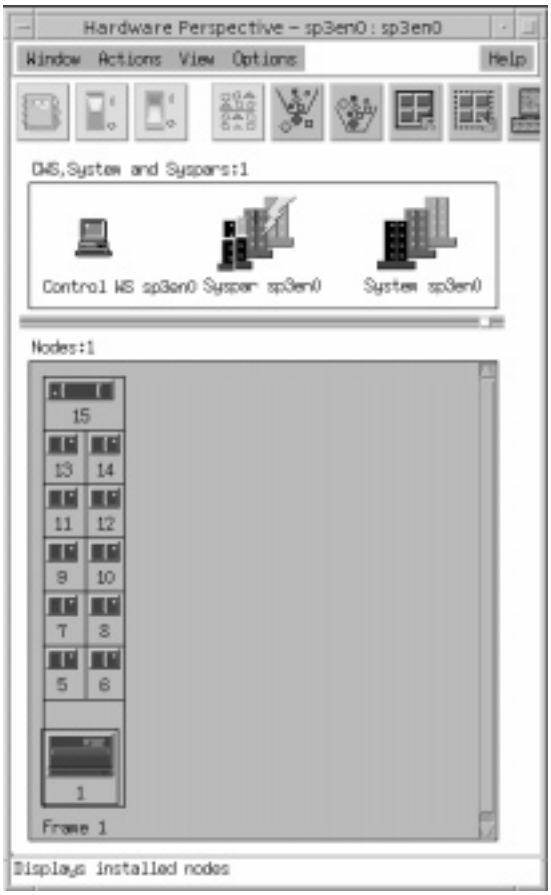


Figure 54. Hardware Perspective

Figure 55 on page 122 shows how S70 servers appear in Perspectives.

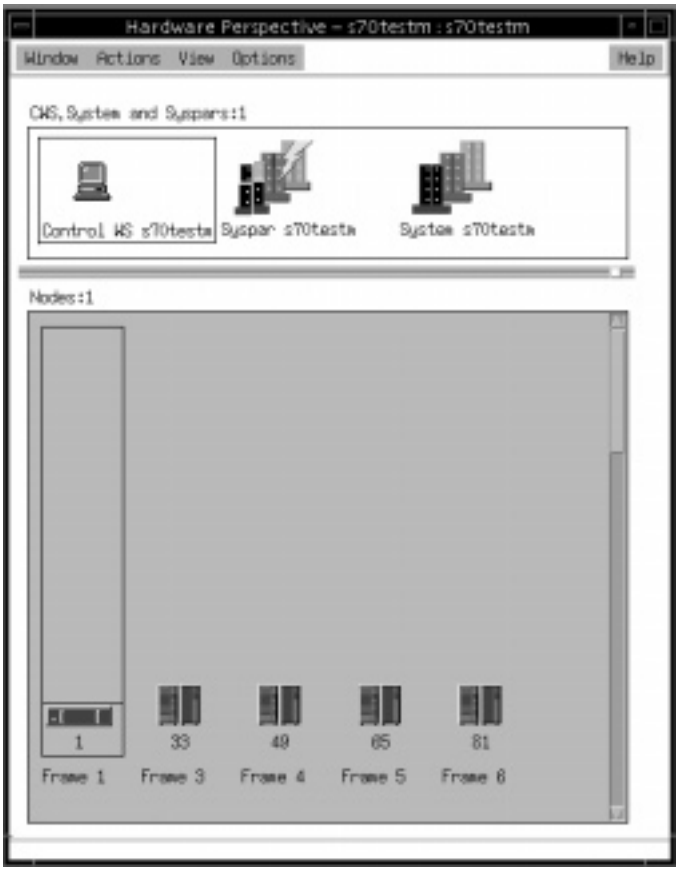


Figure 55. S70 Nodes In Perspectives

Figure 56 on page 123 shows how Netfinity servers appear in Perspectives.

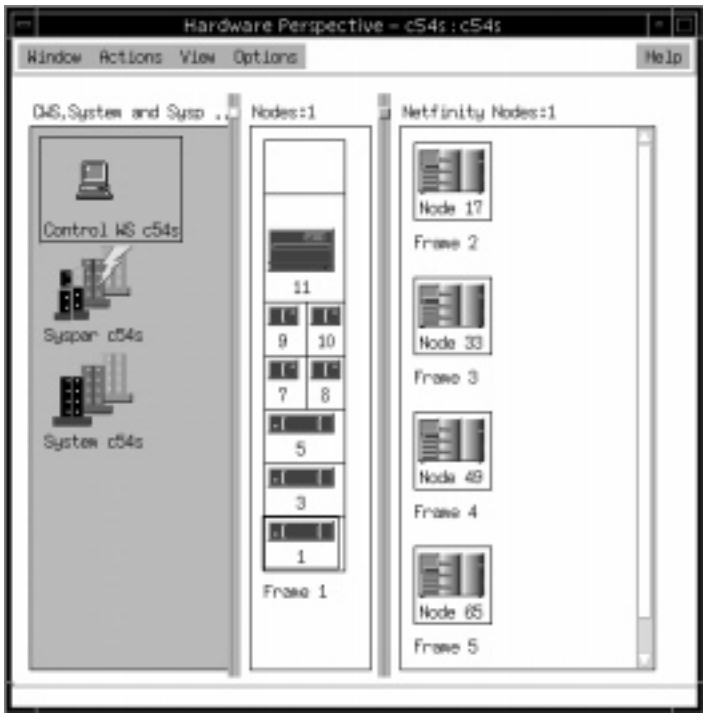


Figure 56. Netfinity Nodes In Perspectives

By selecting a particular node or group of nodes, you can power on/off, network boot, open tty and so on. Figure 57 on page 124 demonstrates how a group of nodes can be powered off by selecting the **Power Off** option from the **Action** pull-down menu.

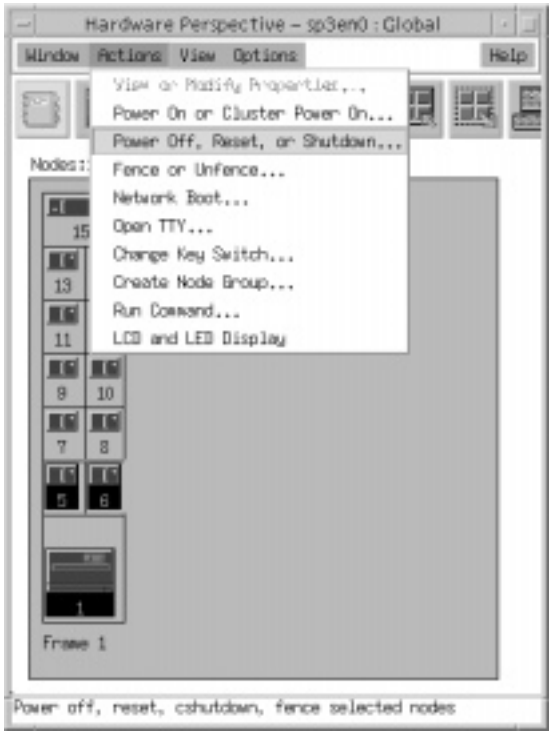


Figure 57. Powering Off Selected Nodes

After selecting the **Power Off** option, another window appears, indicating the nodes selected and prompting the user to select the required mode before continuing with the task. This is shown in Figure 58 on page 125. To shut down and power off a node, select the **Power Off** and **Shutdown** options. Clicking **Apply** will cause the node to be shut down and then powered off.

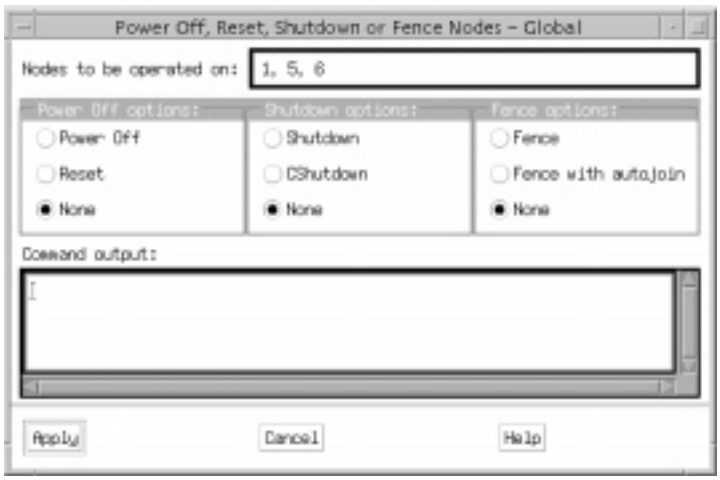


Figure 58. Power Off Mode Selection/Confirmation Window

The next example shows how you can add new panes on a new window to monitor the frame, switch and host_responds for the nodes. To add a new pane, click **View** and select **Add Pane**. A window will appear as shown in Figure 59. Select **Pane type**. In the example shown, the Frames and Switches pane was selected. You have an option to add this new pane to a new window or keep it in the existing window. Here, we added it to a new window.



Figure 59. Adding A New Pane

After selecting the **Apply** button to confirm, a new window with the new pane appears. Select the **Add Pane** option on this new window to add a new Nodes pane to this current window. The resulting window is shown in Figure 60 on page 126.

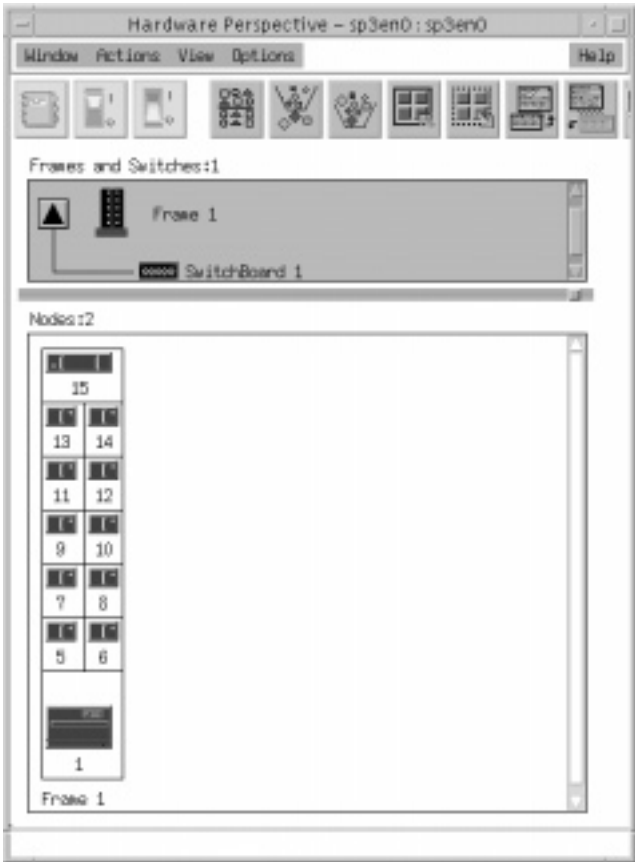


Figure 60. New Window With Two New Panes

To start monitoring the host_responds for all the nodes, select a node in the Nodes pane. Choose **Set Monitoring** from the **View** pull-down menu. A window will appear as shown in Figure 61 on page 127.



Figure 61. Set Monitoring Window

Select the hostResponds condition and click **Apply**. The nodes are now set for monitoring the host_responds. The green color indicates the host_responds is up. A red cross sign means the host_responds is down. You can choose to select more than one condition to monitor. To do so, click the conditions while holding down the Control key on the keyboard. Figure 62 on page 128 shows the Nodes pane set to monitor the host_responds of all the nodes. Note that Node 6 had been powered off.

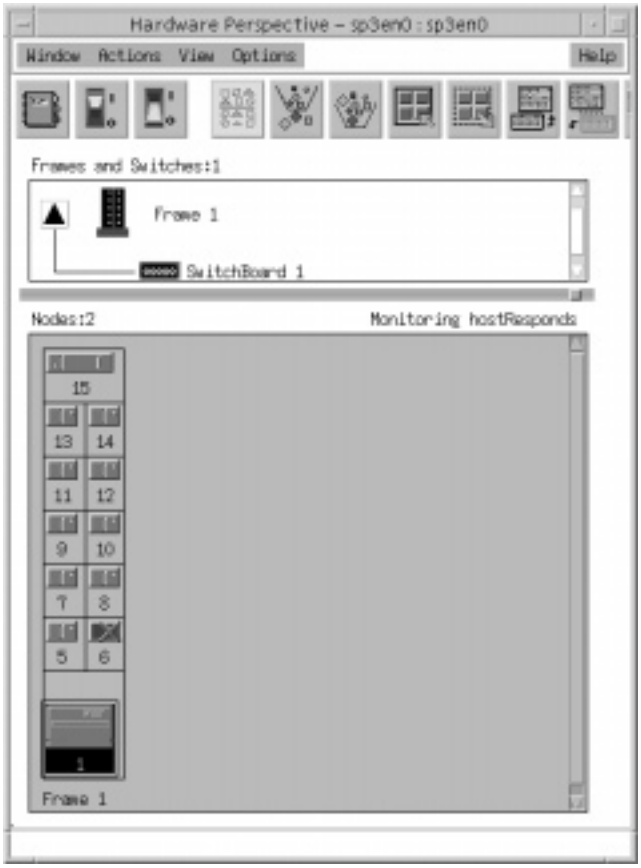


Figure 62. Monitoring Host Responds On The Nodes Pane

You can view or modify the properties of any nodes, frames or switches, by double clicking their respective icons. To illustrate this example, we double click on Frame 1. A window showing the properties appears as shown in Figure 63 on page 129.

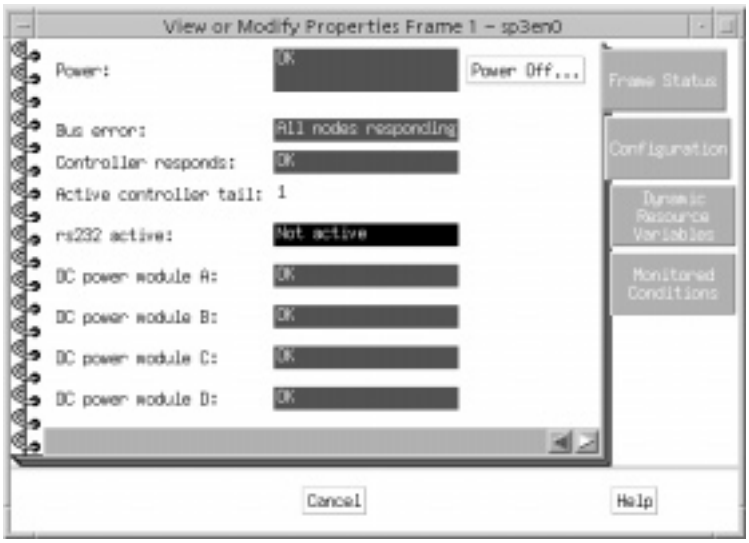


Figure 63. Frame Properties Window

To power off the frame or switch, set your system partition to Global. Select **Change System Partition** from the **View** pull-down menu. A window will appear as shown in Figure 64. Select Global and click **Ok**.



Figure 64. Changing System Partition In Perspectives

To power off Frame 1, click on Frame 1 in the **Frames and Switches** pane and select **Power Off** from the **Actions** pull-down menu. A warning window will appear, as shown in Figure 65 on page 130. Clicking **Enter** will power off the frame.

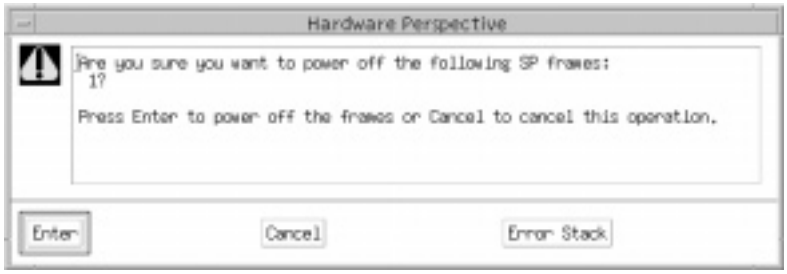


Figure 65. Powering Off A Frame

Hardware Perspective makes it easier to manage the RS/6000 SP system. It reduces the need to use the command line to control and monitor the hardware and produces excellent graphics to improve comprehension. This chapter shows only some of the tasks Hardware Perspective can do. If you explore it more, you will find many ways it can help you with your daily administrative chores. For detailed information about using SP Perspectives refer to *SP Perspectives: A New View of Your SP System*, SG24-5180.

Chapter 6. Time Service

In a single computing system, there is one clock which provides the time of day to all operating system services and applications. This time might not be completely accurate if it is not synchronized with an external source like an atomic clock, but it is always consistent among applications.

In a distributed computing system like the SP, each node has its own clock. Even if they are all set to a consistent time at some point, they will drift away from each other since the clocking hardware is not perfectly stable. Consequently, the nodes in such a distributed system have a different notion of time.

Inconsistent time is a critical problem for all distributed applications which rely on the order of events. An important example is Kerberos, the SP authentication system: it is a distributed client/server application and encodes timestamps in its service tickets. Another example is the RS/6000 Cluster Technology, in particular the Topology Services, which send heartbeat messages containing timestamps from node to node. On the application level, tools like the make utility check the modification time of files and may not work correctly when called on different nodes which have inconsistent time settings.

In order to ensure a consistent time across nodes, a time management system is needed. PSSP by default uses Network Time Protocol (NTP), a recommended Internet standard for managing time in a networked environment. NTP and its exploitation by PSSP is described in 6.1, “Network Time Protocol (NTP)” on page 131.

Another industry standard is Distributed Time Service (DTS), which is a component of Distributed Computing Environment (DCE). Customers who want to operate their SP within a DCE cell should use DTS as the SP’s time service. DTS is discussed in 6.2, “Distributed Time Service (DTS)” on page 135.

6.1 Network Time Protocol (NTP)

Network Time Protocol (NTP) version 3 is a widely used Internet protocol for time management. It is described in detail in RFC 1305. By default, PSSP uses NTP version 3 as the time management system for the SP. During PSSP installation, NTP can be automatically configured for the control workstation and nodes. Due to the hierarchical structure of NTP, it is easy to integrate an SP that uses NTP into a site’s existing NTP time management regime.

© Copyright IBM Corp. 1999

6.1.1 NTP Architecture

NTP uses a hierarchical, master-slave configuration as outlined in Figure 66. At the top of the hierarchy are the primary servers, which should be directly synchronized to a primary reference source, usually a radio clock. Below the primaries are secondary servers, which derive synchronization from a primary server over the network. There may be many levels (hops) in the hierarchy of secondary servers. Each level is referred to as a *stratum*, with the primaries at stratum 0, the first level of secondaries at stratum 1, and so on. The stratum of a server specifies how many hops are required to the primary clock source. The higher a server's stratum number, the more hops are required up the hierarchy, and the less accurate that server is.

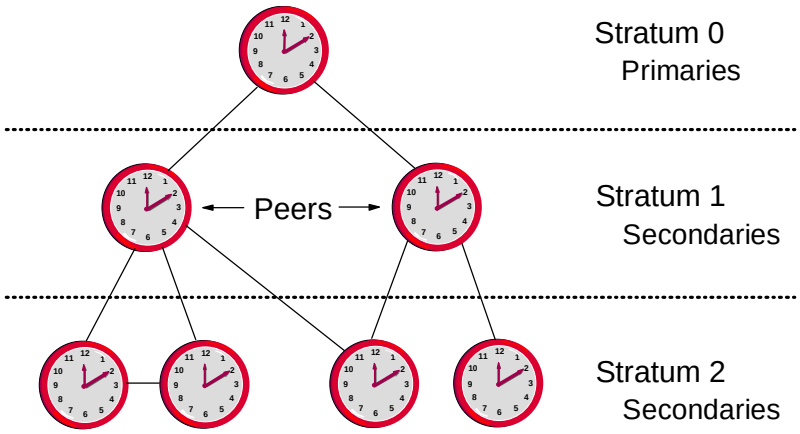


Figure 66. NTP Hierarchy

Secondary servers are both clients of lower-stratum servers and servers to higher-stratum servers. Servers in the same stratum are called *peers*, and may give or request time from each other. NTP uses UDP/IP messages, thus any IP-capable network can be used to form an NTP hierarchy. A host configured to have multiple time servers employs a selection algorithm to determine the most accurate and reliable server with which to synchronize. This also includes mechanisms to identify and isolate malfunctioning servers.

NTP computes the clock offset, or time difference, of the local client relative to the selected reference clock. The phase and frequency of the local clock is then adjusted, either by a step-change or a gradual phase adjustment, to reduce the offset to zero. The accuracy of synchronization depends on a host's position in the stratum and the duration of synchronization: whereas a few measurements are sufficient to establish local time to within a second, long periods of time and multiple comparisons are required to resolve the

oscillator skew, or frequency difference, and maintain local time to within a millisecond. The more frequently a host compares its clock to its time server(s), the better the accuracy.

6.1.2 Configuring and Using NTP on the SP

In PSSP, time service is set up as part of the site environment. The `spsitenv` command has three flags that control the NTP configuration of the SP: ntp_config, ntp_server and ntp_version (the latter defaults to 3 and need not be changed). The ntp_config flag offers four choices which govern the NTP configuration of the control workstation, boot/install server nodes, and regular nodes:

consensus With this option, you run NTP on the SP nodes and control workstation. This configuration uses the control workstation as the primary time server, or time master. Any boot/install servers in the SP become peer secondary servers. Regular nodes can request time from either boot/install server nodes or the control workstation.

Note that unless the control workstation is connected to an external reference clock, this setup does not guarantee that time on the SP is accurate. It only ensures that the nodes' time is consistent with the control workstation.

timemaster With this option, you use the site's existing NTP domain to synchronize the SP nodes and control workstation. The control workstation and boot/install server nodes are peers. They request time from each other and the site's NTP time server(s). Regular nodes can request time from the control workstation and boot/install server nodes, and depending on their connectivity, from the time server(s) directly.

internet With this option, you use an NTP server from the Internet to synchronize the SP nodes and control workstation. The control workstation is assumed to be connected to the Internet time server(s) and is configured alone in a stratum. Boot/install servers become peers at a higher number stratum (one level down the hierarchy from the control workstation). Regular nodes request time from the control workstation and boot/install server nodes.

none With this option, you do not use NTP. Some other means to ensure consistent time across the system should be used.

The default configuration is consensus. When configuring NTP for timemaster or internet, the ntp_server flag specifies the IP name(s) of the time server(s) to use. If the internet option is chosen, we suggest specifying multiple time servers on separate networks to ensure the SP always has access to a time source.

To actually configure NTP, the `spsitenv` command calls `services_config` on the control workstation, boot/install servers and regular nodes, which in turn calls `ntp_config`. The ntp_config script creates (or updates) the /etc/ntp.conf file which describes the NTP server configuration. Lines beginning with the server keyword define lower-stratum servers from which the current machine can request time, and lines beginning with the peer keyword define servers in the same stratum with which this machine can synchronize time.

The following example shows the respective entries in /etc/ntp.conf for an SP with two boot/install servers, which is set up in consensus mode:

```
# On the CWS (192.168.3.130):
#
server 127.127.1.10 version 3

# On the BIS nodes (192.168.3.11 and 192.168.3.17):
#
server 192.168.3.130
peer 192.168.3.11
peer 192.168.3.17

# On regular nodes:
#
server 192.168.3.130
server 192.168.3.11
server 192.168.3.17
```

Note that `ntp_config` converts all IP names to IP addresses. In this example, IP address 127.127.1.10 in the control workstation configuration is used to indicate local time service to NTP and makes the control workstation a stratum-10 time server. This stratum value allows you to hook the SP into a lower-stratum time server at a later time.

The script `ntp_config` also starts the NTP daemon, `xntpd`, by running `/etc/rc.ntp`. Be aware that although xntpd is known to the AIX System Resource Controller (SRC), `rc.ntp` starts it directly and so `lssrc -s xntpd` will show `xntpd` to be inoperative even if the daemon is running. NTP status information can be obtained via the `xntpdc` command, for example `xntpdc -c help` or `xntpdc -c sysinfo`.

Ex.1010 / Page 158 of 572

After each reboot, NTP will be restarted since `/etc/rc.sp` calls `services_config`, which calls `ntp_config`.

6.2 Distributed Time Service (DTS)

Distributed Time Service (DTS) is a core component of the Distributed Computing Environment (DCE) from the Open Software Foundation (OSF). On RS/6000 systems, AIX DCE is available from IBM as a Licensed Program Product. The SP might be integrated into a site's DCE cell to make use of DCE's security and user management features, its Distributed File System DFS, or other DCE applications. In this case, the control workstation and nodes are configured as DCE clients and can utilize DTS to establish a consistent time across the SP (and the rest of the DCE cell). 6.2.1, "Architecture of DCE DTS" on page 135 gives a brief overview of the DTS architecture and its relation to NTP, and 6.2.2, "Configuring and Using DTS on the SP" on page 137 describes how to set up DTS as the SP's time service.

6.2.1 Architecture of DCE DTS

The objective of Distributed Time Service (DTS) is very similar to NTP, which is described in 6.1, "Network Time Protocol (NTP)" on page 131. DTS allows nodes which are distributed on a network to have a consistent notion of time, by providing a way to periodically synchronize their clocks. It also ensures that this synchronized time stays reasonably close to the correct time. However, DTS differs from NTP in three fundamental points:

1. DTS is more secure than NTP, since it uses the DCE Remote Procedure Call (RPC) for its communication. DCE RPC is integrated with the DCE security system which includes Kerberos 5 as a third-party authentication system. Although the NTP protocol provides for an optional security mechanism, this is rarely used.
2. DTS can be managed much easier than NTP. DTS clients, or *clerks*, find their DTS servers via the DCE Cell Directory Service (CDS), and do not have local configuration information pointing to the DTS servers. With NTP, it can be very difficult to change the server hierarchy since there is no easy way to find out which clients reference a specific NTP server.
3. In DTS, distributed time is expressed as an interval rather than as a single point. This accounts for the inaccuracy which is inherent in distributed systems. With DTS, the correct time should always be included in this interval, and the width of the interval provides an estimate of the accuracy.

Like NTP, DTS is based on a client/server model. The DTS time clerk is the client component. The number of time servers in a DCE cell is configurable; three per Local Area Network (LAN) is a typical number. In DCE, a LAN is defined by a LAN profile; the default LAN profile used by DTS is /.:/lan-profile. All DTS servers on a given LAN are called *local time servers*. They all query each other, and are queried by the time clerks in that LAN. Figure 67 shows an example of this relation.

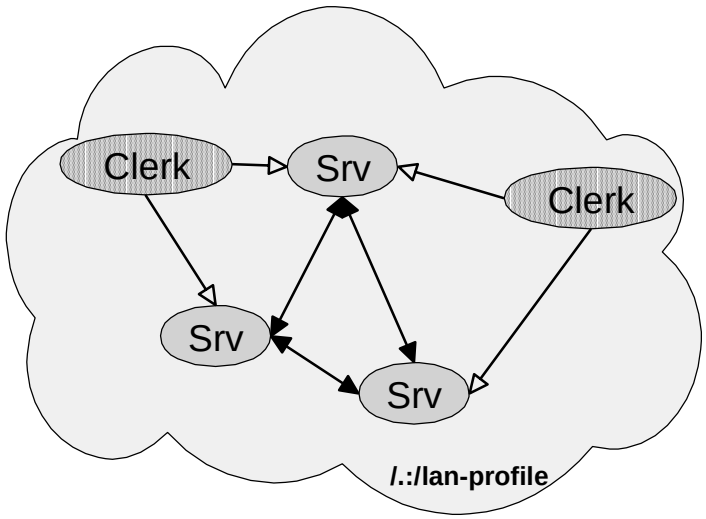


Figure 67. DTS Time Clerks and Local Servers

A *global time server* is a time server outside the LAN. Global time servers usually synchronize their time with external time providers. Local DTS servers are categorized as courier, backup courier, and noncourier servers depending on their relation to global time servers. A *courier* time server is a time server in a LAN which synchronizes with a global time server outside the courier's LAN. So courier time servers import an outside time to the LAN. *Backup courier* time servers become courier time servers if no courier is available in the LAN. *Noncourier* time servers never synchronize with global time servers. Figure 68 on page 137 shows the different local and global time servers.

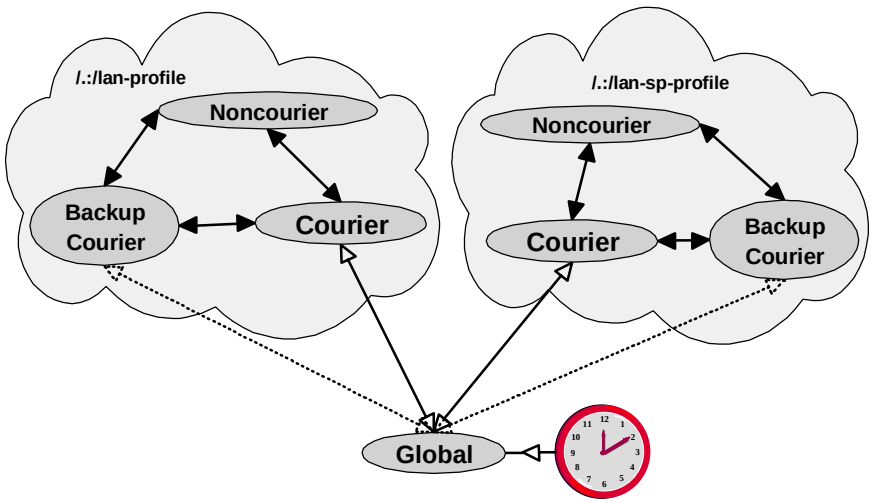


Figure 68. Local (Courier, Backup Courier, Noncourier) and Global DTS Servers

In order to ensure that the distributed systems' clocks are also synchronized to the correct time, DTS provides a time-provider interface which can be used to import a reference source like an atomic clock to a DTS server. This time, which can be very accurate, is then communicated to the other DTS servers.

DTS can coexist with NTP in the sense that both NTP servers can provide time to a DTS system, and DTS servers can provide time to a NTP domain.

6.2.2 Configuring and Using DTS on the SP

If DTS is to be used as the SP time service, NTP should be disabled in the SP site environment setup by calling `spsitenv ntp_config=none`. In this case, no time management configuration is done by PSSP.

The control workstation and SP nodes have to be configured as DCE clients, including the DTS client part. DCE configuration is outside the scope of this book, and should be planned together with the administrator of the DCE cell.

Concerning the required privileges, a sensible approach would be to perform a split configuration. This means that the DCE administrator sets up the required host principals and performs the necessary DCE server configuration with cell_admin privileges, and the SP administrator performs the local DCE client configuration using `mkdce -o local`, which only requires local root privileges:

```
mkdce -o local -n CELL_NAME -c CDS_SERVER -s SEC_SERVER -p LAN_PROFILE \
```


Ex.1010 / Page 161 of 572

```
-h HOSTNAME rpc sec_cl cds_cl dts_cl
```

After this configuration and after each reboot, the DCE client services will be automatically started by `/etc/rc.dce`. DTS is implemented by the `dtsd` daemon, whose personality is controlled by command line options: `dtsd -c` starts it as a clerk, and `dtsd -s` runs dtsd as a server. In server mode, dtsd can be further configured as follows:

- By default, dtsd runs as a backup courier time server.
- Specifying `-k courier` or `-k noncourier` starts dtsd as courier or noncourier time server, respectively.
- The `-g` flag indicates that dtsd should run as a global time server.

After the DCE client configuration, the use of DTS should be transparent. The DTS time, including the inaccuracy described by the interval representation, can be queried through an API or through the command `dtsdate`:

```
> date
Tue Feb  9 04:17:25 CET 1999
> dtsdate
1999-02-09-04:17:18.356+01:00I4.022
```

The format of the `dtsdate` output is date and time in Universal Coordinated Time (UTC) in the form YYYY-MM-DD-hh:mm:ss.sss, followed by the Time Differential Factor (TDF) for use in different time zones, followed by the inaccuracy in seconds.

Depending on the size and network connectivity of the SP (and the reset of the DCE cell), it might be advisable to set up additional DTS servers inside the SP, or even establish a separate LAN profile for the SP.

Ex.1010 / Page 162 of 572

Chapter 7. SP Security

This chapter discusses some security issues which are of special importance on the SP. With respect to security, the SP basically is a cluster of RS/6000 workstations connected on one or more LANs, with the control workstation serving as a central point to monitor and control the system. Therefore, the SP is exposed to the same security threats as any other cluster of workstations connected to a LAN. The control workstation, as the SP's single point of control, is particularly sensitive: if security is compromised here, the whole system will be affected.

Computer security is a large topic, and this book does not attempt to replicate the abundant literature on security of standalone or networked workstations.

Instead, after summarizing some general security concepts, we devote the remainder of this chapter to SP-specific security issues. Some aspects of base AIX which are closely related to security are discussed in 7.2, "AIX Security" on page 141. In 7.3, "How Kerberos Works" on page 148 we give a high-level explanation of Kerberos, the trusted third-party authentication system which is used in PSSP, DCE, and optionally also in the base AIX operating system. In 7.4, "Managing Kerberos on the SP" on page 156 we focus on the implementation of Kerberos on the SP. The services which use Kerberos are described in 7.5, "SP Services Which Utilize Kerberos" on page 170.

7.1 General Security Concepts

Before discussing technical details of computer security, it should be emphasized that security must be based on a clearly expressed and understood security policy. This is a management issue, and each organization should devote sufficient time to establish a reasonable and consistent security concept. This should include a discussion about which individuals or groups should have access to which resources, and what kind of data has to be restricted in which way. In addition, the risks of potential security breaches have to be evaluated with respect to both the costs to implement reasonable security measures, and the influence of these measures on the ease of use of the overall system. An important cost factor is the time required for continuous security administration and auditing, which is crucial to maintain security over time.

When computers were monolithic systems without any network connections to other computers, security was much more confined than today. Each computing system could be treated independently from all others. It had (at

© Copyright IBM Corp. 1999

least conceptually) a single administrator or root user, a single, global user database and access control information, and terminals as well as data storage devices were directly attached to the system. If dial-up connections were used, they were normally much more restricted and reliable than today's internet connections. To access the system from a terminal, users typically had to identify themselves by a user name, and prove this identity by a password which was checked against the central user database. Access rights for all users were recorded and enforced in a central place.

In today's distributed environments, many autonomous computing systems are interacting. They are often connected through an unreliable (in terms of security) network, which is used for remote login or command execution, remote data access, and also for remote system management. Each of these computers may have its own administrator, and its own user database and file space. The services they offer are often based on the client/server model. The client software component requests a service on behalf of the user, like a login request or access to a remote file. The server, which probably runs on a different machine, processes that request after checking the client's identity and permissions. There are normally three steps involved in these checks:

1. Identification. The identity of the user is presented, for example by a user name or numerical user ID(UID).
2. Authentication. Some kind of credentials are provided to prove this identity. In most cases this is a personal password, but other means of authentication can be used.
3. Authorization. The permissions of the client to perform the requested action are checked. These permissions and their management are different depending on the desired action. For example, file access is typically controlled through the UNIX mode bits, and login authorization is normally controlled by the user database (which also happens to store the password, in other words the authentication information).

Apart from the fact that most security exposures are still caused by poorly chosen passwords, the insecure network introduces several new security threats. Both partners of a client/server connection may be impersonated, that is, another party might pretend to be either the client (user) or server. Connections over the network can be easily monitored, and unencrypted data can be stored and reused. This includes capturing and replaying of user passwords or other credentials, which are sent during the setup of such client/server connections.

A common way to prevent impersonation and ensure the integrity of the data that is transferred between client and server is to set up a trusted third party

Ex.1010 / Page 164 of 572

system, which provides authentication services and optionally, encrypts all the communications to allow secure communication over the physically insecure network. A popular system which serves this task is Kerberos, designed and developed by the Massachusetts Institute of Technology (MIT) as part of the Athena project. Kerberos is described in 7.3, “How Kerberos Works” on page 148.

7.2 AIX Security

Since AIX is running on the control workstation and all the SP nodes, it is also AIX which provides the basic security features for the SP. AIX 4.3.1 has been certified to conform to the C2 level of trust classification by the US National Security Agency (NSA), and provides many facilities for discretionary security control. A good description of the basic security features of AIX Version 4 can be found in the redbook *Elements of Security: AIX 4.1*, GG24-4433. This publication discusses the following main topics:

- Managing user accounts and login control
- Access control for files and directories (including device special files)
- Network security and control of remote command execution
- Logging and auditing facilities

Although written for AIX 4.1, most of its content is still valid for AIX 4.3. Here we focus on some issues which are not discussed in detail in that redbook, or have been newly introduced into AIX since then.

7.2.1 Secure Remote Execution Commands

In AIX 4.3.1, the commands `telnet` and `ftp` as well as the r-commands `rcp`, `rlogin` and `rsh` have been enhanced to support multiple authentication methods (note that `rexec` is not included in this list). In earlier releases, the standard AIX methods were used for authentication and authorization:

telnet The `telnet` client establishes a connection to the server, which then presents the login screen of the remote machine, asking for userid and password. These are transferred over the network, and are checked by the server’s `login` command. This process normally performs both authentication and authorization.

ftp Again, userid and password are requested. Alternatively, the login information for the remote system (the server) can be provided in a $HOME/.netrc file on the local machine (the client), which is then read by the `ftp` client rather than querying the user. This

method is discouraged, since plain text passwords should not be stored in the (potentially remote and insecure) file system.

rexec Same as `ftp`. As mentioned previously, use of $HOME/.netrc files is discouraged.

The main security concern with this authentication for these commands is the fact that passwords are sent in plain text over the network. They can be easily captured by any root user on a machine which is on the network(s) through which the connection is established.

rcp, rlogin and rsh The current user name (or a remote user name specified as a command line flag) is used, and the user is prompted for a password. Alternatively, a client can be authenticated by its IP name/address if it matches a list of trusted IP names/addresses, which is stored in files on the server:

- /etc/hosts.equiv lists the hosts from which incoming (client) connections are accepted. This works for all users except root (UID=0).
- $HOME/.rhosts lists additional hosts, optionally restricted to specific userids, which are accepted for incoming connections. This is on a per-user basis, and also works for the root user.

Here, the primary security concern is host impersonation: it is relatively easy for an intruder to set up a machine with an IP name/address listed in one of these files, and gain access to the system. Of course, if a password is requested rather than using $HOME/.rhosts or /etc/hosts.equiv files, this is normally also sent in plain text.

With AIX 4.3.1, all these commands except `rexec` also support Kerberos Version 5 authentication. The base AIX operating system does not include Kerberos; we recommend that DCE for AIX Version 2.2 be used to provide Kerberos authentication. Note that previous versions of DCE did not make the Kerberos services available externally. However, DCE for AIX Version 2.2, which is based on OSF DCE Version 1.2.2, provides the complete Kerberos functionality as specified in RFC 1510, The Kerberos Network Authentication Service (V5).

For backward compatibility with PSSP 3.1 (which still requires Kerberos Version 4 for its own commands), the AIX r-commands `rcp` and `rsh` also support Kerberos Version 4 authentication. See 7.3, “How Kerberos Works” on page 148 for details on Kerberos.

Authentication methods for a machine are selected by the AIX `chauthent` command and can be listed with the `lsauthent` command. These commands call the library routines set_auth_method() and get_auth_method() which are contained in a new library, libauthm.a. Three options are available: `chauthent -std` enables standard AIX authentication, `chauthent -k5` and `-k4` enable Version 5 or 4 Kerberos authentication. More than one method can be specified, and authenticated applications/commands will use them in the order specified by `chauthent` until one is successful (or the last available method fails, in which case access is denied). If standard AIX authentication is specified, it must always be the last method.

Attention

On the SP, the `chauthent` command should not be used directly. The authentication methods for SP nodes and the control workstation are controlled by the partition-based PSSP commands `chauthpar` and `lsauthpar`. Configuration information is stored in the Syspar SDR class, in the auth_install, auth_root_rcmd and auth_methods attributes.

If Kerberos Version 5 is activated as an authentication method, the `telnet` connection is secured by an optional part of the telnet protocol specified in RFC 1416, Telnet Authentication Option. Through this mechanism, client and server can negotiate the authentication method. The `ftp` command uses another mechanism: here the authentication between client and server takes place through a protocol specified in RFC 1508, Generic Security Service API. These extensions are useful, but have no direct relation to SP system administration. An impediment to widespread use of these facilities is that they rely on all clients being known to the Kerberos database, including the clients' secret passwords. Refer to 7.2.3, "Secure Shell" on page 146 for a public key method to provide secure remote login and remote data copy that do not have this restriction.

The Kerberized `rsh` and `rcp` commands are of particular importance for SP, as they replace the corresponding Kerberos Version 4 authenticated r-commands which were part of PSSP Versions 1 and 2. Only the PSSP versions of `rsh`, `rcp` and `kshd` have been removed from PSSP 3.1, it still includes and uses the Kerberos Version 4 server. This Kerberos server can be used to authenticate the AIX r-commands. A full description of the operation of the `rsh` command in the SP environment can be found in 7.5.2, "Remote Execution Commands" on page 173, including all three possible authentication methods.

7.2.2 Securing X11 Connections

If configured improperly, the X Windows system can be a major security hole. It is used to connect X servers (machines with a graphical display like the SP administrator's workstation) with X clients (machines which want to display graphical output on an X server, like the SP control workstation). If the X server is not secured, then everybody can monitor, or even control, the X server's resources. This includes the keyboard and screen, so everything which is typed or displayed on an unprotected X server can be monitored.

There are two access control commands to restrict X connections to an X server: `xhost`, which controls access based on IP names/addresses of the clients, and `xauth`, which provides more fine-grained access control through the use of *cookies*. Both are contained in the X11.apps.config fileset.

The `xhost` command is the simplest way to restrict access; it has to be invoked on the X server (the machine whose display/keyboard is to be secured). Entering the command without parameters lists the current setup, `xhost +` or `xhost -` globally enables or disables access to the X server, and individual client hosts can be added or removed by specifying the host's name after the plus or minus parameter. The following sequence of commands illustrates this. At the end of the sequence only X clients running on the control workstation sp5cw0 can access the local X server.

```
> which xhost
xhost is a tracked alias for /usr/bin/X11/xhost

> xhost +
access control disabled, clients can connect from any host

> xhost -
access control enabled, only authorized clients can connect

> xhost -
access control enabled, only authorized clients can connect
> xhost + sp5cw0
sp5cw0 being added to access control list
> xhost
access control enabled, only authorized clients can connect
sp5cw0.itso.ibm.com
```

However, this is insecure if the client machine is a multi-user machine. Anybody who has a login on sp5cw0 is allowed to connect to the local machine's X server, and can for example record passwords typed in at this workstation.

To limit access to specific users (for example the root user on sp5cw0), the Xauthority mechanism is used. When the X server starts up, it generates a

Ex.1010 / Page 168 of 572

secret key called a cookie (normally in a format called MIT-MAGIC-COOKIE-1, although other formats exist), and stores this key in the file $HOME/.Xauthority in the home directory of the user who started the X server. If an X client wants to access the X server, it needs to know this key and has to present it to the X server.

The local user who started the X server immediately has access to it, but all other users on the X server machine and all users on other X client machines first need to get this key. This transfer has to be secured, of course. Securely transferring the key can be challenging. Using an NFS-mounted file system, for example, cannot be considered secure since it is relatively easy to bypass file access permissions of NFS-exported file systems. If the shared file system is in AFS or DFS, this is much more secure. If there is no shared file system, an actual copy has to be performed, which might also expose the key.

The `xauth` command can be used on the client machine to add the cookie to the .Xauthority file of the user whose processes want to access the X server, as shown in the following example:

```
> which xauth
xauth is a tracked alias for /usr/bin/X11/xauth

> xauth info
Authority file:       /home/joe/.Xauthority
File new:             No
File locked:          Yes
Number of entries:    2
Changes honored:      Yes
Changes made:         No
Current input:        (argv):1

> xauth list
desktop/unix:0  MIT-MAGIC-COOKIE-1 1234567890abcdef084a48716447704c
desktop.itso.ibm.com:0  MIT-MAGIC-COOKIE-1 1234567890abcdef084a48716447704c

> xauth add risc123:0 MIT-MAGIC-COOKIE-1 234567890abcdef084a48716447704c
> xauth remove risc123:0
```

The `xauth list` command displays the contents of the .Xauthority file, showing one line per <hostname>:<display> pair. The cookies are displayed as a string of 32 hex digits. Individual keys for a <hostname>:<display> pair can be added and removed by the `add` and `remove` options, using the same format as the `list` output. Each time the X server starts, it creates a new cookie which has to be transferred to its clients.

The secure shell described in 7.2.3, “Secure Shell” on page 146 transparently encrypts all X traffic in a separate channel. On the remote node, a new

.Xauthority file is created, and the cookie is used to secure the communication between the sshd daemon and the X client. On the local node, the cookie is necessary to set up a secure connection between the X server and the ssh client. This is very convenient, as both the text-based login and any X Windows traffic are automatically protected.

7.2.3 Secure Shell

Although AIX now includes a Kerberized telnet, as outlined in 7.2.1, “Secure Remote Execution Commands” on page 141, using this option for a secure remote login to a machine requires two important prerequisites:

1. The telnet client must support the authentication option. Even though it is available for some platforms (including AIX 4.3.1, of course), this option is definitely not widely used.
2. The Kerberized telnet client must share a secret key with the Kerberos server that manages the telnet server. As discussed in 7.3, “How Kerberos Works” on page 148, Kerberos uses *symmetric* encryption which requires both parties to know a common key. Securely transferring this key from one of the parties to the other before any Kerberized connection takes place can be challenging, and the effort involved might distract from using the feature at all.

The standard method to circumvent the second problem is to use a public key method, where encryption is *asymmetric*. Such encryption methods use two keys, a *private key* and a *public key*. Messages encrypted with one of these keys can only be decrypted with the other one. Using this system, there is no need to share a secret key. The public keys of all participants are publicly available, and the private key needs only be accessible by its owner.

The Secure Shell (SSH) is a de facto industry standard for secure remote login using public key encryption. It is widely used, and available for almost every operating system (which also solves the first problem mentioned previously). SSH provides secure remote login through the `ssh` or `slogin` commands, and a secure remote copy command `scp`. As mentioned in 7.2.2, “Securing X11 Connections” on page 144, a very useful feature of SSH is its ability to transparently and securely handle X Windows traffic. We therefore highly recommend that you use the Secure Shell for remote login and remote data copy.

More information on the Secure Shell can be found at the SSH home page:
`http://www.ssh.fi/sshprotocols2/`

Ex.1010 / Page 170 of 572

7.2.4 Login Control in AIX

By default, AIX uses the local user database (/etc/passwd, /etc/group, and files in /etc/security/) to authenticate and authorize a user for login, with the option to use NIS if this is configured. Through the AIX concept of loadable authentication modules, it is possible to add any customer-supplied login authentication methods to the system. These can supplement or completely replace the normal AIX methods. Which authentication method should be used can be controlled on a per-user basis, through stanzas in the AIX /etc/security/user file.

DCE for AIX has its own user management infrastructure, and provides a loadable authentication module /usr/lib/security/DCE to be used with the standard AIX login mechanism. To enable an integrated login, which provides access to the machine and at the same time establishes a DCE context for the user (including DCE and Kerberos Version 5 credentials), the DCE authentication module should be added as a stanza to /etc/security/login.cfg and configured for all non-system users in /etc/security/user. The standard AIX users, notably root, should normally be authenticated through the local user database only, so that they can still access the system when DCE is down. The relevant stanzas are:

```
/etc/security/login.cfg:

DCE:
        program = /usr/lib/security/DCE



/etc/security/user:

default:
        auth1 = SYSTEM
        auth2 = NONE
        SYSTEM = "DCE"
        dce_export = false
        ...

root:
        SYSTEM = "files"
        registry = files
        ...
```

More details can be found in the DCE for AIX documentation. Note that if the integrated DCE login is enabled, then by default all principals in the DCE cell will be able to log in to that host. This is probably not what is intended, but unfortunately DCE does not provide an easy means to handle this situation.

Ex.1010 / Page 171 of 572

There is a mechanism to restrict access to a host by means of a passwd_overwrite file, which must exist locally on the host and excludes users listed in the file from login. However, maintaining this file for all hosts in the DCE cell is laborious, and care has to be taken that each time a principal is added to the DCE cell the passwd_overwrite file is updated on all hosts in that cell to reflect this addition.

It would be more convenient to include users, rather than exclude all others. Unfortunately, such a feature is not available in DCE.

7.3 How Kerberos Works

Kerberos

Also spelled Cerberus - The watchdog of Hades, whose duty was to guard the entrance (against whom or what does not clearly appear) ... It is known to have had three heads.

- Ambrose Bierce, The Enlarged Devil's Dictionary

Kerberos is a trusted third-party authentication system for use on physically insecure networks. It allows entities communicating over the network to prove their identity to each other while preventing eavesdropping or replay attacks. Figure 69 on page 149 shows the three parties involved in the authentication. The Kerberos system was designed and developed in the 1980's by the Massachusetts Institute of Technology (MIT), as part of the Athena project. The current version of Kerberos is Version 5, which is standardized in RFC 1510, The Kerberos Network Authentication Service (V5).

Ex.1010 / Page 172 of 572

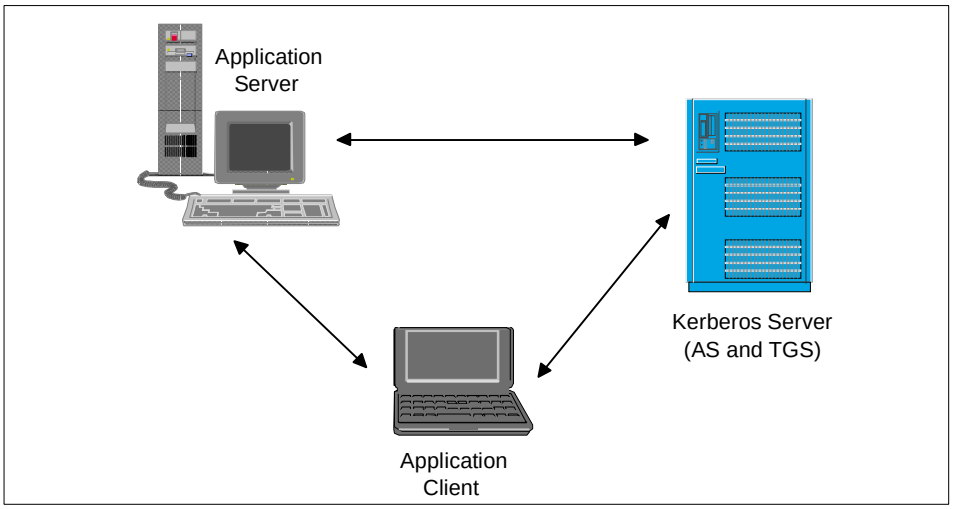


Figure 69. Partners in a Third-Party Authentication

Kerberos provides two services to Kerberos *principals* (users or services): an Authentication Service (AS) and a Ticket-Granting Service (TGS). Principals can prove their identity to the AS by a single sign-on, and will get a *Ticket-Granting Ticket* (TGT) back from the AS. When one authenticated principal (the client) wants to use services of a second authenticated principal (the server), it can get a *Service Ticket* for this service by presenting its TGT to the Kerberos TGS. The Service Ticket is then sent from the client to the server, which can use it to verify the client's identity.

This section describes the protocol that Kerberos uses to provide these services, independently of a specific implementation. A more detailed rationale for the Kerberos design can be found in the MIT article *Designing an Authentication System: a Dialogue in Four Scenes* available from the following URL: `ftp://athena-dist.mit.edu/pub/kerberos/doc/dialogue.PS`

7.3.1 Kerberos Keys and Initial Setup

To encrypt the messages which are sent over the network, Kerberos uses a *symmetric* encryption method, normally the Data Encryption Standard (DES). This means that the same key is used to encrypt and decrypt a message, and consequently the two partners of a communication must share this key if they want to use encryption. The key is called *secret key* for obvious reasons: it must be kept secret by the two parties, otherwise the encryption will not be effective.

Ex.1010 / Page 173 of 572

This approach must be distinguished from *public key* cryptography, which is an *asymmetric* encryption method. There, two keys are used: a *public key* and a *private key.* A message encrypted with one of the keys can only be decrypted by the other key, not by the one which encrypted it. The public keys do not need to be kept secret (hence the name “public”), and a private key is only known to its owner (it is not even to the communication partner as in the case of symmetric cryptography). This has the advantage that no key must be transferred between the partners prior to the first use of encrypted messages.

With symmetric encryption, principals need to provide a password to the Kerberos server before they can use the Kerberos services. The Kerberos server then encrypts it and stores the resulting key in its database. This key is the shared information that the Kerberos server and the principal can use to encrypt and decrypt messages they send each other. Initially, two principals who want to communicate with each other do not share a key, and so cannot encrypt their messages. But since the Kerberos server knows the keys of all the principals, it is called a *trusted third party* and can securely provide a common session key to the two parties.

Obviously, the initial passwords have to be entered securely, if possible at the console of the Kerberos server machine. They might also be generated by the Kerberos server (especially if the principal is a host or service). In that case they must be securely transferred to the principal which stores (or remembers) them.

7.3.2 Authenticating to the Kerberos Server

If a principal (typically a user) wants to use Kerberos services, for example because it wants to use an application service which requires Kerberos authentication, it first has to prove its identity to the Kerberos server. This is done in the following way:

A command to sign on to the Kerberos system is issued on the application client, typically `kinit` or `k4init`. This command sends an authentication request to the Kerberos server, as shown in Figure 70 on page 151. This contains the type of service that is requested (here, the client wants to get service from the Ticket-Granting Service), the client’s (principal’s) name, and the IP address of the client machine. This request is sent in plain text. Note that the principal’s password is not sent in this packet, so there is no security exposure in sending the request in plain text.

Ex.1010 / Page 174 of 572

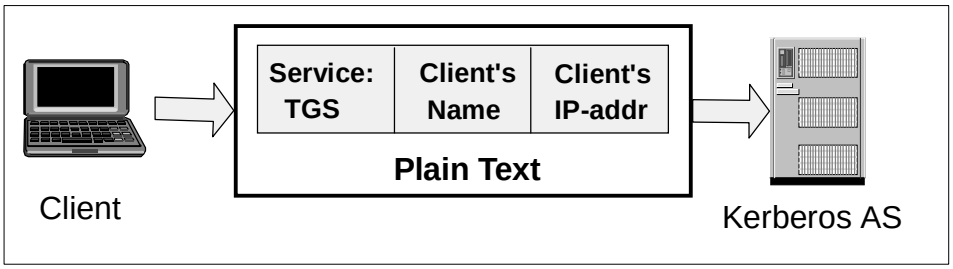


Figure 70. Client's Authentication Request

The request is processed by the Authentication Server. Using the client's name, it looks up the corresponding key in the Kerberos database. It also generates a random *session key* to be shared by the client and the TGS, which will be used to encrypt all future communication of the client with the TGS. With this information, the AS constructs the Ticket-Granting Ticket for the client, which (as all Kerberos tickets) contains six parts:

1. The service for which the ticket is good (here, the TGS)
2. The client's (principal's) name
3. The client machine's IP address
4. A timestamp showing when the ticket was issued
5. The ticket lifetime (maximum 21.25 hours in MIT K4, 30 days in PSSP K4, configurable in K5)
6. The session key for Client/TGS communications

This ticket is encrypted with the secret key of the TGS, so only the TGS can decrypt it. Since the client needs to know the session key, the AS sends back a reply which contains both the TGT and the session key, all of which is encrypted by the client's secret key. This is shown in Figure 71 on page 152.

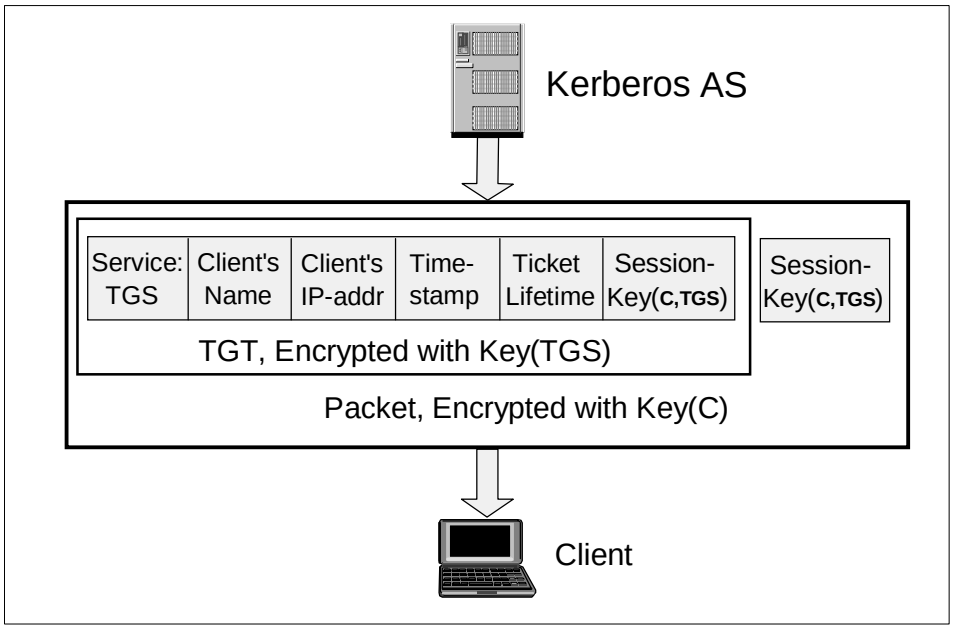


Figure 71. Authentication Server's Reply: TGT

Now the sign-on command prompts the user for the password, and generates a DES key from the password using the same algorithm as the Kerberos server. It then attempts to decrypt the reply message with that key. If this succeeds, the password matched the one used to create the user's key in the Kerberos database, and the user has authenticated herself. If the decryption failed, the sign-on is rejected and the reply message is useless. Assuming success, the client now has the encrypted Ticket-Granting Ticket and the session key for use with the TGS, and stores them both in a safe place. Note that the authentication has been done locally on the client machine, the password has not been transferred over the network.

7.3.3 Authenticating to an Application Server

If the client now wants to access an application service which requires Kerberos authentication, it must get a service ticket from the Ticket-Granting Service. The TGT obtained during the Kerberos sign-on can be used to authenticate the client to the TGS; there is no need to type in a password each time a service ticket is requested.

If the client sent only the (encrypted) TGT to the Kerberos TGS, this might be captured and replayed by an intruder who has impersonated the client

Ex.1010 / Page 176 of 572

machine. To protect the protocol against such attacks, the client also generates an *authenticator* which consists of three parts:

1. The client's (principal's) name
2. The client machine's IP address
3. A timestamp showing when the authenticator was created

The authenticator is encrypted with the session key that the client shares with the TGS. The client then sends a request to the TGS consisting of the name of the service for which a ticket is requested, the encrypted TGT, and the encrypted authenticator. This is shown in Figure 72.

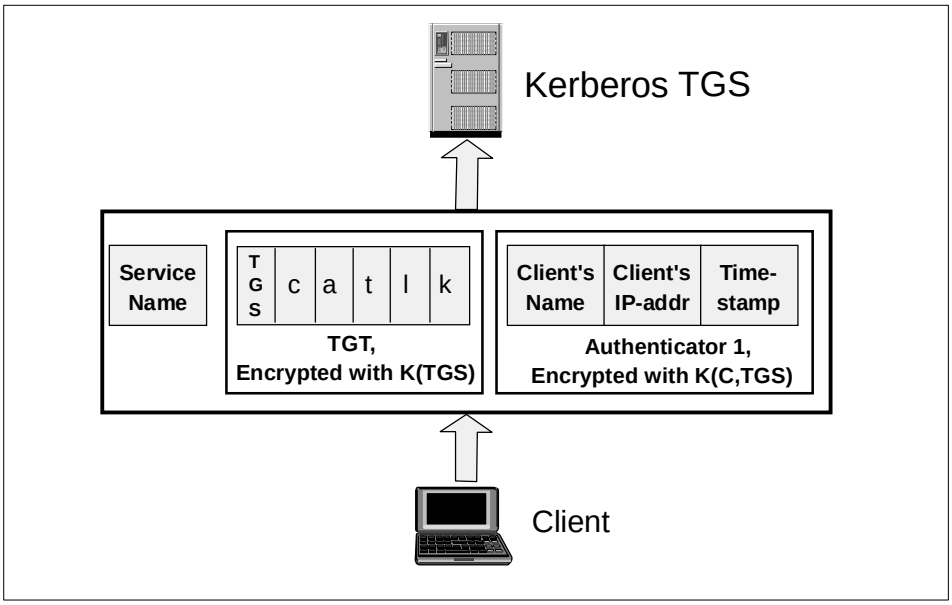


Figure 72. Client's Service Ticket Request

The Ticket-Granting Server can decrypt the TGT since it is encrypted with its own secret key. In that ticket, it finds the session key to share with the client. It uses this session key to decrypt the authenticator, and can then compare the client's name and address in the TGT and the authenticator.

If the timestamp that the TGS finds in the authenticator differs from the current time by more than a prescribed difference (typically 5 minutes), a ticket replay attack is assumed and the request is discarded.

If all checks pass, the TGS generates a service ticket for the service indicated in the client's request. The structure of this service ticket is identical to the

TGT described in 7.3.2, “Authenticating to the Kerberos Server” on page 150. The content differs in the service field (which now indicates the application service rather than the TGS), the timestamp, and the session key. The TGS generates a new, random key that the client and application service will share to encrypt their communications. One copy is put into the service ticket (for the server), and another copy is added to the reply package for the client since the client cannot decrypt the service ticket. The service ticket is encrypted with the secret key of the service, and the whole package is encrypted with the session key that the TGS and the client share. The resulting reply is shown in Figure 73. Compare this to Figure 71 on page 152.

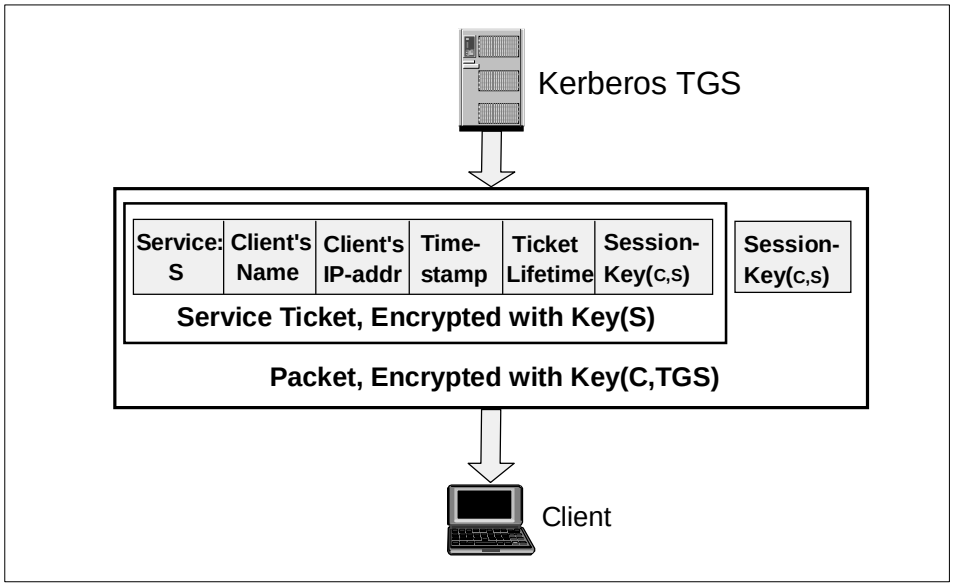


Figure 73. Ticket Granting Service's Reply: Service Ticket

The client can decrypt this message using the session key it shares with the TGS. It then stores the encrypted service ticket and the session key to share with the application server, normally in the same *ticket cache* where it already has stored the TGT and session key for the TGS.

To actually request the application service, the client sends a request to that server which consists of the name of the requested service, the encrypted service ticket, and a newly generated authenticator to protect this message against replay attacks. The authenticator is encrypted with the session key that the client and the service share. The resulting application service request

Ex.1010 / Page 178 of 572

is shown in Figure 74. Note the resemblance to the request for Ticket-Granting Service in Figure 72 on page 153.

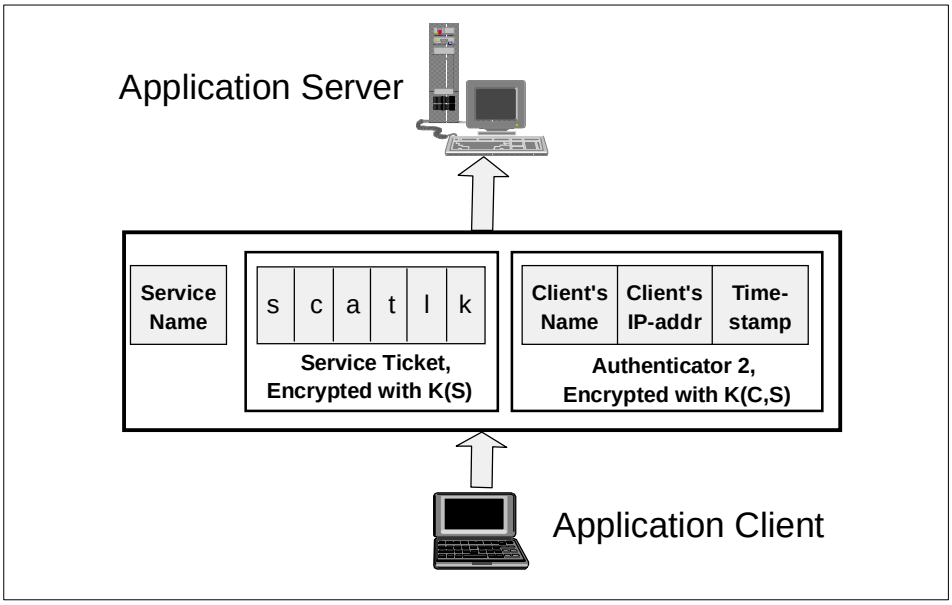


Figure 74. Client's Application Service Request

The application server decrypts the service ticket with its secret key, uses the enclosed session key to decrypt the authenticator, and checks the user's identity and the authenticator's timestamp. Again, this processing is the same as for the TGS processing the service ticket request. If all checks pass, the server performs the requested service on behalf of the user.

Attention

Authorization: Kerberos is only responsible for authenticating the two partners. Any authorization mechanism must be enforced by the application itself.

If the client required mutual authentication (that is, the service has to prove its identity to the client), the server could send back a message which is encrypted by the session key it shares with the client, and an application-dependent contents that the client can verify. Since the service can only know the session key if it was able to decrypt the service ticket, it must have known its secret key and so has proven its identity.

7.4 Managing Kerberos on the SP

The basic functionality of Kerberos and the protocol used to provide its services was described in the previous section. In this section, we focus on the implementations of Kerberos used on the SP.

One of the difficulties in managing Kerberos is that many different implementations exist. There are two widely used versions of the protocol, Version 4 and Version 5. Unfortunately, these two protocols are incompatible. Apart from that, implementations of the same version sometimes also differ. Not only are different commands, files and daemons used to implement the services, sometimes even the interpretation of some of the protocol's details differs or has been modified to fulfil a particular application request. Typical examples include different methods to generate encryption keys from passwords, and different requirements concerning ticket lifetimes.

SP Security in PSSP 3.1 is still based on Kerberos Version 4. The main differences among the PSSP, MIT, and AFS implementations of Kerberos Version 4 are described in an IBM white paper *Security on the RS/6000 SP system,* available at:
`http://www.rs6000.ibm.com/resource/technology/sp_security.ps`

More details on the PSSP and AFS implementations can also be found in Chapter 12, "Security Features of the SP System" of the *PSSP for AIX Administration Guide*, SA22-7348.

Note that although PSSP 3.1 also tolerates Kerberos Version 5 authentication (for the AIX authenticated r-commands only), it does not actively support it. In 7.4.3, "DCE Kerberos (Version 5)" on page 168 we describe the manual setup of Kerberos Version 5 on the SP. Full PSSP support for DCE/Kerberos Version 5 will probably be added in later releases.

7.4.1 PSSP Kerberos (Version 4)

In this section, we describe the components (files, commands, daemons) which make up the PSSP implementation of Kerberos Version 4, and summarize how they are initially created and maintained. This is a high level description to provide some overview of the general concepts and procedures. To actually perform the individual administrative steps, the PSSP documentation should be consulted. Also note that we describe the logical components independently from each other. More than one of them might be present on an actual machine. For example, the control workstation typically is the Kerberos server, an application client where principals log on, and an

Ex.1010 / Page 180 of 572

application server for all the services described in 7.5, “SP Services Which Utilize Kerberos” on page 170.

7.4.1.1 Global Configuration Information

In Kerberos, the set of machines which is administered together, and served by a single logical authentication database, is called a *realm*. The PSSP software by default sets up the IP domain name of the Kerberos server machine (in upper case letters) as the realm name, but any name can be selected. This realm name, and the name of the Kerberos server for that realm, is stored in plain text in the file /etc/krb.conf, which must be available on all machines which belong to the realm. For example, the /etc/krb.conf file for one of the ITSO SP systems is:

```
MSC.ITSO.IBM.COM
MSC.ITSO.IBM.COM  sp4en0.msc.itso.ibm.com  admin  server
```

It is possible to set up read-only copies of the Kerberos database on secondary Kerberos servers. In such cases, the configuration file would list the secondaries also. We recommend that you not set up secondary Kerberos servers unless there are performance problems with a single server.

Attention

Multiple Kerberos Realms with Same Name: It is possible to have several independent Kerberos realms with the same name (like two independent SP systems in the same IP domain). Since the basic configuration information of a Kerberos realm includes the realm name and the names of the Kerberos servers in that realm, this removes the ambiguity: each machine in a cells knows only the Kerberos servers within that cell. However, you might want to set up non-default realm names (like SP1.DOMAIN and SP2.DOMAIN) to avoid confusion.

A second global configuration file is /etc/krb.realms, which is also a plain text file. This file maps machines’ IP names to Kerberos realm names. It is primarily needed when non-default realm names are used, since a host which is not listed in this file will by default be allocated to a realm which equals its IP domain name. Note that if a node is listed in this file, there must be an entry for each IP name (network interface) which is configured on that host: Kerberos Version 4 is not able to deal with multi-homed hosts (hosts with more than one network connection) without knowing all the IP names. A sample /etc/krb.realms file would be:

```
sp4cw0.itso.ibm.com  MSC.ITSO.IBM.COM
```


Ex.1010 / Page 181 of 572

This lists the IP name of an external network connection of the control workstation, which is in the IP domain ITSO.IBM.COM rather than in the IP domain MCS.ITSO.IBM.COM which contains the SP system itself (and is the realm name). All other IP names of the system are in the default realm, and are therefore not listed here.

The third piece of global configuration information consists of the service names and port numbers used for communication between the Kerberos servers and clients. The file /etc/services reserves service names and port numbers. However, by default the entries used by PSSP 3.1 are not listed in this file. They are:

```
kerberos4       750/udp
kerberos_admin  751/tcp
krb_prop        754/tcp
```

We recommend that you add these services to the /etc/services file to make their use explicit, and also reserve both the tcp and udp ports. Note that there may be two other ports registered, for the services kerberos (port 88) and kerberos-adm (port 749). These are used for Kerberos Version 5, as provided by DCE for AIX Version 2.2.

Kerberos principals are identified by their name, instance, and realm.

- The realm by default is the IP domain name.
- For users, the name typically is their AIX login name. However, since there is no technical relation between these two, any name could be used. It is generally more convenient to the user and system administrator to match them.
- The instance part of a user principal can be empty, or can be the string "admin" which is interpreted by the PSSP authenticated services as an administrative account. The service principals bear the name of the service (or one which is at least closely related to the name of the service), and the instance part specifies on which machine the service is available. This is normally the hostname, and one principal exists for each node in the system (even for each network interface on each host, since Kerberos Version 4 cannot transparently handle multi-homed hosts).
- The output of the `lskp` command on a typical SP system, which lists all these principals as well as some principals which Kerberos uses internally, is shown in the following example.

Ex.1010 / Page 182 of 572

```
> /usr/lpp/ssp/kerberos/etc/lskp
K.M                tkt-life: 30d      key-vers: 1  expires: 2037-12-31 23:59
changepw.kerberos   tkt-life: 30d      key-vers: 1  expires: 2037-12-31 23:59
default            tkt-life: 30d      key-vers: 1  expires: 2037-12-31 23:59
krbtgt.MSC.ITSO.IBM.COM
                   tkt-life: 30d      key-vers: 1  expires: 2037-12-31 23:59
root.admin         tkt-life: 30d      key-vers: 6  expires: 2037-12-31 23:59
michael.admin      tkt-life: 30d      key-vers: 9  expires: 2037-12-31 23:59
hardmon.sp4cw0     tkt-life: Unlimited key-vers: 1  expires: 2037-12-31 23:59
hardmon.sp4en0     tkt-life: Unlimited key-vers: 1  expires: 2037-12-31 23:59
rcmd.sp4cw0        tkt-life: Unlimited key-vers: 1  expires: 2037-12-31 23:59
rcmd.sp4en0        tkt-life: Unlimited key-vers: 1  expires: 2037-12-31 23:59
rcmd.sp4n01        tkt-life: Unlimited key-vers: 1  expires: 2037-12-31 23:59
rcmd.sp4n05        tkt-life: Unlimited key-vers: 1  expires: 2037-12-31 23:59
rcmd.sp4n06        tkt-life: Unlimited key-vers: 1  expires: 2037-12-31 23:59
rcmd.sp4n07        tkt-life: Unlimited key-vers: 1  expires: 2037-12-31 23:59
rcmd.sp4n08        tkt-life: Unlimited key-vers: 1  expires: 2037-12-31 23:59
rcmd.sp4sw01       tkt-life: Unlimited key-vers: 1  expires: 2037-12-31 23:59
rcmd.sp4sw05       tkt-life: Unlimited key-vers: 1  expires: 2037-12-31 23:59
rcmd.sp4sw06       tkt-life: Unlimited key-vers: 1  expires: 2037-12-31 23:59
rcmd.sp4sw07       tkt-life: Unlimited key-vers: 1  expires: 2037-12-31 23:59
rcmd.sp4sw08       tkt-life: Unlimited key-vers: 1  expires: 2037-12-31 23:59
```

Note that hardmon and rcmd service principals are allowed to have unlimited lifetime, whereas all others are restricted to the default maximum of 30 days.

7.4.1.2 Application Client Components

Apart from the information described in 7.4.1.1, “Global Configuration Information” on page 157, the application client also needs the commands that allow a user principal to authenticate to the Kerberos system. These commands are `k4init`, `k4list` and `k4destroy`. Another command available to the client user is `kpasswd`, which can be used to change the user’s password in the Kerberos database.

Attention

DCE/PSSP name conflicts: DCE for AIX provides commands `/usr/bin/kinit`, `/usr/bin/klist` and `/usr/bin/kdestroy` for its own (Kerberos Version 5) authentication. PSSP used to establish symbolic links in /usr/bin/ to the PSSP versions of these commands, which reside in /usr/lpp/ssp/kerberos/bin/. To avoid name conflicts with DCE, these symbolic links have been renamed to `k4init`, `k4list` and `k4destroy` in PSSP 3.1.

Note that the PSSP executables in /usr/lpp/ssp/kerberos/bin/ are still called `kinit`, `klist` and `kdestroy`. So the $PATH variable must list /usr/bin before that PSSP path, otherwise PSSP will mask the DCE commands.

The sign-on by using the `k4init` command prompts for the principal’s password, and on success creates a ticket cache file which stores the

Ex.1010 / Page 183 of 572

Ticket-Granting Ticket, obtained through the protocol described in 7.3.2, “Authenticating to the Kerberos Server” on page 150. The ticket cache file is placed in the default location /tmp/tkt<uid>, where <uid> is the AIX numeric UID of the AIX user which issues `k4init`. This default location can be changed by setting the environment variable $KRBTKFILE. If the ticket cache file existed before, it will be replaced by the new one.

Figure 75 shows these components. It also shows a Kerberos authenticated application client, which normally uses Kerberos library calls to communicate with the Kerberos server and the application server. On its first invocation after a `k4init`, the application client will acquire a service ticket from the Kerberos server as described in 7.3.3, “Authenticating to an Application Server” on page 152, and store it in the ticket cache file. It then passes it to the application server each time it is invoked.

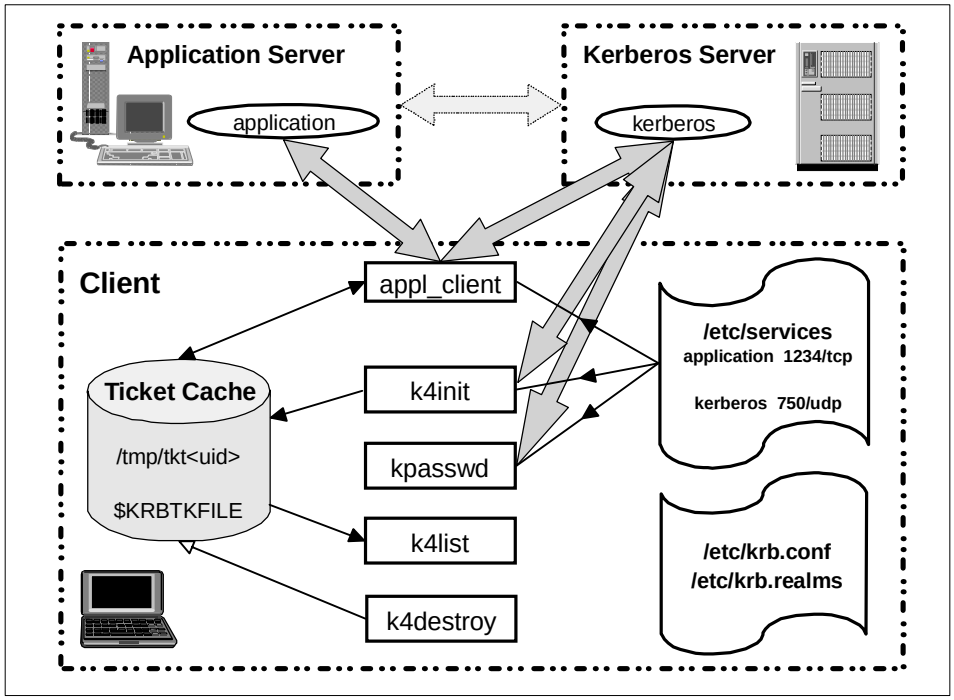


Figure 75. Application Client Components

7.4.1.3 Application Server Components

Like the application client, the application server also needs the files described in 7.4.1.1, “Global Configuration Information” on page 157.

Ex.1010 / Page 184 of 572

But while the application client's sign-on can use a password prompt when the user issues the `kinit` command, the server components typically run in the background, often as daemon processes. For these, it is inadequate to prompt for a password. So application servers directly store their secret key in a keytab file, which is located in /etc/krb-srvtab. This file must be protected from unauthorized access, as it contains the application server's credentials. It should normally be owned by the user which runs the application (often root), and have Unix mode bits set to 400. The contents of this file can be listed with the `k4list -srvtab` command. It can also be viewed and modified by the `ksrvutil` command, but this should not be necessary in normal operation.

If the application server needs to decrypt a request from a client, it uses its key in the /etc/krb-srvtab file. It does not require any communication with the Kerberos server. These interactions are shown in Figure 76.

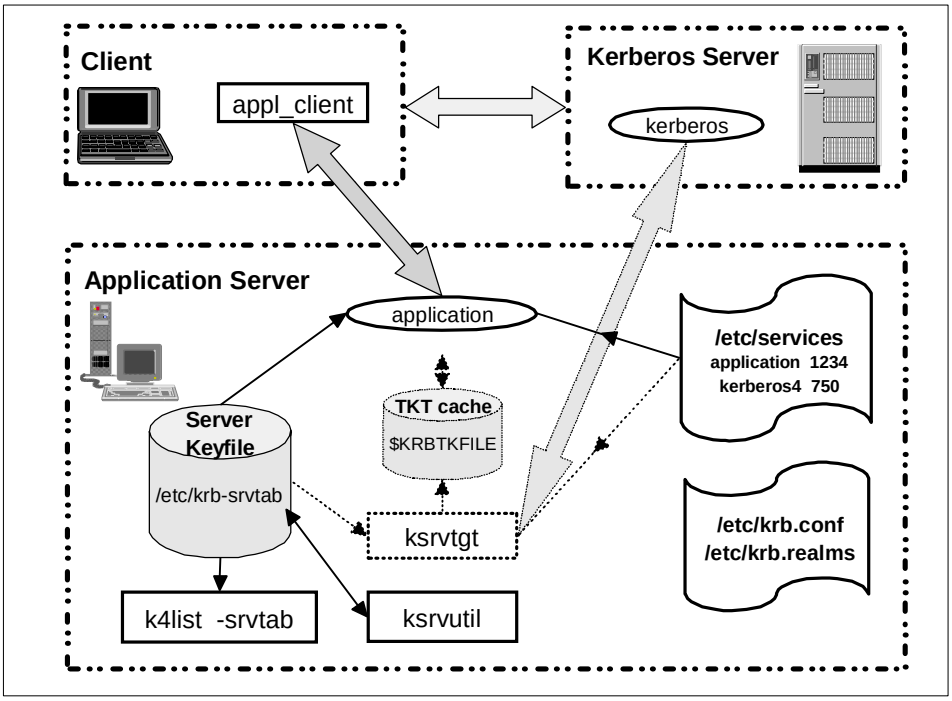


Figure 76. Application Server Components

However, there may be cases where the application is itself a client to another Kerberized service. In this case, the application needs to get authenticated to the Authentication Server (which returns a TGT), and eventually gets a

service ticket using the TGT. To enable this kind of scenario, the `ksrvtgt` command is available, which acquires a TGT from the Kerberos server by presenting the server's key from the /etc/krb-srvtab file. That TGT has only a lifetime of five minutes, to allow the application to get the desired service ticket, and then expires. The ticket cache file for such cases will be specified by the $KRBTKFILE environment variable, set by the application. The dashed part of Figure 76 shows this optional part.

7.4.1.4 Kerberos Server Components

The Kerberos server holds the Kerberos database and runs the kerberos daemon, which handles requests from Kerberos clients and reads the clients' secret keys and other properties from the database. The Kerberos server also runs the kadmind daemon, which can be used to administer parts of the database by the `add_principal`, `kadmin` and `kpasswd` commands. These commands can be issued on the local server machine, or on remote machines. There are a number of other administrative commands, the most important of which probably is `kdb_util`, which have to be executed on the Kerberos server machine. They are summarized in Figure 77 on page 163. Details on their use can be found in *PSSP for AIX Administration Guide*, SA22-7348.

Ex.1010 / Page 186 of 572

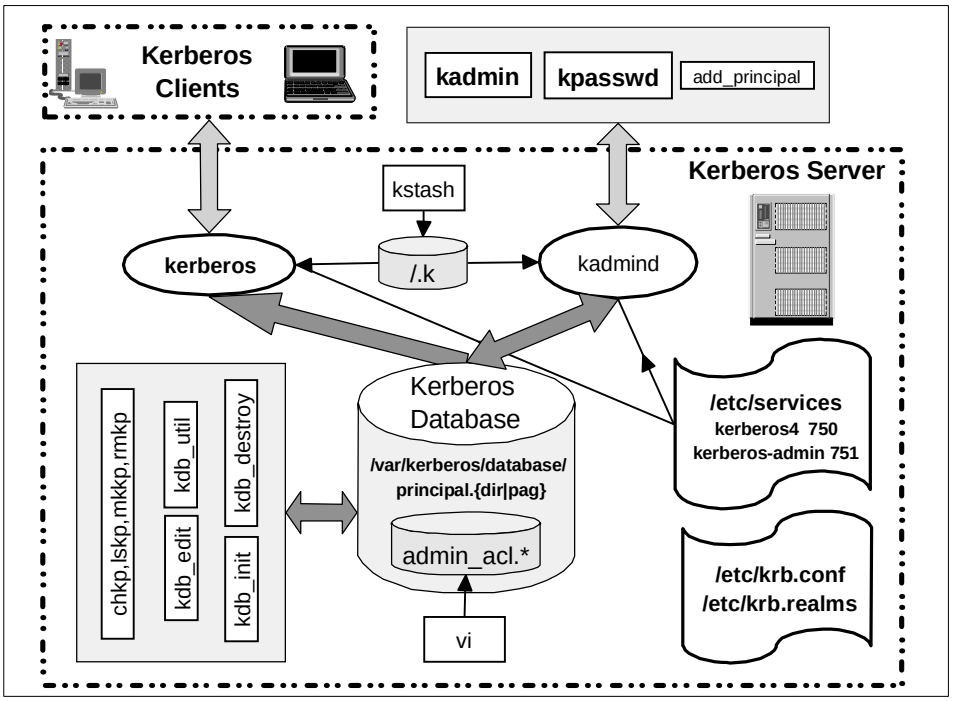


Figure 77. Kerberos Server Components

There are two aspects of PSSP Kerberos security that require special attention: the protection of the database itself, and access control to the administration commands.

The Kerberos database, which is stored in the two files principal.dir and principal.pag in /var/kerberos/database/, should only be accessible by the root user. In addition, it is encrypted with a key called the Kerberos master key. This key (and the password from which it is generated) should not be confused with the password of the administrative principal of the root user, root.admin. In particular, these two passwords should be different, and access to the master password should be even more restricted than access to the root or root.admin passwords.

The kerberos and kadmind daemons have to be able to access the encrypted database. To start them automatically, PSSP stores the master key in the /.k file, and the daemons read it during startup. The `kstash` command can be used to convert the master password to the corresponding key, and store it in the /.k file. This is shown in Figure 77. Like the database, the /.k file should be

Ex.1010 / Page 187 of 572

owned by root, and it should have permissions 600. The same file is also used by some of the administrative commands.

PSSP Kerberos also maintains its own Access Control Lists for Kerberos administrative commands, stored in three files: admin.add, admin.get and admin.mod. They are in the same directory as the database, /var/kerberos/database/. These files are plain text files, and list the principals which are authorized to perform the corresponding operations. An example is:

```
# cat /var/kerberos/database/admin_acl.add
root.admin

# cat /var/kerberos/database/admin_acl.get
root.admin
michael.admin

# cat /var/kerberos/database/admin_acl.mod
root.admin
```

This grants all rights to root.admin, and the principal michael.admin has read-only access to the database (through the administrative commands).

7.4.1.5 Initial Setup of PSSP Kerberos

Fortunately, the setup of PSSP Kerberos is managed transparently by the PSSP installation process. With reference to Chapter 2, "Installing and Configuring a New RS/6000 SP System" of the *PSSP for AIX Installation and Migration Guide*, GA22-7347, the following steps in the installation process are related to PSSP Kerberos:

Step 19: Initialize RS/6000 SP Authentication Services

This step performs most of the setup. It is implemented by the `setup_authent` command, which can be used to set up the control workstation as a Kerberos Version 4 server or client, and does support both PSSP Kerberos and AFS. In the default case that the control workstation is to be set up as the (only) PSSP Kerberos server, `setup_authent` calls its subcommand `setup_ssp_server`, which generates the following:

- The files /etc/krb.conf and /etc/krb.realms
- Entries in /etc/inittab to automatically start the kerberos and kadmind daemons
- The file /.k which holds the Kerberos master key
- The Kerberos database and ACL files in /var/kerberos/database/

- The control workstation's keytab file, /etc/krb-srvtab with entries for the hardmon and rcmd services
- The administration principal's ticket cache file (since `k4init` happens during this setup), normally in /tmp/tkt0 for the root.admin principal
- The Kerberos Version 4 authorization file /.klogin for r-commands running under the root user id

Attention

The /.klogin file: Strictly speaking, the /.klogin file is not a Kerberos file. Kerberos is only responsible for authentication, but the /.klogin file specifies the authorization for Kerberos Version 4 authenticated r-commands (and Sysctl). As such, it belongs to these applications and not to Kerberos itself. Also compare to step 33.

The `setup_authent` command prompts for the Kerberos master password as well as for the names and passwords of administrative users. These are used to create the keys for these (user) principals. Keys for service principals are derived from randomly generated passwords.

A flow chart for the internals of `setup_authent` is in the redbook *RS/6000 SP: Problem Determination Guide*, SG24-4778. Although based on PSSP 2.1, these diagrams should still be accurate (except for the additional logic to support AIX authenticated r-commands).

Step 33: Create Authorization Files

This step has been newly introduced with PSSP 3.1 to support the AIX authenticated r-commands. It controls which authorization files for root user access to the r-commands should be created, by using the `spsetauth` command or the SMIT spauth_rcmd fastpath. The selection is per partition, and stored in the auth_root_rcmd attribute of the Syspar SDR class.

Kerberos Version 4 authorization is always required, and the corresponding /.klogin file has already been created in step 19. Kerberos Version 5 is not supported by the `spsetauth` command. See 7.4.3, "DCE Kerberos (Version 5)" on page 168 for information on how to manually set up the corresponding /.k5login file.

The only choice actually made here is whether a standard /.rhosts file for the root user should be created or not. We recommend that you not have PSSP create the /.rhosts entries for all the SP nodes, since this is much less secure than the Kerberos Version 4-based security. If standard AIX is selected,

PSSP creates (or adds to) the /.rhosts file, with entries for the control workstation and the nodes of all partitions for which standard AIX is selected.

Note that running `spsetauth` on the control workstation only adds or removes the entries in the /.rhosts file which resides on the control workstation. The nodes are reconfigured only when they execute /etc/rc.sp or its subcommand spauthconfig. This is not an issue for the initial installation, but must be kept in mind if `spsetauth` is called later to change the initial settings.

Step 34: Enable Selected Authentication Methods

This step has also been newly introduced with PSSP 3.1, and controls which AIX r-command authentication methods should be used on the SP. Again, this choice can only be made on a per-partition base, through the `chauthpar` command or the SMIT spauth_methods fastpath. It is stored in the auth_methods attribute of the Syspar SDR class.

The `chauthpar` command supports Kerberos Version 5, Kerberos Version 4, and standard AIX. Again, Kerberos Version 4 is required by PSSP. If either of the other two options are selected, they must be in this order. This rule is more restrictive than the underlying AIX command `chauthent`, which only requires standard AIX to be the last method.

Since PSSP 3.1 and AIX use the same set of AIX authenticated r-commands and daemons, the choices made in this step will not only affect the root user or SP system administration tasks, they will be valid for all users. We therefore recommend that you enable all options which should be available to normal users. In particular, specifying neither standard AIX nor Kerberos Version 5 means that the `telnet` and `ftp` commands will not work at all (since they do not support Kerberos Version 4), and the `rcp` and `rsh` commands will only work for authenticated Kerberos Version 4 principals. Of course, specifying Kerberos Version 5 only makes sense if DCE for AIX Version 2.2 is actually used on the SP system.

If the -c flag is specified with the `chauthpar` command, the SDR will be updated with any changes, but only the control workstation will be immediately reconfigured (using the AIX `chauthent` command). If the -f flag is specified, all the nodes in the selected partition will be reconfigured (by running `chauthent` on the nodes through an rsh from the control workstation), whether or not the command actually changes the current SDR settings. If none of these flags are given, nodes will only be reconfigured if the settings in the SDR are changed.

Attention

Fixing Corrupted Settings: Since `chauthpar` relies on a working `rsh` to change the settings on the nodes, it might not succeed if the authentication methods settings are corrupted. There are two ways to fix such a situation: the affected nodes can be rebooted (/etc/rc.sp will reconfigure them to the SDR settings by calling the `spauthconfig` command), or the local root user can run either `spauthconfig` or the AIX `chauthent` command manually on the node.

Step 43: Configure the Control Workstation as Boot/Install Server

When the `setup_server` command prepares a node for network installation, one of the things it does is call the `create_krb_files` command. This creates the Kerberos Version 4 server keytab file for that node, named /tftpboot/<node>-new-srvtab. This file is owned and only readable by the nobody user, since it will be transferred to the node via tftp. The only service principal in the keytab file is the rcmd principal.

Task E. Power On and Install the Nodes

When an SP node is installed, pssp_script (running on the node) adds the required entries to /etc/services and pulls the Kerberos Version 4 files from the control workstation, using tftp. These three files are:

- /etc/krb.conf, copied from cws:/etc/krb.conf, LED code c67
- /etc/krb.realms, copied from cws:/etc/krb.realms, LED code c68
- /etc/krb-srvtab, copied from /tftpboot/<node>-new-srvtab, LED code c69

After this, Kerberos services can be used on the node. For example, the firstboot.cust script might use the kerberized `rcmdtgt` and `rcp` commands to copy more files from the control workstation to the node.

7.4.1.6 Maintaining PSSP Kerberos

In general, as soon as Kerberos is up and running there is not much to do to keep it healthy. If Kerberos-related problems are experienced, a good source of information is Chapter 3, "Kerberos" in the redbook *RS/6000 SP: Problem Determination Guide*, SG24-4778. This book is based on PSSP 2.1, but very few details have changed since then.

It might be helpful to back up the Kerberos files on the control workstation separately from a full system backup, and it might also be useful to add some more administrative principals to the Kerberos database to delegate system management responsibilities. The *PSSP for AIX Administration Guide*,

SA22-7348, contains detailed descriptions of all the administrative commands.

The only command every system administrator will have to use regularly is the `k4init` command, used to re-authenticate to the Kerberos server when the Ticket-Granting Ticket has expired.

7.4.2 AFS Kerberos (Version 4)

PSSP supports the use of an existing AFS server to provide Kerberos Version 4 services to the SP. It does not include the AFS server itself.

Before installing PSSP on the control workstation, an AFS server must be configured and accessible. The `setup_authent` script, which initializes the SP's authentication environment, supports AFS as the underlying Kerberos server. This is mainly contained in its `setup_afs_server` sub-command.

The *PSSP for AIX Installation and Migration Guide*, GA22-7347 explains the steps required to initially set up SP security using an AFS server, and the *PSSP for AIX Administration Guide*, SA22-7348, describes the differences in the management commands of PSSP Kerberos and AFS Kerberos.

7.4.3 DCE Kerberos (Version 5)

The use of Kerberos Version 5, as provided by DCE for AIX Version 2.2, is optional on the SP. PSSP 3.1 does not support Kerberos Version 5 to authenticate its own kerberized applications (the hardware control subsystem and Sysctl). It also does not support the automatic configuration of DCE clients services during network installation of SP nodes. This has to be done separately from the PSSP installation process.

However, PSSP 3.1 does support the AIX authenticated r-commands, which can work with Kerberos Version 5. If you want to use Kerberos Version 5 for the AIX authenticated r-commands, you first have to set up a DCE security and CDS server, either on an external DCE server machine or on the control workstation. It would be sensible to also set up the DCE Distributed Time Service, DTS, as your time management system when using DCE.

Be aware that the DCE servers must be running DCE for AIX Version 2.2. We recommend that you apply at least DCE22 PTF Set 3, since earlier levels had some Kerberos Version 5 related problems in the security service. The following DCE/Kerberos interoperability enhancements were provided with DCE for AIX 2.2, and are described in *DCE for AIX Administration Guide: Core Components*, which is available in softcopy only with the product:

Ex.1010 / Page 192 of 572

- KDC interoperability: The DCE Security Server can be used as a Kerberos Key Distribution Center (KDC) for Kerberos V5 clients.
- Credential cache and keytab file compatibility: DCE and Kerberos V5 applications can share credential cache and keytab files.
- Support of Kerberos flags with `dce_login` and `dceunixd`: The DCE `dce_login` and `dceunixd` commands can be used to obtain credentials with the Kerberos renewable, forwardable, postdated, or proxiable flags.
- `k5dcelogin` command and API: An application can use the `k5dcelogin` command or API to promote Kerberos V5 credentials to DCE credentials. By doing this, the credentials can be used to access DCE objects, such as Distributed File System (DFS) files.
- Configuration and administration support: DCE provides tools that can be used to configure and update the Kerberos V5 /etc/krb5.conf file, as well as accounts used by the secure AIX V4.3 remote services commands.

Install the DCE client components on the control workstation and nodes. This is not managed by PSSP 3.1. If the configuration is done by the `config.dce` command of DCE for AIX Version 2.2, this step will automatically create:

- The /etc/krb5.conf file, which describes the Kerberos Version 5 realm.
- The DCE principal /.:/ftp/<ip_hostname>, which is the service principal for the AIX authenticated `ftp` command.
- The DCE principal /.:/host/<ip_hostname>, which is the service principal for the AIX authenticated r-commands, notably `rcp` and `rsh`. Be aware that this service principal is different from the machine principal /.:/hosts/<ip_host_name>/self.
- The keytab entries for these ftp and host service principals.

Attention

DCE migration: These steps are not automatically performed when a DCE 2.1 client is migrated to DCE 2.2. The command `kerberos.dce` can be used to set up the Kerberos environment.

You can verify that the service principals have been created by issuing `/usr/bin/dcecp -c principal catalog -simplename` and, for example, piping the output to `grep -E "^host\/|^ftp\/"`.

Manually create the root user's Kerberos Version 5 authorization file, /.k5login. This task corresponds to "Step 33: Create Authorization Files" on

page 165, which only handles PSSP authorization (which is a no-op, as /.klogin has already been created by `setup_authent`) and AIX authorization.

The /.k5login file should contain the DCE/Kerberos Version 5 principals that are allowed to access the root account through the AIX authenticated r-comands. The entries must be in Kerberos Version 5 format, not in DCE format. Typically, this file should include the machine principals of the control workstation and nodes; these are the principals which the local root user on these machines will be identified with. You can also add user principals of authorized system administrators. For example:

```
# cat /.k5login
mario@itso.ibm.com
michael@itso.ibm.com
mrbo@itso.ibm.com

hosts/sp4en0/self@itso.ibm.com

hosts/sp4n01/self@itso.ibm.com
hosts/sp4n05/self@itso.ibm.com
hosts/sp4n06/self@itso.ibm.com
hosts/sp4n07/self@itso.ibm.com
hosts/sp4n08/self@itso.ibm.com
```

This file has to be transferred to all the nodes. Since PSSP Kerberos is always required, you can use the Kerberos Version 4 authenticated `rcp` command to copy this file to the nodes.

After that, Kerberos Version 5 must be enabled as a valid authentication method using the `chauthpar` command. Refer to the comments on “Step 34: Enable Selected Authentication Methods” on page 166 for details. More information on actually using the AIX authenticated r-commands can be found in 7.5.2, “Remote Execution Commands” on page 173. This section also discusses Kerberos Version 5 authentication.

7.5 SP Services Which Utilize Kerberos

On the SP, there are three different sets of services which use Kerberos authentication: the hardware control subsystem, the remote execution commands, and the sysctl facility. This section describes the authentication of these services, and the different means they use to authorize clients that have been successfully authenticated.

Ex.1010 / Page 194 of 572

7.5.1 Hardware Control Subsystem

The SP hardware control subsystem is implemented through the hardmon and splogd daemons, which run on the control workstation and interface with the SP hardware through the serial lines. To secure access to the hardware, Kerberos authentication is used and authorization is controlled through hardmon-specific Access Control Lists (ACLs). PSSP 3.1 and earlier releases only support Kerberos Version 4, not Version 5 authentication.

The following commands are the primary clients to the hardware control subsystem:

- `hmmon` — Monitors the hardware state.
- `hmcmds` — Changes the hardware state.
- `s1term` — Provides access to the node's console.
- nodecond — For network booting, uses `hmmon`, `hmcmds` and `s1term`.
- `spmon` — For hardware monitoring and control. Some parameters are used to monitor, and others are used to change the hardware state. The `spmon -open` command opens a s1term connection.

Other commands, like `sphardware` from the SP Perspectives, communicate directly with an internal hardmon API, which is also Kerberos Version 4 authenticated.

To Kerberos, the hardware control subsystem is a service, represented by the principal name hardmon. PSSP sets up one instance of that principal for each network interface of the control workstation, including IP aliases in case of multiple partitions. The secret keys of these hardmon principals are stored in the /etc/krb-srvtab file of the control workstation. The `k4list -srvtab` command shows the existence of these service keys:

```
# k4list -srvtab
Server key file:   /etc/krb-srvtab
Service         Instance        Realm      Key Version
-----------------------------------------------------
hardmon         sp4cw0          MSC.ITSO.IBM.COM 1
rcmd            sp4cw0          MSC.ITSO.IBM.COM 1
hardmon         sp4en0          MSC.ITSO.IBM.COM 1
rcmd            sp4en0          MSC.ITSO.IBM.COM 1
```

The client commands perform a Kerberos Version 4 authentication: they require that the user who invokes them has signed on to Kerberos by the `k4init` command, and pass the user's Ticket-Granting Ticket to the Kerberos

Ex.1010 / Page 195 of 572

server to acquire a service ticket for the hardmon service. This service ticket is then presented to the hardmon daemon, which decrypts it using its secret key stored in the /etc/krb-srvtab file.

Authorization to use the hardware control subsystem is controlled through entries in the /spdata/sys1/spmon/hmacls file, which is read by hardmon when it starts up. Since hardmon runs only on the control workstation, this authorization file also only exists on the control workstation. An example follows:

```
> cat /spdata/sys1/spmon/hmacls
sp4en0 root.admin a
sp4en0 hardmon.sp4en0 a
1 root.admin vsm
1 hardmon.sp4en0 vsm
2 root.admin vsm
2 hardmon.sp4en0 vsm
```

Each line in that file lists an object, a Kerberos principal, and the associated permissions. Objects can either be host names or frame numbers. By default, PSSP creates entries for the control workstation and for each frame in the system, and the only principals which are authorized are root.admin and the instance of hardmon for the SP Ethernet adapter. There are four different sets of permissions, each indicated by a single lowercase letter:

- m (Monitor) - monitor hardware status
- v (Virtual Front Operator Panel) - control/change hardware status
- s (S1) - access to node's console via the serial port (s1term)
- a (Administrative) - use hardmon administrative commands

Note that for the control workstation, only administrative rights are granted. For frames, the monitor, control, and S1 rights are granted. These default entries should never be changed. However, other principals might be added. For example, a site might want to grant operating personnel access to the monitoring facilities without giving them the ability to change the state of the hardware, or access the nodes' console.

Ex.1010 / Page 196 of 572

Attention

Refreshing hardmon: When the hmacls file is changed, the `hmadm setacls` command must be issued on the control workstation to notify the hardmon daemon of the change, and cause it to reread that file. The principal which issues the `hmadm` command must have administrative rights in the original hmacls file, otherwise the refresh will not take effect. However, hardmon can always be completely stopped and restarted by the root user. This will reread the hmacls file.

Care must been taken if any of the hardware monitoring/control commands are issued by users authenticated to Kerberos, but without the required hardmon authorization. In some cases an error message will be returned, for example

```
hmmon: 0026-614 You do not have authorization to access the Hardware Monitor.
```

In other cases, no messages or misleading error messages may be returned. This mostly happens when the principal is listed in the hmacls file but not with the authorization required by the command.

In addition to these commands, which are normally invoked by the system administrator, two SP daemons are also hardmon clients: the splogd daemon and the hmrmd daemon. These daemons use two separate ticket cache files: /tmp/tkt_splogd and /tmp/tkt_hmrmd. Both contain tickets for the hardmon principal, which can be used to communicate with the hardmon daemon without the need to type in passwords.

7.5.2 Remote Execution Commands

In releases prior to PSSP 3.1, PSSP provided its own remote execution commands and the corresponding krshd daemon, which were Kerberos Version 4 authenticated. These were located in /usr/lpp/ssp/rcmd/. All the SP management commands used the PSSP version of `rsh` and `rcp`, and AIX provided the original r-commands in /usr/bin/. This is shown in Figure 78 on page 174.

Ex.1010 / Page 197 of 572

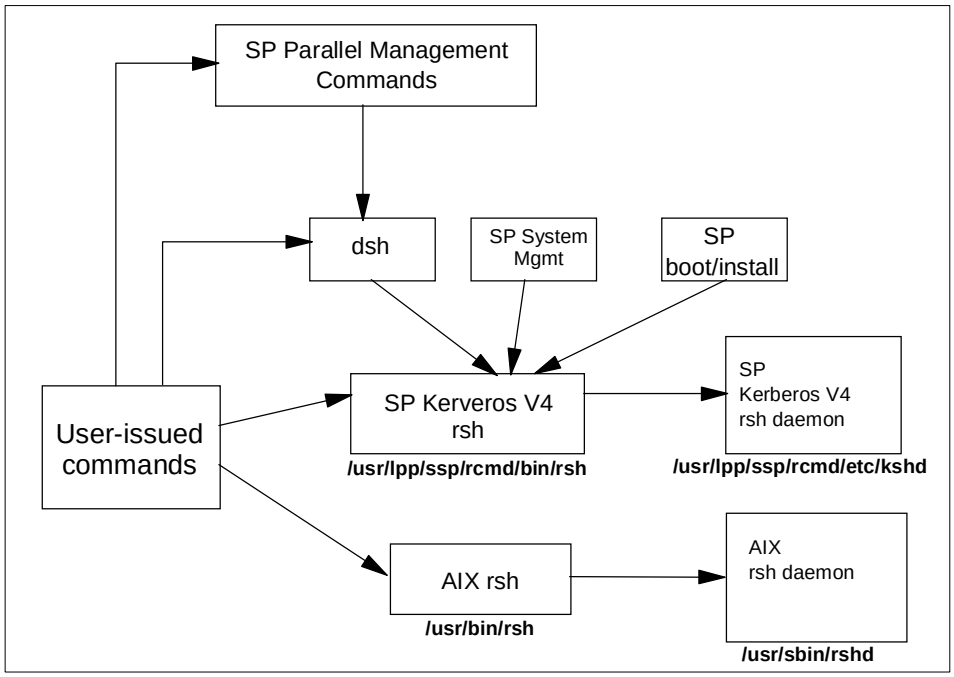


Figure 78. Remote Shell Structure Before PSSP 3.1

In PSSP 3.1, the authenticated r-commands in the base AIX 4.3.2 operating system are used instead. As described in 7.4, “Managing Kerberos on the SP” on page 156, they can be configured for multiple authentication methods, including the PSSP implementation of Kerberos Version 4. To allow applications which use the full PSSP paths to work properly, the PSSP commands `rcp` and `remsh`/`rsh` have not been simply removed, but have been replaced by links to the corresponding AIX commands. This new calling structure is shown in Figure 79 on page 175.

Ex.1010 / Page 198 of 572

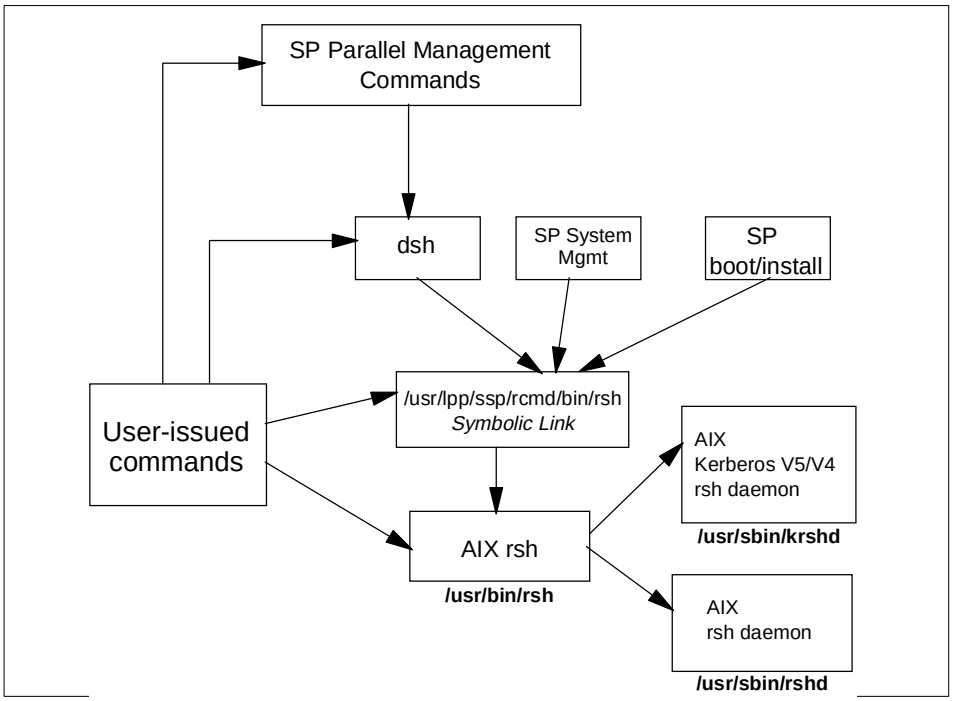


Figure 79. Remote Shell Structure in PSSP 3.1

For the user, this switchover from PSSP-provided authenticated r-commands to the AIX-provided authenticated r-commands should be transparent. In the remainder of this section, we look at some of the implementation details of the security integration of the r-commands, focussing on the AIX `rsh` command and the corresponding rshd and krshd daemons.

7.5.2.1 Control Flow in the rsh Command

The full syntax of the AIX authenticated `rsh` command is

```
/usr/bin/rsh RemoteHost [-n] [-l RemoteUser] \
             [-f|-F] [-k Realm] [Command]
```

Here we assume that a command is present. When the `rsh` command is called, it issues the get_auth_method() system call, which returns the list of authentication methods which are enabled on the machine. It then attempts a remote shell connection using these methods, in the order they are returned, until one of the methods succeeds or all have failed.

Ex.1010 / Page 199 of 572

Attention

K5MUTE: Authentication methods are set on a system level, not on a user level. This means that, for example, on an SP where Kerberos Version 4 and Standard AIX is set, a user's `rsh` command will produce a Kerberos authentication failure if that user has no Kerberos credentials (which is normally the case unless the user is an SP system administrator). After that failure, the `rsh` attempts to use the standard AIX methods. The delay caused by attempting both methods can not be prevented, but there is a means to suppress the error messages of failed authentication requests, which may confuse users. Suppress these messages by setting the environment variable K5MUTE=1. Authorization failures will still be reported, though.

The following describes each of the three authentication methods:

STD When the standard AIX authentication is to be used, `rsh` uses the rcmd() system call from the standard C library (libc.a). The shell port (normally 514/tcp) is used to establish a connection to the /usr/sbin/rshd daemon on the remote host. The name of the local user, the name of the remote user, and the command to be executed are sent. This is the normal BSD-style behavior.

K5 For Kerberos Version 5 authentication, the kcmd() system call is issued (this call is not provided in any library). It acquires a service ticket for the /.:/host/<ip_hostname> service principal from the Kerberos Version 5 server, over the kerberos port (normally 88). It then uses the kshell port (normally 544/tcp) to establish a connection to the /usr/sbin/krshd daemon on the remote host. In addition to the information for STD authentication, kcmd() sends the Kerberos Version 5 service ticket for the rcmd service on the remote host for authentication. If the -f or -F flag of `rsh` is present, it also *forwards* the Ticket-Granting Ticket of the principal which invoked `rsh` to the krshd daemon. Note that Ticket-forwarding is possible with Kerberos Version 5, but not with Version 4.

K4 Kerberos Version 4 authentication is provided by the PSSP software. The system call spk4rsh(), contained in libspk4rcmd.a in the ssp.client fileset, is invoked by the AIX `rsh` command. It acquires a service ticket for the rcmd.<ip_hostname> service principal from the Kerberos Version 4 server, over the kerberos4 port 750. Like kcmd(), the spk4rsh() subroutine uses the kshell port (normally 544/tcp) to connect to the /usr/sbin/krshd daemon

Ex.1010 / Page 200 of 572

on the remote host. It sends the STD information and the Kerberos Version 4 rcmd service ticket, but ignores the -f and -F flags since Version 4 Ticket-Granting Tickets are not forwardable.

These requests are then processed by the rshd and krshd daemons.

Attention

Enabling DCE credential forwarding: DCE credentials can only be forwarded by `rsh -f|-F` if they are obtained by `dce_login -f` or through `dceunix -t`.

7.5.2.2 The Standard rshd Daemon

The /usr/sbin/rshd daemon listening on the shell port (normally 514/tcp) of the target machine implements the standard, BSD-style `rsh` service. Details are found in "rshd Daemon" in *AIX Version 4.3 Commands Reference*, SC23-4119. Notably, the rshd daemon:

- Does some health checks like verifying that the request comes from a well-known port
- Verifies that the local user name (remote user name from the client's view) exists in the user database, gets its UID, home directory, and login shell
- Performs a chdir() to the user's home directory (terminates if this fails)
- If the UID is not zero, checks if the client host is listed in /etc/hosts.equiv
- If the previous check is negative, checks if the client host is listed in $HOME/.rhosts
- If either of these checks succeeds, it executes the command under the user's login shell

Be aware that the daemon itself does not call the get_auth_method() subroutine to check if STD is among the authentication methods. The `chauthent` command simply removes the shell service from the /etc/inetd.conf file when it is called without the `-std` option, so inetd will refuse connections on the shell port. But if the shell service is enabled again by editing /etc/inetd.conf and refreshing inetd, the rshd daemon will honor requests, even though `lsauthent` still reports that Standard AIX authentication is disabled.

7.5.2.3 The Kerberized krshd Daemon

The /usr/sbin/krshd daemon implements the kerberized remote shell service of AIX. It listens on the kshell port (normally 544/tcp), and processes the requests from both the kcmd() and spk4rsh() client calls.

In contrast to rshd, the krshd daemon actually uses get_auth_methods() to check if Kerberos Version 4 or 5 is a valid authentication method. For example, if a request with a Kerberos Version 4 service ticket is received but this authentication method is not configured, the daemon replies with:

```
krshd: Kerberos 4 Authentication Failed: This server is not configured
to support Kerberos 4.
```

After checking if the requested method is valid, the krshd daemon then processes the request. This depends on the protocol version, of course.

Handling Kerberos Version 5 Requests

To authenticate the user, krshd uses the Kerberos Version 5 secret key of the host/<ip_hostname> service and attempts to decrypt the service ticket sent by the client. If this succeeds, the client has authenticated itself.

The daemon then calls the kvalid_user() subroutine, from libvaliduser.a, with the local user name (remote user name from the client's view) and the principal's name. The kvalid_user() subroutine checks if the principal is authorized to access the local AIX user's account. Access is granted if one of the following conditions is true:

1. The $HOME/.k5login file exists, and lists the principal (in Kerberos form). See 7.4.3, "DCE Kerberos (Version 5)" on page 168 for a sample $HOME/.klogin5 file.
2. The $HOME/.k5login file does not exist, and the principal name is the same as the local AIX user's name.

Case (1) is what is expected. But be aware that case (2) is quite contra-intuitive: it means that if the file does exist and is empty, access is denied but if it does not exist access is granted! This is completely reverse to the behavior of both the AIX $HOME/.rhosts file and the Kerberos Version 4 $HOME/.klogin file. However, it is documented to behave this way (and actually follows these rules) in both the kvalid_user() man page and *AIX Version 4.3 System User's Guide: Communications and Networks*, SC23-4127.

If the authorization check is passed, the krshd daemon checks if a Kerberos Version 5 TGT has been forwarded. If this is the case, it calls the `k5dcelogin` command which upgrades the Kerberos TGT to full DCE credentials and

executes the command in that context. If this `k5dcelogin` cannot be done because no TGT was forwarded, the user's login shell is used to execute the command without full DCE credentials.

Attention

DFS home directories: This design may cause trouble if the user's home directory is located in DFS. Since the kvalid_user() subroutine is called by krshd before establishing a full DCE context via k5dcelogin, kvalid_user() does not have user credentials. It runs with the machine credentials of the local host, and so can only access the user's files if they are open to the "other" group of users. The files do not need to be open for the "any_other" group (and this would not help, either), since the daemon always runs as root and so has the hosts/<ip_hostname>/self credentials of the machine.

Handling Kerberos Version 4 Requests

To authenticate the user, krshd uses the Kerberos Version 4 secret key of the rcmd.<ip_hostname> service and attempts to decrypt the service ticket sent by the client. If this succeeds, the client has authenticated itself.

The daemon then checks the Kerberos Version 4 $HOME/.klogin file, and grants access if the principal is listed in it. This is all done by code provided by the PSSP software, which is called by the base AIX krshd daemon. For this reason, Kerberos Version 4 authentication is only available on SP systems, not on normal RS/6000 machines.

Attention

rcmdtgt: PSPP 3.1 still includes the `/usr/lpp/ssp/rcmd/bin/rcmdtgt` command, which can be used by the root user to obtain a ticket-granting ticket by means of the secret key of the rcmd.<localhost> principal stored in /etc/krb-srvtab.

7.5.2.4 NIM and Remote Shell

There is one important exception to keep in mind with respect to the security integration of the `rsh` command. When using boot/install servers, the Network Installation Manager (NIM) will use a remote shell connection from the boot/install server to the control workstation to update status information about the installation process which is stored on the control workstation. This connection is made by using the rcmd() system call rather than the authenticated `rsh` command. The rcmd() system call always uses standard AIX authentication and authorization.

To work around this problem, PSSP uses the authenticated `rsh` command to temporarily add the boot/install server's root user to the /.rhosts file of the control workstation, and removes this entry after network installation.

7.5.3 Sysctl

Sysctl is an authenticated client/server application that runs commands with root privileges on remote nodes, potentially in parallel. It is implemented by the sysctld server daemon running with root privileges on the control workstation and all the nodes, and a `sysctl` client command. The functionality provided by `sysctl` is similar to the `dsh` parallel management command in the sense that it provides remote, parallel command execution for authenticated users. It differs from `dsh` in three fundamental ways:

1. Authentication: Sysctl requires Kerberos Version 4 for authenticating users that issue `sysctl` commands. It does not support Kerberos Version 5, and it is not based on the AIX authenticated r-commands (like `dsh` is).
2. Authorization can be controlled in a more fine-grained way. With `dsh` and the underlying authenticated `rsh`, an authenticated principal which is listed in the authorization file of the remote node (.k5login, .klogin, or .rhosts) can run arbitrary commands since it has access to a login shell. With `sysctl`, authorization can be set for individual commands, by using sysctl-specific Access Control Lists (ACLs) which are checked by an authorization callback mechanism in the sysctld server. Command execution is under control of the sysctld daemon.
3. Language: `dsh` provides access to a shell, whereas sysctl uses its own scripting language, based on Tcl and a number of built-in commands provided by IBM. This might ease the development of site-specific sysctl applications, but also requires you to learn the Tcl/Sysctl language.

The set of commands that are understood by the sysctld daemon is specified through the sysctl configuration file, by default /etc/sysctl.conf, which is parsed by the daemon when it starts up. Note that this file can access other configuration information through include statements. The default configuration file contains include statements for configuration files from `/usr/lpp/ssp/sysctl/bin/`, which contain more sysctl procedures. The sysctl facilities are described in detail in Chapter 13, "Sysctl" of *IBM PSSP for AIX Administration Guide*, SA22-7348. Here we focus on the security aspects of the sysctl system.

To Kerberos, the sysctld server is a service, represented by the same principal name rcmd as the authenticated r-commands which have been described in 7.5.2, "Remote Execution Commands" on page 173. On each machine which runs a sysctld daemon (by default the control workstation and

Ex.1010 / Page 204 of 572

all nodes), there must be a secret key for the principal rcmd.<ip_hostname>, stored in the machine's /etc/krb-srvtab file. This is automatically set up by PSSP, as described in 7.4, "Managing Kerberos on the SP" on page 156.

The `sysctl` client commands perform a Kerberos Version 4 authentication. This requires that the user who invokes `sysctl` has signed on to Kerberos by the `k4init` command. It passes the user's Ticket-Granting Ticket to the Kerberos server to acquire a service ticket for the rcmd service, for each machine on which the sysctl application will be run. These service tickets are then presented to the sysctld daemons on each target machine, which decrypt them using the secret key of the rcmd principal stored in each machine's /etc/krb-srvtab file. At that point, the authentication phase is complete.

For authorization, sysctl uses *authorization callbacks*. Each command in sysctl has an accompanying authorization callback, which specifies the way in which authorization for that command is controlled. An authorization callback is a required part of any sysctl procedure definition. These callbacks are Tcl scripts, which are called by the sysctld daemon immediately before the actual command. If the authorization callback returns without raising an error, sysctld will execute the command. If it returns an error, sysctl will not execute the command and will issue an error message indicating that there is insufficient authorization.

The sysctl system provides four built-in authorization callbacks:

NONE No authorization check is done. Commands with the NONE callback will always be executed, whether or not the invoking user has valid Kerberos credentials.

SYSTEM This callback will always return an error. This implies that commands with the SYSTEM callback will never be authorized to run from the command line (or the sysctl prompt). Such commands can only run when authentication is bypassed. This is the case when sysctld reads its configuration files during startup, within a procedure which is an authorization callback, and within a procedure body whose execution has already been authorized.

AUTH This callback succeeds only if the client has valid Kerberos credentials. This authentication is validated by the successful decryption of the client's service ticket for the rcmd service, as described previously. The user (principal) does not need any specific authorization.

ACL The ACL callback is the most important callback for authorizing user commands. It requires that the user is authenticated to

Kerberos as the AUTH callback does, but in addition requires that this Kerberos principal is explicitly listed in an Access Control List.

Attention

Although *PSSP for AIX Administration Guide*, SA22-7348 lists a couple of sysctl commands which have NONE callbacks, none of those commands actually works without having Kerberos credentials. Unfortunately, this is even the case for the `sysctl help` command.

This behavior is probably caused by the fact that sysctl client always issues the `svcconnect` command to the server before sending the actual command. Since `svcconnect` has the AUTH callback, it will fail if the user does not have Kerberos credentials, and terminate the connection.

The AUTH callback returns OK if the sysctld server has verified the authentication of the client user by decrypting the rcmd service ticket sent by the sysctl client. The ACL callback also requires this, but additionally checks if the Kerberos principal is listed in the server's Access Control List which applies to the command that is to be executed. The default ACL file for all built-in sysctl commands is /etc/sysctl.acl. Sysctl ACL files can contain entries for principals, or include statements for other ACL files, such as the following:

```
> cat /etc/sysctl.acl
#acl#
_PRINCIPAL root.admin@MSC.ITSO.IBM.COM
_PRINCIPAL operator.admin@MSC.ITSO.IBM.COM
_ACL_FILE /etc/sysctl.sp-admins.acl
```

This example lists the default root.admin principal as well as an operator principal, and then includes a separate ACL file which may contain additional principals. The #acl# in the first line must not be changed, since this indicates that the file is a sysctl ACL file. Also note that the underscore character is part of the keywords.

The authorization checks of the sysctld daemons are performed locally on each node. This means that the ACL files must also be copied to all nodes.

In the Tcl/Sysctl source of a sysctl procedure, the ACL callback phrase can optionally contain the full path name of an ACL file. If this is done, the ACL callback will use the specified file for authorization checks. This allows a modular design, where different sets of commands can be controlled by

Ex.1010 / Page 206 of 572

different sets of Access Control Lists. If no specific ACL file name is given in the ACL callback phrase, the default /etc/sysctl.acl file is used. Two important examples of separate ACL files for different groups of commands can be found among the standard configuration files in /usr/lpp/ssp/sysctl/bin/:

- The Problem Management commands in pman.cmds reference their own ACL file named /etc/sysctl.pman.acl. Problem Management is described in section 11.4, “Problem Management” on page 326.
- The SP Error Log Management described in Chapter 27, “Managing Error Logs” of *PSSP for AIX Administration Guide*, SA22-7348, uses the sysctl application. Its sysctl commands are contained in logmgt.cmds and use the /etc/sysctl.logmgt.acl ACL file.

Another important example is GPFS, which is described in 14.4, “General Parallel File System (GPFS)” on page 409. GPFS uses sysctl internally to run the `mmmkvsd` and `mmremote` commands on remote nodes. During GPFS installation, an include statement for the configuration file /usr/lpp/mmfs/bin/mmcmdsysctl is added to the /etc/sysctl.conf file. This file contains sysctl wrappers for the GPFS commands, and uses its own ACLs in /etc/sysctl.mmcmd.acl, as in the following example:

```
> grep -v ^# /usr/lpp/mmfs/bin/mmcmdsysctl
create proc mmmkvsd args {ACL /etc/sysctl.mmcmd.acl} {
        return [eval exec /usr/lpp/mmfs/bin/mmmkvsd $args]
}
create proc mmremote args {ACL /etc/sysctl.mmcmd.acl} {
        return [eval exec /usr/lpp/mmfs/bin/mmremote $args]
}
```

In summary, sysctl allows for a high degree of authorization control through a modular, expandable configuration. However, it also requires that you keep track of multiple files which hold ACLs for the various components. Unfortunately, there is no tool which lists all the ACL files which are used by the commands known to the sysctld daemon. A single point to control the authorization information would clearly be beneficial.

Chapter 8. RS/6000 Cluster Technology

This chapter aims to concisely explain the key concepts of RS/6000 Cluster Technology (RSCT) formerly known as HAI, illustrated by real-life examples of RSCT in action. There are entire books devoted to the topic, such as:

- *RS/6000 SP High Availability Infrastructure*, SG24-4838
- *RS/6000 SP Monitoring: Keeping it Alive*, SG24-4873
- *RSCT: Event Management Programming Guide and Reference,* SA22-7354
- *RSCT: Group Services Programming Guide and Reference,* SA22-7355

See these references for the details of RSCT implementation and exploitation.

8.1 "In the Beginning ..."

In 1994-95, IBM focussed on high availability on the SP. The inherent design of many nodes simplified the hardware aspect - nodes could back each other up. The challenge was, and still is, in the upper layers of the system, such as the database and application.

IBM's High Availability Cluster Multiprocessing (HACMP) product was available at the time (HACMP is still marketed today). It robustly protected hardware resources of small clusters of RS/6000 machines or SP nodes. HACMP had two major limitations:

1. **Scalability**. In HACMP, every node in the cluster communicated with every other node. This was fine for a cluster of up to 8 nodes, but on an SP with 512 nodes, you would have 512x511 simultaneous conversations. The overhead of the basic node communication would have overwhelmed network and processor resources. This is known as the n^2 problem, because as the number of nodes, n, increases, the number of conversations approaches n^2.
2. **Scope**. HACMP concerned itself fundamentally with network adapters, networks, disks and nodes. Hooks were provided to manipulate the upper layers of software, but HACMP did not have a strong application view of availability.

IBM took HACMP's cluster manager code - the heart of the product - and reworked it to scale. Around this IBM built High Availability Infrastructure (HAI). Internally, HAI was known as Phoenix. In 1998, with the announcement

© Copyright IBM Corp. 1999

of PSSP 3.1, IBM renamed this HAI to RSCT, and it made it available to RS/6000 machines, not necessarily SP nodes. RSCT provides a set of services to monitor and manage the availability of applications.

Although this technology was implemented only on the SP, IBM's offering includes clusters of RS/6000 machines through HACMP Enhanced Scalability (up to 32 machines connected in a cluster or 128 SP nodes). RSCT is designed to be a cross-platform set of services, potentially implemented on IBM's and even other manufacturer's server platforms.

8.2 RSCT Packaging

This technology made its first appearance on the SP with PSSP 2.2 in 1996. It is a standard part of PSSP, and it was packaged in the ssp.ha file set until PSSP 2.4. With PSSP 3.1 this technology was repackaged to facilitate its portability to non-SP nodes (RS/6000 machines at this time).

The new files sets are called *rsct.basic* and *rsct.clients*. The current version at the time of this writing is version 1.1 (PTF set 4). The RSCT file sets come standard with PSSP 3.1 and HACMP Enhanced Scalability 4.3. Table 4 shows details of the RSCT install images.

Table 4. RSCT Install Images

Fileset	Description
rsct.basic.rte	RSCT basic function (all realms)
rsct.basic.sp	RSCT basic function (SP realm)
rsct.basic.hacmp	RSCT basic function (HACMP realm)
rsct.clients.rte	RSCT client function (all realms)
rsct.clients.sp	RSCT client function (SP realm)
rsct.clients.hacmp	RSCT client function (HACMP realm)

The Resource Monitor component, closely linked with RSCT, partially relies on the Performance Agent Tools component of AIX (perfagent.tools). So, this file set (which comes with AIX 4.3.2) is a prerequisite for RSCT.

In previous versions of PSSP (below PSSP 3.1), this high availability infrastructure (ssp.ha) has a dependency on the Performance Agent Server (perfagent.server) component of the IBM's Performance Toolbox, which comes with the Performance Toolbox Aide (PAIDE) and it is shipped along with PSSP 2.4 or below.

Ex.1010 / Page 210 of 572

This Performance Agent Server component must also be installed on the control workstation running PSSP 3.1, if there are nodes running older levels.

8.3 RSCT Architecture Overview - “The Picture”

No discussion of RSCT is complete without “The Picture”, as shown in Figure 80.

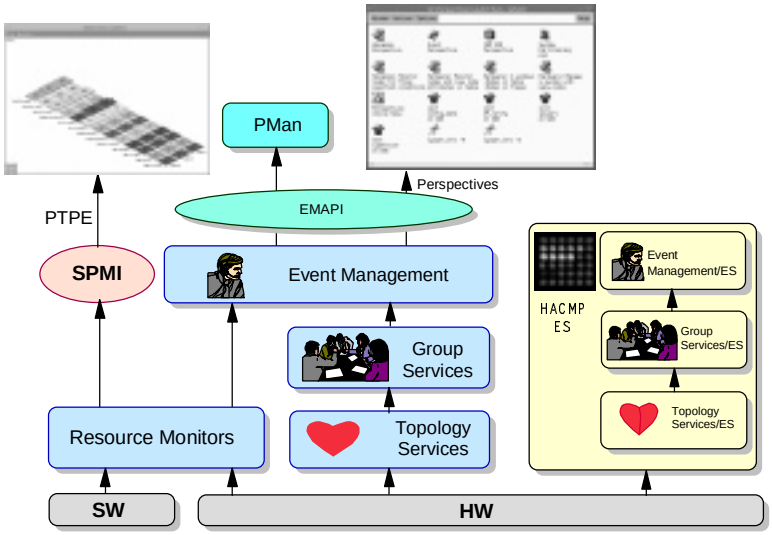


Figure 80. RSCT Infrastructure - “The Picture”

Although many components are presented in the figure (all of which will be explained throughout the book), RSCT comprises three principle elements:

1. Topology Services (TS)
2. Group Services (GS)
3. Event Manager (EM)

Sitting above EM is the Event Manager Application Programming Interface (EMAPI). Through this API, higher level components gain knowledge of the state and availability of system resources. Examples of these components include Problem Management (PMAN), a system monitor such as the Perspectives graphical user interface, and others (potentially databases, enablers and applications).

As you can see from Figure 80 on page 187, there is a separate stack for HACMP/ES. As we mentioned earlier in the chapter, RSCT can function also

Ex.1010 / Page 211 of 572

in a cluster of RS/6000 machines, even in a mix of RS/6000 and SP nodes (up to 32 nodes in an HACMP/ES cluster). The picture shows what you may see in a node that is part of an SP “domain” and an HACMP/ES “domain”. (The concept of domain in RSCT is very similar to system partitions in the SP. Actually they are the same if RSCT is running on an RS/6000 SP.)

The RSCT components will behave slightly different if they are functioning on an SP domain or an HACMP/ES domain. For example, it they are running on an SP domain, the configuration data is taken from the SDR, while if they are running on an HACMP/ES domain, the configuration data is taken from the Global ODM (GODM).

Feeding EM with SP resource information are Resource Monitors (RM) and the System Performance Measurement Interface (SPMI). SPMI is a construct of Performance Agent. SPMI and EM supply data to higher level performance monitoring tools such as Performance Toolbox Parallel Extensions (PTPE).

RM, SPMI, and PTPE are SP-specific implementations of resource and performance monitoring. They are not strictly part of base RSCT. They changed as RSCT moved from platform to platform, whereas TS, GS, and EM are designed to be platform-independent services.

We now discuss the core elements of RSCT: TS, GS, and EM.

8.4 Topology Services (TS)

Topology Services (TS) is the foundation of RSCT. A distributed subsystem, TS is implemented by the hatsd daemon on each node and the CWS. TS lives to maintain availability information about nodes and their network adapters. TS considers a node to be “up” if it can be reached through at least one communication path. Determining valid communication paths is why TS concerns itself with network adapters.

Currently, TS supports only the SP administrative Ethernet and switch adapters if running on SP domains. (On HACMP, it supports a larger number of networks. See Chapter 15, “High Availability” on page 433 for details.) TS monitors all SP node types and the CWS.

TS works tirelessly to update node and network adapter states, but does not care about the implications of, for example, a node going down. However, Group Services (GS) does care about this. GS subscribes to TS for node and network adapter availability. To take action on this information, GS needs a reliable network roadmap to broadcast its messages to the nodes. This leads us to TS’s other primary responsibility: maintaining the network roadmap or

Network Connectivity Table (NCT) for use by GS's Reliable Messaging component. GS accesses the NCT via a shared memory segment. This is illustrated in Figure 81

The hatsd daemons communicate via UDP. As UDP is an unreliable communication protocol, the daemons must validate and acknowledge the UDP packets. TS and GS communicate via UNIX Domain Stream (UDS) sockets.

Before describing how TS performs its duties, a few terms are explained:

Adapter Membership is the process of monitoring the availability of network adapters in the environment and establishing routes between nodes.

Adapter Membership Group is an association of adapters, governed by monitoring protocols. A Singleton group is an Adapter Membership Group with exactly one member.

Node Membership is the process of maintaining node availability based on Adapter Membership. If adapters on different nodes are in the same Adapter Membership Group, then the nodes can communicate. Nodes can also communicate indirectly across different Adapter Membership Groups. This is all captured in TS's NCT.

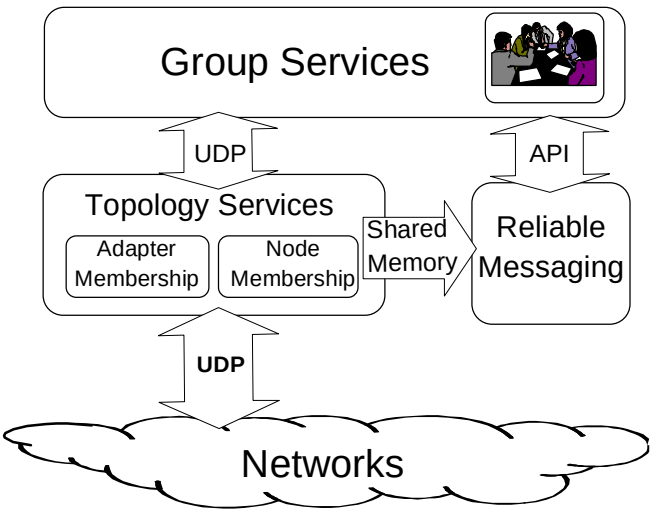


Figure 81. TS and GS Interfaces

Ex.1010 / Page 213 of 572

8.4.1 Group Leaders, Crown Princes, and Death in the Family

You power up an SP. As each node comes up, it starts its hatsd daemon. TS comes alive. How does TS perform Adapter Membership? Since it uses UDP, TS must eventually become aware of the IP addresses of the network adapters on the other nodes. TS goes to the source of SP configuration: the SDR.

Each node builds a *machine list* after receiving information from the SDR. The machine list defines all nodes in the system, and the switch and administrative Ethernet IP addresses for each node. Note that the administrative Ethernet interface is typically en0, but could be en1. You specify which interface when defining the reliable hostname in the SDR at installation time, and the corresponding IP address is sent in the machine list. When new nodes are added to a running SP system, the SDR does not automatically send the configuration changes to the nodes. The RSCT stack must be refreshed for all affected partitions to update the machine lists and membership groups. TS could easily be ported to different platforms by essentially changing the front-end process, served by the SDR on the SP, that supplies the initial node and adapter information.

Machine lists are partition-specific and always include the CWS. TS has the dependency that each node has to be able to communicate with the CWS, thus the CWS is included in every machine list. The CWS runs one instance of hatsd for each partition.

Before continuing the description of the formation of Adapter Membership Groups, we need to explain a little more terminology. Every group has two special adapters with specific personalities[1]:

- **Group Leader**. This adapter has the highest IP address in the Adapter Membership Group. The Group Leader maintains the topology and connectivity information for the group and distributes it to the members.
- **Crown Prince**. This adapter takes over as Group Leader when the Group Leader disappears. Possible causes of this succession are failure of the Group Leader's node, adapter, or IP communication subsystem. The Crown Prince has the second highest IP address in the group.

There is a third special personality that is present when a network (ring) is made of multiple segments (subnets) and routing is involved. This personality is known as the *Mayor*, and it is defined as follows:

Mayor - A daemon, 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

[1] Personality is a duty that a daemon has to carry out. It is common that a daemon assumes multiple duties.

Ex.1010 / Page 214 of 572

group that are also members of its subnet. The way a daemon knows that it has been picked is when it receives a message that has to be broadcast.

Now each node knows, via its machine lists, all the potential members of Adapter Membership Groups for the Ethernet and switch. Each node first initializes itself into a Singleton group - making it its own Group Leader and Crown Prince. The Group Leaders periodically send proclamations to all lower IP address adapters. The lower IP address Group Leaders eventually join the rings of the higher IP address Group Leaders, which incorporate the joiners' topology. Group Leaders, Crown Princes and member lists are constantly updated in the new, bigger rings. Finally, one ring for each adapter type exists and the final personality of the adapters is determined by IP address order. With the rings established, Adapter Membership Groups are created.

Every five seconds, using a Proclaim Packet, the Group Leader in an Adapter Membership Group invites other adapters that are in the machine list but not currently part of the group, to join the group. This is how adapters in new or rebooted nodes, for example, become part of the group.

In the steady state, each adapter in a group is responsible for passively monitoring its *Neighbor*, which is the adapter with the next highest IP address. By passively monitoring we mean that the Neighbor sends heartbeat packets to the adapter, which simply takes note of receiving them. For example, given Adapter_A with an IP address of x.x.x.10 and Adapter _B with an IP address of x.x.x.9, Adapter_A is the Neighbor of Adapter_B and sends Adapter_B heartbeat packets. Adapter_B does not acknowledge the packets from its Neighbor.

The Group Leader "wraps around": its Neighbor is the adapter with the lowest IP address in a group.

The monitoring process continues until a change to the group topology occurs, such as an adapter failure. The n^2 problem is avoided because in steady state, a given adapter is in communication with only one other adapter, independent of the number of nodes installed.

By default, heartbeats are sent every second (the *frequency*), and if four successive heartbeats are not received (the *sensitivity*), the Neighbor adapter is declared unavailable. Heartbeat frequency and sensitivity are tunable. If an adapter declares its neighbor unavailable, the hatsd daemon notifies the Group Leader with a DEATH_IN_FAMILY message. The Group Leader updates the membership status of the group and distributes it to the

remaining group members, starting with a Prepare To Commit (to a new topology) message.

8.4.2 Network Connectivity Table - The Network Roadmap

TS creates several layers of information about nodes and adapters, culminating in the Network Connectivity Table; see Figure 82.

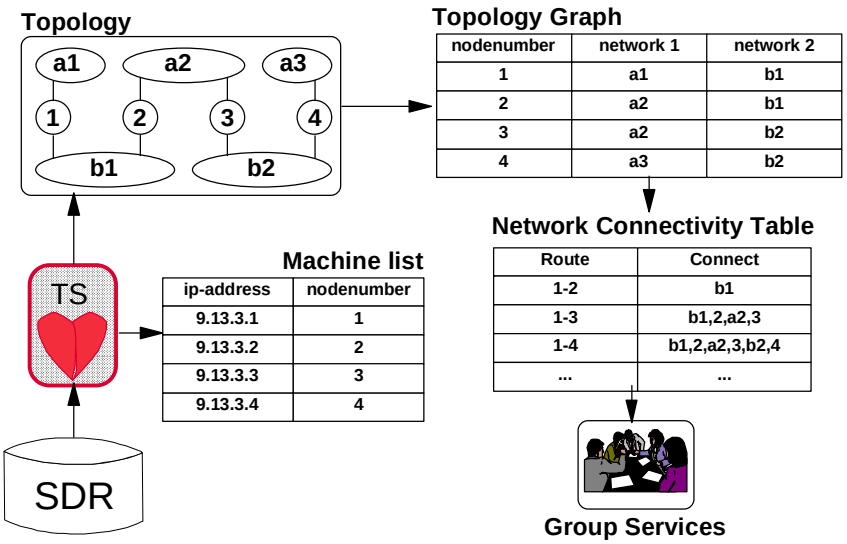


Figure 82. TS Process Flow

With reference to Figure 82, TS goes through the following steps:

- Receive node and adapter configuration information from the SDR at the CWS
- Build machine lists
- Form the adapter rings
- Create Adapter Membership Groups
- Build a topology table and graph to indicate individual node connectivity, therefore availability
- Build the Network Connectivity Table in shared memory specifying all the valid routes between all node combinations
- Accept connections from local clients, primarily GS (up to this point TS would refuse connections because it has not completely stabilized its information)

Ex.1010 / Page 216 of 572

- Monitor adapters via heartbeats and update the tables as necessary

Now that we know what nodes are available and how to talk to them, GS steps into the picture.

8.5 Group Services (GS)

Applications that best exploit the SP architecture typically comprise several cooperating processes running on multiple nodes. How does a distributed application first notice something has gone wrong in the system, then coordinate its own recovery? Group Services (GS) is a general purpose facility for coordinating and monitoring changes to the state of applications running on a set of nodes. The application view of high availability in RSCT begins with GS.

With reference to Figure 83 on page 194, a GS daemon, hagsd, runs on all nodes and the CWS. Like TS, GS is partition-sensitive and multiple hagsd instances can be running at the CWS. Also like TS, GS daemons exchange their own reliable-protocol messages over UDP/IP. The communication routes selected are determined from the NCT created by TS.

As the name implies, groups are an important concept in GS. To take advantage of the coordination and monitoring functions of GS, an application must first form a group. The active members of the group correspond to the processes on each node that support a distributed application. Such group members are called *providers*. Another process may only wish to be informed of the group's activities. Such a process can ask to subscribe to a group, and if accepted, becomes a *subscriber* to the group. Subscribers are not group members. Subscribers just watch. Providers and subscribers are collectively called *GS clients*.

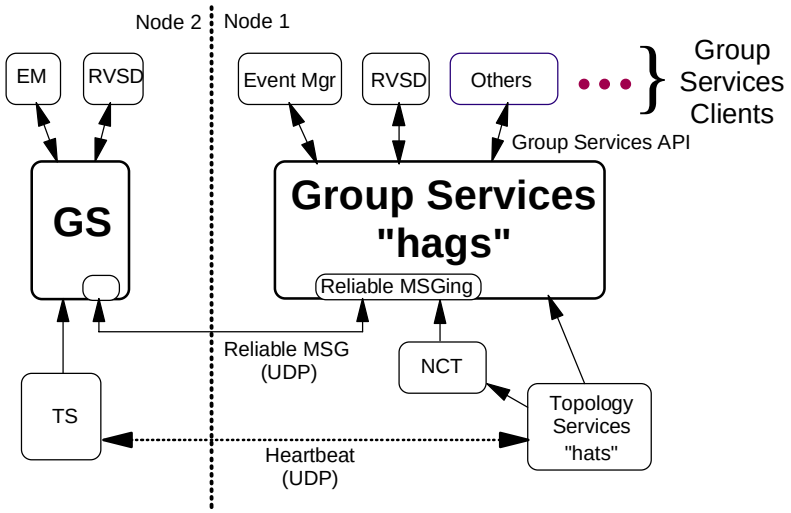


Figure 83. Group Services Structure

Groups are defined by three attributes:

1. Name. This must uniquely identify the group.
2. Membership List. This is a list of one or more providers. The list is maintained in order of age, with the oldest - the first provider to join the group - at the top and the youngest at the bottom. The provider is identified by an identifier, which comprises the instance ID and node number on which the provider is running. Instance IDs are defined by the provider itself when it joins the group. Identifiers must be unique in the group; for example, on a given node two identical processes may be running in support of a distributed application. They must be uniquely identified to participate properly in GS functions.
3. Group State Value. This is a byte field up to 256 bytes long. It is not interpreted by GS. It can be used as a group's billboard for advertising its state. EM uses the state value of one of its groups for version control of the Resource Manager database, to ensure each node in the group is monitoring from the same vintage of resource variables.

By default, the SP maintains three *interna*l groups: host membership, Ethernet adapter membership, and switch membership. GS clients can subscribe to these groups to keep track of hardware status. In addition, GS clients can create *external,* or user, groups.

Ex.1010 / Page 218 of 572

Who can be a GS client? Who can create groups? The answer is: any process of an application or subsystem that uses the Group Services Application Programming Interface (GSAPI). Examples from IBM include EM, RVSD, GPFS, and HACMP ES. If you are writing your own applications or updating an existing one, you could embed GSAPI function calls to:

- Coordinate your peer processes running on other nodes.
- Create a “bulletin board” to advertise your application state to other applications.
- Subscribe to changes in the state of other applications that yours may depend on.

Use of GS is optional. GS itself does not perform recovery actions, but provides the synchronization and commit protocols that application programmers typically find complex, expensive, and error-prone. Some applications have already incorporated their own methods for fault tolerance; GS does not demand that you re-engineer these facilities. For these applications GS could be used only for inter-application coordination.

8.5.1 GS Components - Functional Overview

This section describes the internal components of GS and how GS clients can get services from GS - by forming a group.

With reference to Figure 84 on page 196, GS consists of several components:

- **TS Client Module**. GS receives services from TS here.
- **Reliable Messaging Module**. This module, using the NCT created by TS in shared memory, provides reliable, sequenced delivery of messages between GS daemons.
- **Name Server Module**. This module controls the GS namespace. GS provides a single group namespace per domain, managing all internal and external groups. A domain is the set of nodes within a partition, and only one GS daemon is elected GS nameserver in a domain. After the TS Client Module successfully connects to TS, GS determines the lowest-numbered operational node by sorting node membership data. The GS daemon corresponding to this node becomes the GS nameserver for the domain. Should the GS nameserver daemon fail, the GS daemon on the next lowest-numbered node is elected nameserver. If the original GS daemon comes back on-line, it does not resume its responsibilities, but goes to the end of the succession line.

- **Client Control Module**. This module accepts and manages connections to GS clients. GS clients employ UNIX domain socket connections with the GS daemon.
- **Meta-Group Control Module**. This module handles communications between the GS daemons. A *meta-group* is the collection of GS daemons that support a GS client's group. An example of communications handled by this module is the following: a request comes in from the Client Control Module (interfacing a GS client, such as Application_A) to create a new group called *elliot*. After reviewing the names of the groups it already supports, the Meta-Group Control Module communicates with the Name Server GS daemon to ensure that no other group by that name exists in the domain. The group *elliot* may already exist, created by Application_B on some other group of nodes in the domain.

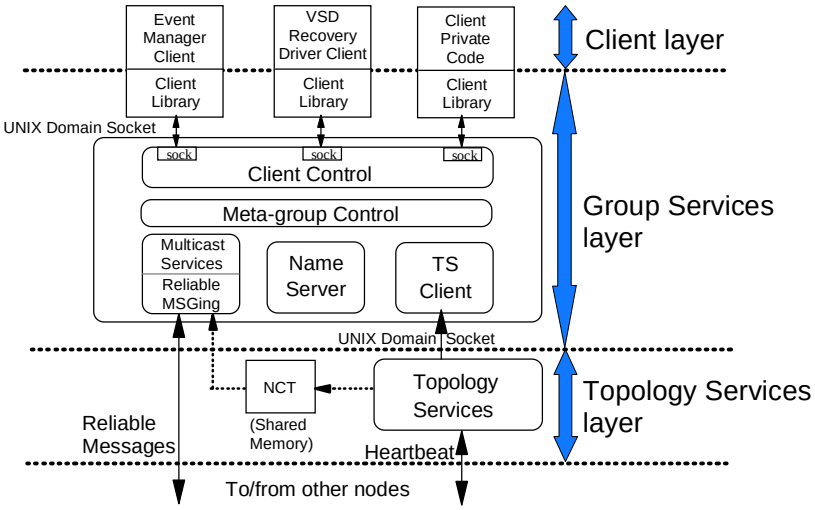


Figure 84. GS Internals

The GS subsystem comprises two daemons:

1. hagsd - provides most of the services of GS.
2. hagsglsmd - a client of hagsd that provides global synchronization services for the SP Switch adapter membership group.

8.5.2 Creating GS Groups

A GS client creates an external group by executing a join request function call in the application program. The first join request creates a group, provided that group name does not already exist in the domain. Subsequent join

Ex.1010 / Page 220 of 572

requests to the group add new providers to the group. Internal groups, those for host and adapter membership, are created automatically via the same process by which GS daemons come alive. In effect, TS is a pseudo-GS client in that membership of the internal groups derives from TS data.

It is important to understand that although GS is globally aware of all groups existing in the system (the GS nameserver must have them all registered, complete with membership lists), not every GS daemon knows about every group. For example, refer to Figure 85 on page 197. In a 9-node SP, a parallel application is distributed across nodes 1, 2, 3, 4, 5 and 6. The processes on those nodes form a group called *glynn.* Similarly, the application *clarabut* is spread across nodes 2, 6, 7, and 8. Both groups *glynn* and *clarabut* are registered to the GS nameserver on node 9. If you query nodes 7 and 8 for the defined external groups, they will report only on *clarabut*. To maintain knowledge of *glynn* and its gyrations in GS daemons independent of *glynn* wastes processor and communications bandwidth on nodes 7 and 8. Similarly, nodes 1, 3, and 5 know nothing of *clarabut*.

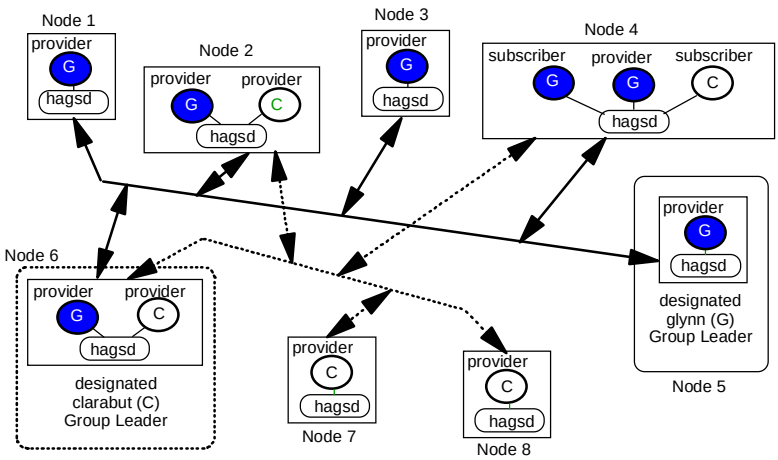


Figure 85. "Group Awareness" of GS Daemons

For each new group, one GS daemon will become the Group Leader. Yes, this is the same term as in TS, but GS Group Leaders are chosen by the age of the provider in the membership list, not by the IP address. The GS daemon

Ex.1010 / Page 221 of 572

on the node with the oldest provider becomes Group Leader. A Group Leader acts as the coordinator for all operations in the group, including:

- Managing group information (membership lists, state value)
- Managing group meta-data (items such as protocols, coordination of voting, default votes, recovery actions)

Group Leaders replicate data to all GS daemons on nodes where the group's providers reside for availability in case the Group Leader node fails.

8.5.3 Democracy Comes to the SP - Voting Protocols

So now we have created groups and are all set to enjoy the rich function of GS. How does a member act in a group in a controlled, synchronized manner? This happens via *protocols*.

Protocols do the work in GS. A *protocol* is a mechanism that coordinates state changes and membership in a group. Consider typical actions that might occur in a group:

- **Membership Changes**. Providers can willfully join or leave the group. They can also leave by failing or being cast out by the other providers.
- **State Value Changes**. A provider may want to update the state value or the group's "billboard" for interpretation by another application; for example, a database wants to let a dependent application know that it is unavailable.
- **Message Broadcasts**. A provider may wish to send a message to all other members.

Protocols are somewhat analogous to a well-mannered democracy. Providers must respect group integrity and follow due process. Protocol communications are handled by the Group Leader. Protocols execute in the following sequence:

1. Proposal
   Any provider, or GS itself, can start or propose an action. Suppose a process wants to join a group. It issues a join request, and GS will submit the request to the existing members of the group.
2. Asking Vote
   Depending on the proposal, the group may have to vote. In our example, there are circumstances when the group does not want new members to join. If a distributed application is trying to recover from a failure, it may not necessarily want new processing nodes, ready to do application work, to come on-line until recovery is complete.

Voting is defined by the number of phases, or rounds of voting, required. The proposer of the protocol asks for n phases, where voting is repeated n-1 times. If a proposal requires 1-phase voting, the proposal is automatically approved. A 2-phase protocol requires one round of voting. The flexible n-phase voting mechanism allows application programmers to tailor their synchronization and coordination requirements.

3. Voting
   If the protocol is 2 phases or higher, the members vote. Vote responses are Approve, Reject, or Continue. Unlike in most democracies, voting must be unanimous for proposal acceptance - one Reject vote causes the proposal to be refused. A Reject ends the protocol, regardless of additional rounds of voting requested. Continue is a conditional Approve, but with the desire to go to another voting round. A single Continue vote causes another voting round.

   If a provider does not vote in the time allowed (for example, the provider's node fails), the Group Leader uses the group's default vote for that provider.

4. Notification of Result
   GS notifies the provider of the result, and updates the membership list and state value accordingly. In our example, the potential provider has been voted into the group, is made a member, and put on the membership list of the other providers.

An illustration of GS protocol execution is part of 8.7, "Putting It All Together - A Simple Example of Exploiting RSCT" on page 209.

GS guarantees orderly and coherent execution of protocols. For a given group, GS may receive many simultaneous protocol requests. GS will execute exactly one at a time (unless it is advantageous to batch up requests, such as multiple joins or failure leaves and cast outs). GS uses *barrier synchronization* in protocols. All providers must proceed to a certain point, or barrier, before the protocol continues. Completion of voting is a barrier: GS will wait until every vote is received, or the voting time limit expires and the default vote is cast for a provider, before taking any other action in the protocol.

8.5.4 Other GS Facilities

This section generally describes two other services of GS: Source-Target Groups, and sundered network recovery. Details on these services may be found in *RS/6000 SP High Availability Infrastructure*, SG24-4838.

Source-Target Groups

GS does not normally support services between groups beyond subscription. The Source-Target facility provides some level of synchronization between groups. Consider two applications: a disk availability subsystem, and a distributed database. Each forms a group in GS. If a node crashes, the database application may wish to begin its recovery protocols only after the disk recovery application uses GS to help make the database disks available again.

Sundered Network Recovery

Given an unfortunate set of network failures in a partition, GS daemons may become split and unable to communicate across the split. The GS namespace becomes *sundered*, although each GS daemon in each sundered portion of the partition has enough information to reconstruct the original group.

Bad things can happen in sundered namespaces. Two nodes, each owning a tail of a twin-tailed disk subsystem, could be on different sides of the split. The disk recovery application, relying on GS for node availability, may mistakenly inform each node in the split group that its availability partner is gone. Each node may try to acquire the disk subsystem, leading potentially to data corruption. A quorum mechanism in the application (GS does not provide such as mechanism) would be useful in this case.

When the network is repaired, GS has protocols to automatically reintegrate the sundered namespace and groups.

8.6 Event Management (EM)

GS provides applications with a rich infrastructure of coordination and synchronization services. In terms of giving the application detailed knowledge of the resources in the environment, about all GS can report is what it knows from TS: whether nodes and network adapters are up or down. How do distributed applications know about CPU utilization on a critical database node, or whether a file system is about to fill up, or if network errors are mounting on a key communications link? From the point of view of availability, any distributed application would be very interested in this type of information.

EM provides an application for comprehensive monitoring of hardware and software resources in the system. A *resource* is simply an entity in the system that provides a set of services. CPUs execute instructions, disks store data, database subsystems enable applications. You define what system events

Ex.1010 / Page 224 of 572

are of interest to your application, register them with EM, and let EM efficiently monitor the system. Should the event occur, EM will notify your application. Figure 86 illustrates EM's functional design.

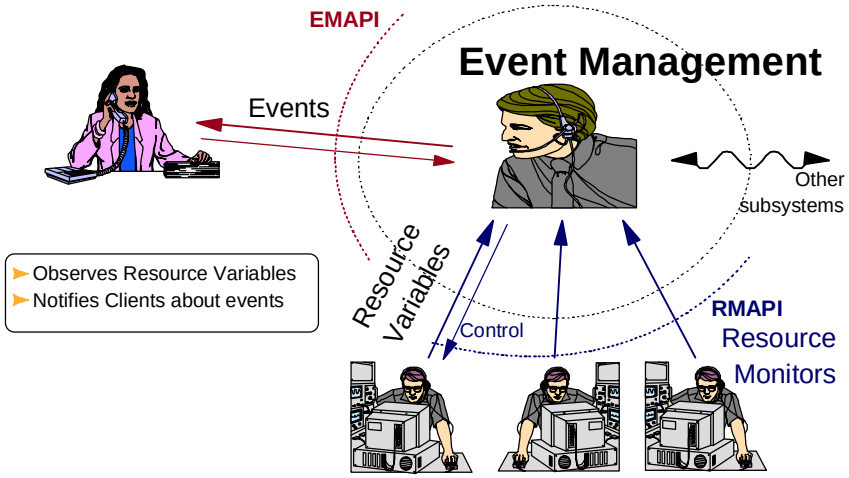


Figure 86. EM Design

EM gathers information on system resources by using *Resource Monitors (RM)*. RMs provide the actual data on system resources to the event-determining algorithms of EM. RMs are integral to EM, therefore RSCT, but how do RM get their data? Data-gathering mechanisms would vary according to platform. The SP-specific implementation of resource data-gathering mechanisms is described in 8.6.4, "Resource Monitors - The SP Implementation" on page 205.

EM is a distributed application, implemented by the EM daemon (haemd) running on each node and the CWS. Like TS and GS, EM is partition-sensitive, thus the CWS may run multiple instances of haemd. To manage its distributed daemons, EM exploits GS. GS lives to serve applications like EM. As EM must communicate reliably among its daemons, it uses the Reliable Messaging information built from TS. This is shown in Figure 87 on page 202.

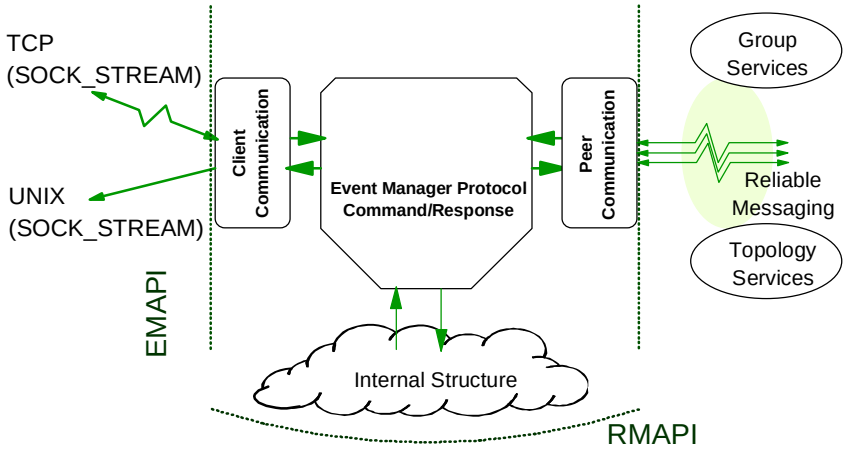


Figure 87. EM Client and Peer Communication

EM receives resource data across the Resource Monitor Application Programming Interface (RMAPI). Clients communicate with EM through the Event Manager Application Programming Interface (EMAPI). An EM client can comprise many processes spread across nodes in a partition. A local process, that is, one executing on the same node as a given EM daemon, uses reliable UNIX domain sockets to talk to EM. On the CWS, a local process connects to the EM daemon that is running in the same system partition as the overall client. In this manner, the client can get events from anywhere in its partition.

Remote clients, that is, clients executing in a separate partition or outside the SP entirely, use TCP/IP sockets, which is a less reliable method because of the protocol that cannot always properly deal with crashed communications sessions between programs. Remote clients usually connect only to the EM daemon on the CWS. When connecting, a remote client specifies the name of the target partition on the call to the EMAPI. The remote client will then connect to the EM daemon on the CWS that is running in the target partition. A client could connect directly to any EM daemon in the target partition and get the same events, but you would need an algorithm to determine the target node. It is easier to just connect to the appropriate daemon on the CWS.

8.6.1 Scalable Design

An EM client can efficiently monitor an event SP-wide by exploiting the distributed EM architecture. This is critical to the scalability of the monitoring and notification infrastructure. A client, executing on node *steve*, registers for

events occurring on node *scott* by communicating only with the EM daemon on node *steve*. Peer communication paths, based on Reliable Messaging, allow the EM daemon on node *scott* to transfer event notifications to node *steve*'s EM daemon, and therefore to the client. If node *steve*'s EM daemon tried to remotely connect to the RM on node *scott*, a great deal of resource data would go across the network for haemd on node *steve* to analyze for events. Why waste system resources when you have a node *scott* haemd there to look at the data and transmit registered events only as required? *RMs are always local to the EM daemon.*

Clients must use the *event registration* process to inform EM of their interest in specific events. Applications perform event registration through function calls to the EMAPI. Perspectives provides a menu-driven interface to register events, and further specifies actions to take when events occur. Event registration defines:

- The *resource variable*, or attribute of a specific resource. A resource variable can be a counter (such as the number of packets transmitted on en0), quantity (such as average CPU busy), or state (such as the network interface up or down). Resource variables must be associated with a RM so that EM knows who to ask for data. This association is made via the *resource class*.
- The *Resource Identifier* of the resource variable. You want to monitor average CPU busy, but on which node? Which occurrence of that resource in the system? Resource identifiers pinpoint specific resources in the system.
- The *expression* or event-triggering rule to apply against the resource variable (such as "average CPU busy is greater than 90%"). Each variable has a default expression.

  You can also specify a *rearm expression* for a resource variable. If set, the rearm expression is monitored after the initial expression is true. When the rearm expression is true, then the initial expression is once again monitored. For example, you are interested in CPU busy on node *sam*. Your expression will trigger an event whenever *sam*'s CPU busy rises above 90%, but if the CPU workload is fluctuating, the resource variable value may repeatedly cross the 90% threshold, generating many identical events. If you specify a rearm expression of CPU busy going below 60%, then after *sam* goes above 90% CPU busy, EM will not generate another event on this resource variable for *sam* until *sam*'s CPU busy falls below 60%. You assume that the situation has stabilized on *sam* if the CPU busy falls back below 60%.

Whenever a resource is added or changed in the SDR, the EM Configuration Database (or EMCDB) must be compiled, time-stamped for version control, and redistributed to the EM daemons. Restarting all EM daemons accomplishes the redistribution.

8.6.2 Your Application and EM

To enable your application to use EM, you follow these steps:

1. Check the availability of a RM that gives you the data you need. If none exists, you would have to write a program using the RMAPI.
2. Check the SDR for your event definition. If it is there, you may activate it, else you will have to go through event registration.
3. Using the EMAPI, register your application as a client and receive events from EM. In addition to event notification, the EMAPI provides two query services for your application:
    1. The current value of a resource variable, whether or not it is triggering an event
    2. A list of defined resource variables and their default expressions

8.6.3 EM and GS

EM initialization illustrates how EM exploits GS. When haemd starts on a node, it fetches the EMCDB version from the SDR. Operating with the correct version of the EMCDB is crucial to service EM clients. After more initialization steps, haemd connects to GS.

All EM daemons are providers of a group called ha_em_peers. This group allows EM to synchronize the joining and leaving of EM daemons, which collectively form EM's distributed execution. When the *first* EM daemon requests to join the ha_em_peers group, the group does not actually exist. The daemon will create the ha_em_peers group and initialize the group state variable with the EMCDB version stored in the SDR.

Consider the case when an ha_em_peers group already exists. With reference to Figure 88 on page 205, EM is running and the ha_em_peers group's state variable contains the current operational version of EMCDB. Once the new EM daemon has successfully joined the ha_em_peers group, it must load the correct version of the EMCDB.

Ex.1010 / Page 228 of 572

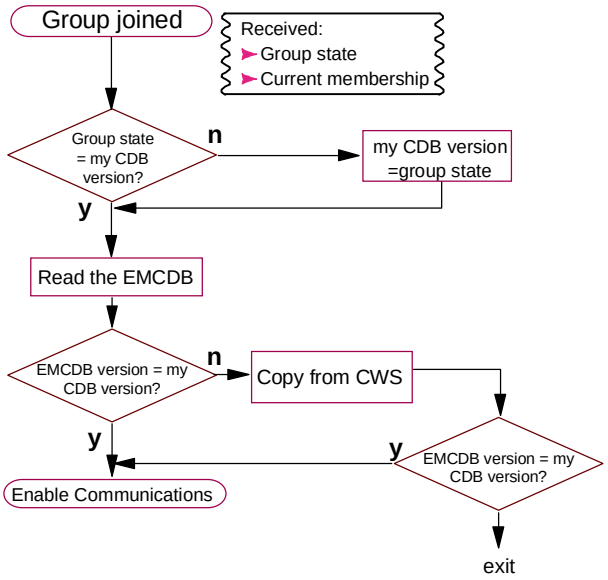


Figure 88. haemd Joining Group Services

From the group state value, the EM daemon knows what version of EMCDB is being used by the already-running EM daemons (peers). It compares the state value with the one it obtained from the SDR earlier in the initialization. If they do not match, the EM daemon uses the state variable value to remain consistent with the group. Now the EM daemon attempts to load the database from a local copy. It examines the EMCDB in the appropriate file to verify consistency with what the group is currently using. If the versions match, EM daemon loads the database and is ready to accept clients. If not, the EM daemon checks one more place: an EMCDB staging area on the CWS. If still no match, then the EM daemon is not configured and refuses any requests for event monitoring.

8.6.4 Resource Monitors - The SP Implementation

Figure 89 on page 206 summarizes the key elements of resource monitoring for EM. As you can see, EM is one busy little daemon. EM uses a number of RM, both internal (inside the EM daemon) and external (below the RMAPI line) to supply the data of the resource variables. The RMs are discussed in more detail later; first we describe how the data of the different types of resource variables is made available to the EM daemon.

Ex.1010 / Page 229 of 572

A shared memory segment is created by the EM daemon when it connects to a resource monitor that can supply Counter and Quantity resource variables. There is one shared memory segment for each such resource monitor.

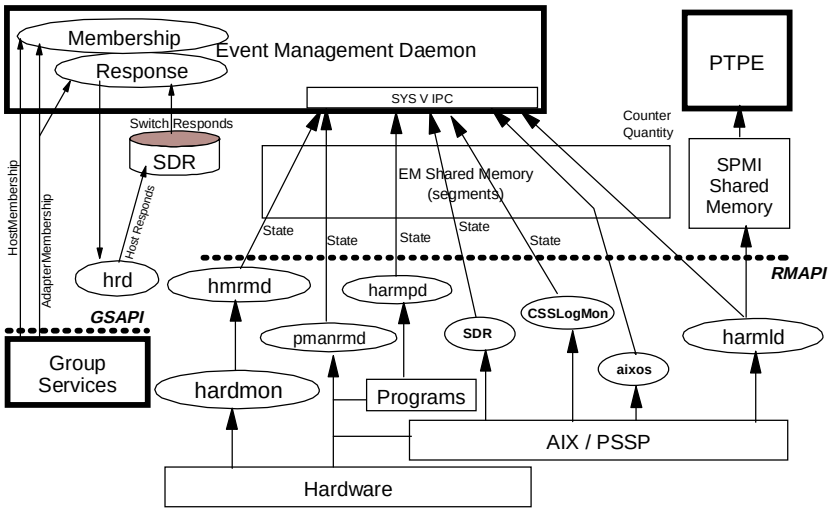


Figure 89. SP Resource Monitors

If Performance Toolbox Parallel Extension (PTPE) is running, the System Performance Measurement Interface (SPMI) shared memory segments are used by the RMAPI to pass data to PTX/PTPE. But these SPMI shared memory segments are not used by the EM daemon. The aixos resource monitor continues to use SPMI to obtain AIX stats because SPMI technology is kept in step with new releases of AIX. By simply making the correct version of perfagent.tools (for the appropriate AIX level) a prerequisite for RSCT, PSSP development avoids the laborious task of writing and maintaining their own mechanism for collecting AIX-level resource data.

You monitor resource variables of type counter and quantity with respect to time. You control how you monitor a resource by associating it with a *resource class*. The resource class defines the following for a resource variable:

- The associated RM
- The observation interval, in seconds
- The reporting interval, in seconds, to the associated RM

The observation and reporting values may be different if the resource variable values are also being passed to PTPE/6000 and/or PTX/6000. For performance measurements, you may want to report the values more frequently, but for EM, there is no need to observe the values as often.

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.

Resource variables of type state are handled differently. Every time they change, the resource monitor responsible for supplying the resource variable sends a message containing the data directly to the EM event processor. The rationale is that, unlike resource variables of type counter or quantity, the Event Manager daemon needs to be aware of every change of resource variables of type state, so the sampling method used with counter and quantity variables cannot be applied to state variables.

The EM daemon also subscribes to GS to receive data about node and adapter resource variables. The EM daemon is a data client and supplier of the Host_Responds daemon (hrd). Notice also that the SDR supplies switch responds data. On the SP, the EM daemon is truly versatile in obtaining data on its resource variables!

The SP implementation provides over 400 defined resource variables. In the spirit of cross-platform RSCT support, the names of resource variables and RMs are minimally prefixed with the vendor and product, so that uniqueness can be maintained across platforms.

We now examine the RMs.

External Resource Monitors

The SP ships with seven external resource monitors that supply data to the EM daemon. Again with reference to Figure 89 on page 206, the seven external RMs are:

- **IBM.PSSP.hmrmd**

  This monitor provides the state of the SP hardware. The information is obtained from the PSSP hardware monitoring subsystem (hardmon). The resource variables are of type state and sent directly to the EM daemon as a message.

- **IBM.PSSP.harmpd**

This monitor examines processes that are executing a particular application. The resource variables can be used to determine whether or not a particular system daemon is running. The resource variables are of type state and sent directly to the EM daemon as a message.

- **IBM.PSSP.harmld**

  This monitor provides switch, VSD, Loadleveler, processor on-line information, and internal variables. It uses shared memory, as it only reports on resource variables of type counter and quantity.

- **IBM.PSSP.pmanrmd**

  This monitor supplies the resource variables of the Problem Management subsystem (PMAN). On your behalf, pmanrmd can execute a program, script, or command to report on various aspects of the system. The resource variables are of type state and sent directly to the EM daemon as a message.

- **aixos**

  This monitor provides resource variables that represent AIX operating system resources. This is a daemon (harmad) with a connection type of server. The SPMI library provides the means to directly query AIX structures (such as kernel and Logical Volume Manager) to supply data for a range of operating system resource variables. The aixos resource monitor calls the SPMI library (part of the perfagent.tools component).

- **IBM.PSSP.CSSLogMon**

  This monitor 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**

  This monitor provides a resource variable that represents the modification state of SDR classes. This is a command-based resource monitor with a connection type of client.

Internal Resource Monitors

The two internal RMs are:

- Membership

  This monitor supplies the resource variables that correspond to node and network adapter state. The information is obtained directly from GS by subscribing to the system groups hostMembership, enMembership, and cssMembership.

- Response

Ex.1010 / Page 232 of 572

This monitor supplies the IBM.PSSP.Response resource variables. The IBM.PSSP.Response.Host.state resource variable is updated based on information coming from the adapter membership information (en0 adapter in particular) supplied by Group Services. The IBM.PSSP.Response.Switch.state resource variable is updated based on the switch_responds class in the SDR. This SDR class is updated by the switch daemon itself.

The Host_Responds daemon (hrd) obtains the host state information by subscribing to the IBM.PSSP.Response.Host.state resource variable through the EMAPI. It receives events from the Response monitor.

You may wonder why these resource monitors are internal. Primarily, the reason is performance.

8.7 Putting It All Together - A Simple Example of Exploiting RSCT

To illustrate many of the concepts in this section, we present a common scenario on the SP: after booting your nodes, how does their host_responds state change to that satisfying green color on the Perspectives hardware monitor?

The example assumes a default SP installation, with PSSP 3.1 installed on each node.

8.7.1 Scenario

Your SP was turned off for the weekend due to scheduled power testing in the building. Early Monday morning (or Saturday morning, depending on your part of the world), you come into the office to restart the SP. You have already booted the CWS, and using Perspectives, are ready to boot the nodes. Your SP has one frame, four thin nodes, a switch, and one (default) partition. The hostnames and IP addresses for the system are as follows:

Slot	Reliable Host Name	en0	css0
1	node1	5.5.5.10	6.6.6.10
2	node2	5.5.5.20	6.6.6.20
3	node3	5.5.5.30	6.6.6.30
4	node4	5.5.5.40	6.6.6.40
CWS	spcw	5.5.5.1	N/A

Ex.1010 / Page 233 of 572

You start the nodes booting from their internal disks. You are staring at the Perspectives monitor, waiting for the icons representing your nodes to change from red to green.

8.7.2 TS Level

Since the CWS is already up, TS has already initialized node and adapter membership groups and created an NCT. The CWS is the only member of the en0 group and node group, and there is no internode connectivity in the NCT. The CWS is both the Group Leader and Crown Prince of the en0 ring, and from its machine list is aware there should be four other members. It is periodically issuing proclaims to the en0 IP addresses of the nodes, asking them to join its Singleton group.

The nodes begin processing the /etc/inittab entries. The TS daemons begin on each node and obtain configuration data from the SDR. After a flurry of proclaims, joins, distributions of topology, and ring mergers, we end up with the following adapter membership groups:

- en0: Group Leader is node4; Crown Prince is node3; members are node2, node1 and spcw.
- css0: Group Leader is node4; Crown Prince is node3; members are node2 and node1.

Figure 90 on page 211 shows the topology graph for en0 (Ethernet) and css0 (switch), both before and after an `Estart` command is issued to bring up the switch. The Group Leaders are designated GL. Notice that each switch adapter is in a Singleton group before the `Estart`. Heartbeating begins. Each TS daemon builds (or updates in the case of the CWS) its topology tables and NCT, puts the NCT in shared memory, then begins accepting requests from clients, namely GS.

Ex.1010 / Page 234 of 572

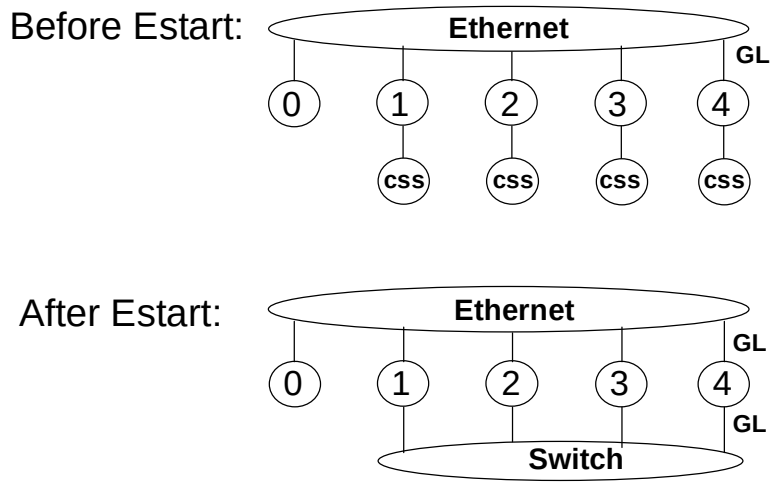


Figure 90. Topology Table for Scenario

8.7.3 GS Level

Before node booting, the CWS was already up and GS on the CWS had already created the internal node and adapter membership groups. The CWS is the only member of the en0 group and node group, and the switch group does not exist. The CWS is both the Group Leader of the en0 membership group and GS nameserver.

From processing /etc/inittab, GS daemons initialize on each node. The GS daemon determines the node number and partition it is running on, then connects to TS (if TS is not accepting connections yet, GS retries every second). Based on the information from TS, GS becomes aware of the other GS daemons that should be out there, and via NCT and Reliable Messaging, starts communicating with them. The CWS, with the lowest node number (0), is confirmed as the GS nameserver.

Since the CWS has the oldest GS daemon, it becomes Group Leader for the en0 and node membership internal groups. The first GS daemon on the node to issue a join request to the css0 internal group becomes the Group Leader of that group (in our case, say it was node3). Membership lists are maintained in the order of GS daemons joining the group. The providers of the groups are in effect the TS daemons running on each node. We have the following groups:

- en0 adapter group: Group Leader is CWS; other providers are node3, node4, node1, and node2. The membership list is maintained in the order of GS daemons joining the group.
- css0: Group Leader is node3; other providers are node4, node1 and node2.
- ha_em_peers: created by the CWS EM daemon, Group Leader is CWS, known only to the CWS GS daemon (which is the nameserver and GSAPI server for the EM daemon on the CWS).

The GS daemons now accept client requests. The hagsglsmd daemon on each node connects to its GS daemon.

8.7.4 EM Level

Before node booting, the CWS was already up and EM on the CWS had already created the ha_em_peers group. The CWS is the only member of the group, and has set the state variable to be the version of the EMCDB it retrieved from the SDR at initialization.

From processing /etc/inittab, EM daemons initialize on each node. Each determines its node number and fetches the version of the EMCDB from the SDR. The EM daemon will not accept any clients until it successfully joins the ha_em_peers group. EM tries to connect to GS every 5 seconds.

An EM daemon request to join the ha_em_peers group involves a 2-phase protocol with one voting phase. The first phase deals with group establishment and the second with actual membership request voting. Let us watch node3, the first node to connect to GS, join:

- First node3 issues a join request to the ha_em_peers group. GS puts a vote request to the other group members including node3, in our case the CWS EM daemon is the only other member. The vote request contains the group state value, which could be empty if this is the first node, or could contain the EMCDB version number.
- In our case, the group state value contains the EMCDB version number. The CWS and node3 EM daemons vote *approve*. If the state variable was null, then this would be the first daemon to join. It would vote *approve*, and the group would be newly formed (this happened to the CWS).
- GS submits a second vote request to the group (second phase), this time as to whether node3 should be allowed to formally join the group. The node3 EM daemon, eager to join, votes APPROVE. The CWS EM daemon votes APPROVE, and the node3 is allowed to join. A join is refused only if a group member is still recovering from a prior termination of the joining

Ex.1010 / Page 236 of 572

daemon. EM daemon will try to join ha_em_peers every 15 seconds until successful.

- The GS nameserver, the CWS GS daemon, updates its membership list for ha_em_peers.

After each EM daemon successfully joins ha_em_peers, it gets the proper version of the EMCDB as described in 8.6.3, “EM and GS” on page 204. Once this is done, EM accepts client requests.

At this point, we have TS, GS, and EM successfully initialized throughout the SP. The EM daemons’ internal Membership RMs update the IBM.PSSP.Membership.LANadapter.state resource variable from its connection to GS, which includes data on the state of the en0 and css0 interfaces by node. Happily, the news is good: TS is successfully heartbeating all en0 and css0 adapters, which GS learns of by subscribing to TS, which EM learns of by its Response monitor connecting to GS.

8.7.5 ES Client - Host_Responds Daemon

On the CWS, the Host_Responds subsystem (hrd) began just after TS was started. Note that hrd runs only on the CWS. It has been patiently waiting for EM to fully initialize, and now connects to the EM daemon as a client.

Across the EMAPI, hrd periodically receives events from the EM daemon. The events describe the state of the en0 adapters for all nodes (IBM.PSSP.Response.Host.state variable). Using this information, hrd modifies the host_responds class in the SDR.

8.7.6 Perspectives Shows Green

Perspectives, an EM client, subscribes to events concerning Response.Host.state. It receives the events reporting the new state of the nodes, and turns their icons green. Now you can go for your first cup of coffee (or tea).

Ex.1010 / Page 237 of 572

Chapter 9. SP Switch Software

The hardware architecture of the SP Switch is outlined in 3.6, “SP Switch Communication Network” on page 58. In the following discussion we concentrate on the operation and management of the software that interfaces with the SP Switch, and review the files and daemons the switch software uses in defining and managing the switch network.

The SP Switch is a high-speed communication network. Internode communication over the SP Switch is supported by two communication protocols

- The so-called *user space* protocol, used to provide direct application access to the switch adapter memory
- The industry standard Internet Protocol (IP) family of communication protocols (which include TCP/IP and UDP/IP)

The user space protocol is sometimes referred to as a lightweight protocol because it requires fewer processor cycles to transmit a given amount of data compared to heavier protocols like TCP/IP. User space is most commonly used for scientific and technical computing applications via a message passing interface or the Low-level API (LAPI). The applications that can utilize the user space protocol include:

- Parallel Operating Environment (POE) for AIX. Applications can be written to use MPI, the originally-supplied and IBM-defined Message Passing Library (MPL), or the Low-Level API (LAPI) introduced in PSSP 2.3.
- Parallel Virtual Machine (PVMe). This product is no longer marketed by IBM.

Prior to the availability of Parallel Environment v2.4, only one user space process per partition could access this switch network in this mode of operation. With Parallel Environment v2.4, support for Multiple User Space Processes Per Adapter was provided.

TCP/IP is utilized in socket communication for many commercial applications, and is the basis for popular network protocols and Web-based services such as Hyper Text Transfer Protocol (HTTP), Network File System (NFS) and File Transfer Protocol (FTP); and applications such as Distributed Computing Environment (DCE). With the possible exception of applications that depend on network-specific functions (for example, LAN broadcasts), most applications work over the switch without modification.

© Copyright IBM Corp. 1999

9.1 Switch Operating Principles

The SP Switch network (at a high level) provides data communication between nodes in a way similar to other networking technologies (such as Ethernet, Token-Ring, FDDI or ATM). However, when it comes to the operating principles which guide the SP Switch network, the similarity ends.

The switch network function is supported by service software known as the *Worm*. This service software is coupled with the *Route Table Generator*, which examines the state of the Switch as determined by the Worm execution, and generates valid routes from every node to any other node. These two components are implemented in the fault-service daemon (fault_service_Worm_RTG_SP) which runs on every node in an SP complex and is central to the functioning of the switch. A diagram illustrating the relationships between the switch communication subsystem components is shown in Figure 91.

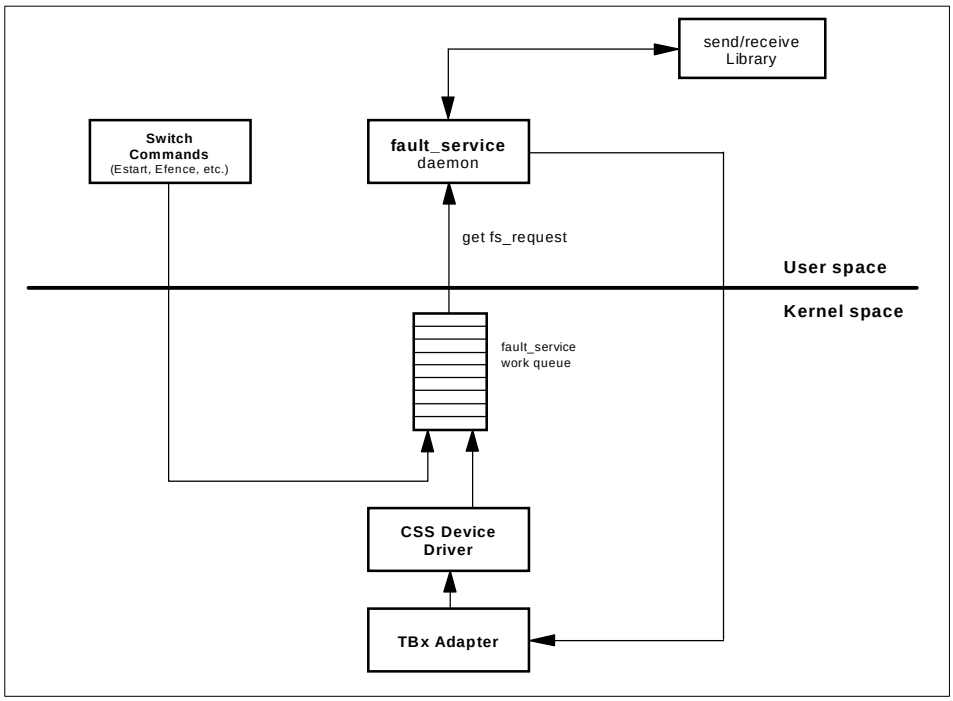


Figure 91. Switch Subsystem Component Architecture

The Worm daemon plays a key role in the coordination of the switch network. It is a non-concurrent server, and therefore can only service one switch event

Ex.1010 / Page 240 of 572

to completion before servicing the next. Examples of switch events (or faults) include switch initialization, Switch Chip error detection/recovery and node (Switch Port) failure. These events are queued by an internal Worm process on the work queue. The events are serviced by the Worm when a get_fs_request system call is made to satisfy a single work queue element.

9.2 Primary Node Behavior

Worm daemons on the nodes, or the Switch Chips themselves, report and log events to a Worm daemon running on one of the nodes. By default, the first node in a partition is defined as this primary node. The primary node also processes switch commands issued from the control workstation. In addition, a primary backup node is also specified, and by default, this is the last node in a partition. These default selections can be overridden.

Error recovery on the SP Switch is done locally: error detection and fault isolation is done at the link level while normal traffic flows across the rest of the switch 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 non-disruptively informed of status changes. (On the older HiPS Switch, a fault on a link would cause the entire switch fabric to be interrupted while recovery was performed.)

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 a predetermined time, it assumes the role of the primary node. This takeover involves reinitializing the switch fabric in a nondisruptive way, selecting another primary backup, and updating the SDR accordingly.

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.

Oncoming primary and oncoming backup primary are used only by the SP Switch. With oncoming nodes, you are now able to change the primary and the primary backup for the next restart of the switch. This is convenient for maintenance purposes, since it allows you to change the oncoming nodes without taking the switch down. Actually, these settings stored in the SDR are informative only, and will only be read by the switch subsystem when an `Estart` command is issued. To summarize, the primary and primary backup fields in the SDR reflect the current state of the system and the oncoming fields are not applicable until the next invocation of the `Estart` command.

9.3 Secondary Nodes Behavior

The nodes that are neither primary nor primary backup nodes are referred to as secondary nodes. The main tasks performed by secondary nodes (in relation to the switch network) are:

- Build the optimal routing table database based on the topology file. This step is performed only when the Worm daemon is started (for example, at boot time).
- Modify the routing table database based on changes received from the primary node (these are messages indicating un-initialized/fenced nodes and/or uninitialized/fenced links).
- Call the Route Table Generator to build the routes from this node to all other nodes.
- Inform the switch protocol what destinations are available.
- Load the new routes down to the adapter.

Secondary nodes send/receive service packets to/from the primary node only. The secondary node acknowledges the primary node for each database change message it receives from the primary node.

9.4 Switch Topology File

The Worm daemon verifies the actual switch topology against an anticipated topology as specified in the switch topology file. This file tells the Worm how many nodes in your configuration are connected via the switch, and is used as the basis for creating the routing tables which get propagated to every node on the switch network. Create this file by copying one of the default topology files provided for each SP configuration. During switch customization, you are required to choose one topology file based on the number and type of the switch boards that are installed in the SP complex. The default topology files are shipped standard with the PSSP software (fileset names are ssp.top and ssp.top.gui) and are located on the control workstation in the /etc/SP directory. These are symbolic links to the files in the /spdata/sys1/syspar_configs/topologies directory.

The topology filename format is:

```
expected.top.<NSBnum>nsb.<ISBnum>isb.type
```

where <NSBnum> is the number of Node Switch Boards that are installed, and <ISBnum> is the number of Intermediate Switch Boards installed in the

SP complex, and the type field specifies the topology type. For example, an installation with three frames and two switches (perhaps 16 thin nodes in the first frame and 16 wide nodes in the next two frames) would use the expected.top.2nsb.0isb.0 file. The SP Switch-8 (8-port, non-scalable switch) uses one unique-namely topology file: `expected.top.1nsb_8.0isb.1`

An extract from a sample topology file is shown in Figure 92.

```
# Node connections in frame L01 to switch 1 in L01
s 15 3  tb0 0 0     L01-S00-BH-J18 to L01-N1
s 15 2  tb0 1 0     L01-S00-BH-J16 to L01-N2
s 16 0  tb0 2 0     L01-S00-BH-J20 to L01-N3
s 16 1  tb0 3 0     L01-S00-BH-J22 to L01-N4
s 15 1  tb0 4 0     L01-S00-BH-J14 to L01-N5
s 15 0  tb0 5 0     L01-S00-BH-J12 to L01-N6
s 16 2  tb0 6 0     L01-S00-BH-J24 to L01-N7
s 16 3  tb0 7 0     L01-S00-BH-J26 to L01-N8
s 14 3  tb0 8 0     L01-S00-BH-J10 to L01-N9
s 14 2  tb0 9 0     L01-S00-BH-J8  to L01-N10
s 17 0  tb0 10 0    L01-S00-BH-J28 to L01-N11
s 17 1  tb0 11 0    L01-S00-BH-J30 to L01-N12
s 14 1  tb0 12 0    L01-S00-BH-J6  to L01-N13
s 14 0  tb0 13 0    L01-S00-BH-J4  to L01-N14
s 17 2  tb0 14 0    L01-S00-BH-J32 to L01-N15
s 17 3  tb0 15 0    L01-S00-BH-J34 to L01-N16
# On board connections between switch chips on switch 1 in Frame L01
s 14 7    s 13 4    L01-S00-SC
s 14 6    s 12 4    L01-S00-SC
s 14 5    s 11 4    L01-S00-SC
s 14 4    s 10 4    L01-S00-SC
s 15 7    s 13 5    L01-S00-SC
s 15 6    s 12 5    L01-S00-SC
s 15 5    s 11 5    L01-S00-SC
s 15 4    s 10 5    L01-S00-SC
s 16 7    s 13 6    L01-S00-SC
s 16 6    s 12 6    L01-S00-SC
s 16 5    s 11 6    L01-S00-SC
s 16 4    s 10 6    L01-S00-SC
s 17 7    s 13 7    L01-S00-SC
s 17 6    s 12 7    L01-S00-SC
s 17 5    s 11 7    L01-S00-SC
s 17 4    s 10 7    L01-S00-SC
```

Figure 92. Sample Topology File

To understand the nomenclature used in the topology file, refer to Figure Figure 93.

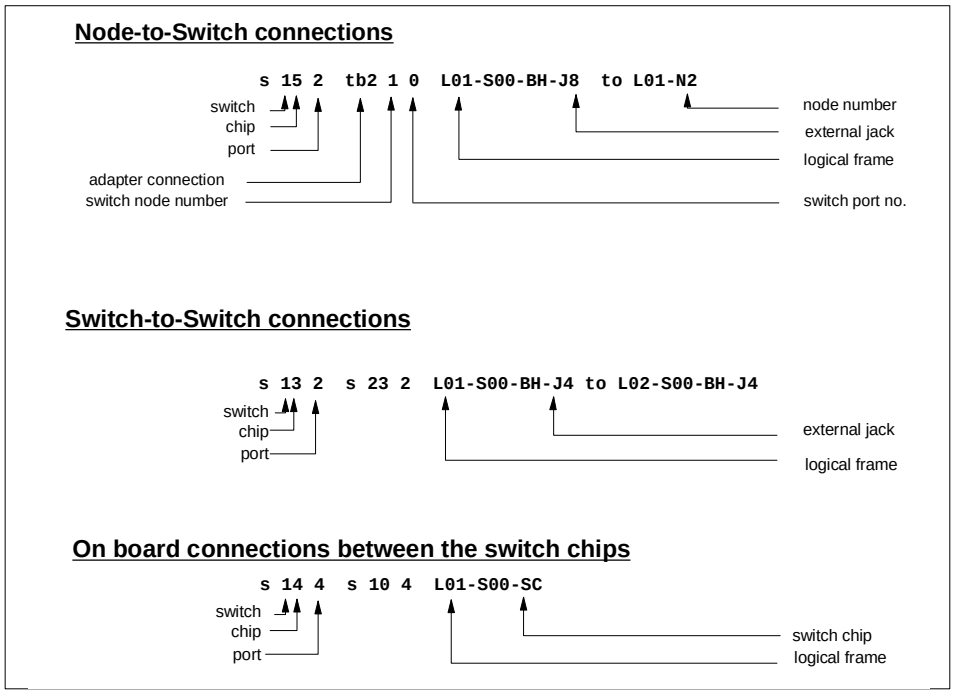


Figure 93. Topology File Field Description

Each line in the topology file describes a link, either between nodes connected to the same switch board or between switches. Fields 2 to 4 describe the switch end of the connectivity, whereas fields 5 to 8 describe the second end (node or another switch) of the connectivity. These fields are further defined as follows:

- Field 1: This is always "s", signifying that this end of the cabling is connected to a port on the switch board.
- Field 2: Describes the switch number where the link is connected to. A "1" in this example means that it is switch board number 1, located inside a normal frame. If this were a switch in a intermediate switch board (ISB) frame, 1000 is added to differentiate it from a switch board in a normal frame.
- Field 3: The switch chip number. The valid values are from 0 to 7, as there could be 8 switch chips on a switch board. Note that this field follows immediately after Field 2 - there is no space separating these two fields. A

number such as 15 would mean that it is switch chip number 5 on switch board number 1. The number 10025 means that it is switch chip 5 on switch board number 2, which is housed in a ISB frame.

- Field 4: The port number on the switch chip.
- Field 5: Notations such as tb0 or tb2 have no relation to the actual type of adapters used in the nodes. They only signify that this end of the connection is used to connect to a node.
- Field 6: The switch port number. Valid range is from 0 to 15.
- Field 7: This single digit field identifies the switch port number at this end of the connection. For a switch adapter on the node, this field is always zero (0). For switch-to-switch connections, it is the switch port number of the receiving switch chip.
- Field 8: Provides the same connectivity information but uses the actual labels that exist on the physical equipment: the jack number, logical frame number and actual node number. The “S” character indicates which slot the switch occupies in the frame, the “BH” characters indicate a Bulk Head connection, the “J” character indicates which jack provides the connection from the switch board, the “N” character indicates the node being connected to the switch, and the “SC” characters indicate the Switch Chip connection. In this example, bulk head (BH), jack 8, on switch 1 (S00), in frame 1 (L01), is connected to node number 2 (N2) in frame 1.

The second diagram in Figure 93 on page 220 shows an example of switch-to-switch connections; notice that there are now two sets of switch end connections that are described. The third and final diagram in this figure illustrates on-board connections between the switch chips of an individual switch board.

An example illustrating how the notations used in the topology file map to the physical layout of the switch is shown in Figure 94 on page 222. It graphically illustrates the sample topology file shown in Figure 92 on page 219.

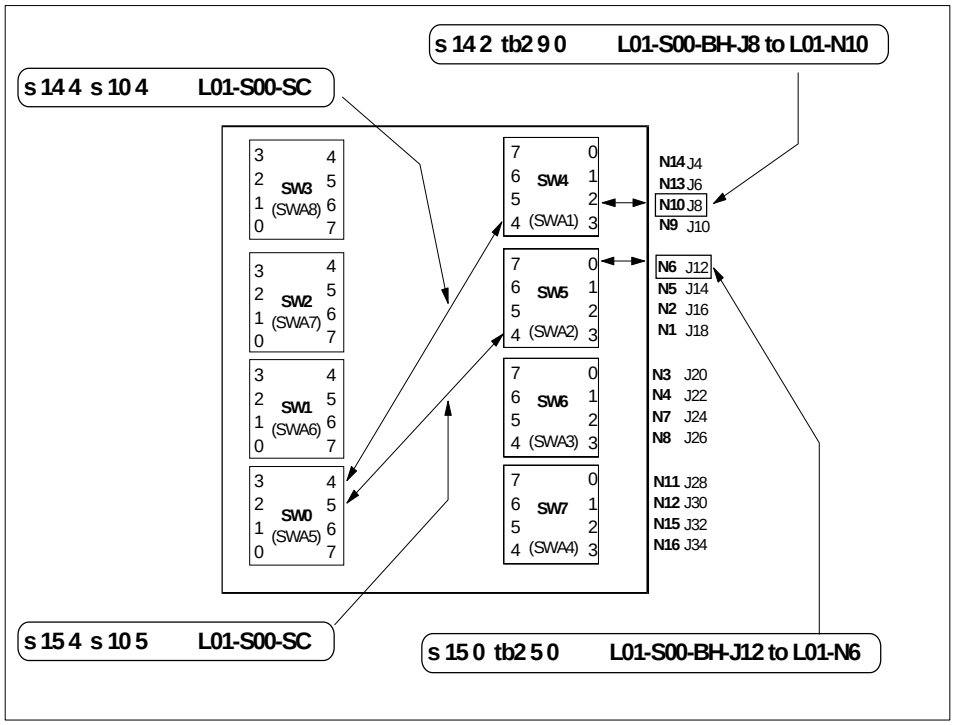


Figure 94. Topology File Information Mapped to Physical Switch Layout

The contents of a topology file are enhanced by running the `Eannotator` command against it. Depending on the customer's physical switch frame configuration defined in the SDR, the `Eannotator` command retrieves switch node and dependent node objects from the SDR and applies proper connection information to the topology file. If the input topology file contains existing connection information, the `Eannotator` command replaces the existing connection label with the new connection labels. It converts what is initially logical information to actual physical information. If the input topology file does not contain connection labels, the `Eannotator` command appends the proper connection label to each line in the topology file.

The precoded connection labels on the topology file start with an “L” which indicate logical frames. The `Eannotator` command replaces the “L” character with an “E”, which indicates physical frames.

`Eannotator` physically checks which connections it is going through (that is, the jack socket or the switch connection) to get to a node or switch chip. This process adds comments to the records of the topology file, namely jack

Ex.1010 / Page 246 of 572

information, which helps anyone reading the file to understand the cabling connections between elements of the system.

The following samples show extracts from a topology file with entries before and after the execution of the `Eannotator` command.

Before:

```
s 15 3 tb0 0 0 L01-S00-BH-J18 to L01-N1
```

After:

```
s 15 3 tb3 0 0 E01-S17-BH-J18 to E01-N1
```

Logical frame L01 is defined as physical frame 1 in the SDR Switch object.

Before:

```
s 10016 0 s 51 3 L09-S1-BH-J20 to L05-S00-BH-J19
```

After:

```
s 10016 0 s 51 3 E10-S1-BH-J20 to E05-S17-BH-J19
```

Logical frame L09 is defined as physical frame 10 in the SDR Switch object.

Before:

```
s 15 3 tb0 0 0 L03-S00-BH-J18 to L03-N3
```

After:

```
s 15 3 tb3 0 0 E03-S17-BH-J18 to E03-N3 # Dependent Node
```

Logical frame L03 is defined as physical frame 3 in the SDR Switch object and the node was determined to be a dependent node.

The additional information embodies an understanding of the cabling of a given switch type used in the system. The system uses this information when reporting errors in the error log or the switch error logs. If these labels are incorrect, it will be difficult to debug problems because the errors may be attributed to the wrong node.

9.5 Routing in SP Switch

The SP routing algorithm, called the Route Table Generator (RTG), selects the routes required for node-to-node communication over the SP Switch network. The selection of the routes is important because of its impact on

performance. In an attempt to prevent congestion in the network, the RTG selects routes to include the links and switches in node-to-node paths in a balanced manner. Furthermore, system software uses the RTG generated multiple routes between each node pair in a round robin fashion to uniformly utilize the network.

The route table approach enables routing to be done in a topology-independent fashion which is important for scalability and fault-tolerance; larger networks can be implemented easily without having to change the switch hardware or the routing algorithm; and the RTG routes around the missing links and switches reported unacceptable by the Worm.

As described earlier, each Switch Board in the SP Switch contains eight physical Switch Chips. Four of the Switch Chips (labeled SW4, SW5, SW6, and SW7 in Figure 94 on page 222) handle all the communications with the nodes, each using chip ports 0, 1, 2, and 3.

The other four chip ports (4, 5, 6, and 7) are used to communicate with the other four switch chips on the opposite side switch board (SW3, SW2, SW1, and SW0). It is possible for each of the chips that service the nodes to reach the other three node chips through four different routes. The switch chips connect together to form multi-stage switching networks.

To see how routing actually works, let us consider the possible paths a switch packet could take from node 14 (N14) communicating to node 16 (N16). Initially, the packet enters SW4 through port 0, and can exit the switch chip through one of the following four routes.

- Port 7 across to SW3, onto that chip through port 4, exiting on port 7, over to SW7, onto the chip on port 7.
- Port 6 across to SW2, onto that chip through port 4, exiting on port 7, over to SW7, onto the chip on port 6.
- Port 5 across to SW1, onto that chip through port 4, exiting on port 7, over to SW7, onto the chip on port 5.
- Port 4 across to SW0, onto that chip through port 4, exiting on port 7, over to SW7, onto the chip on port 4

Once onto SW7, it will exit through port 3, go through J15 to the cable, and to the switch adapter on node 16 (N16). The four possible paths are highlighted in Figure 95 on page 225.

Ex.1010 / Page 248 of 572

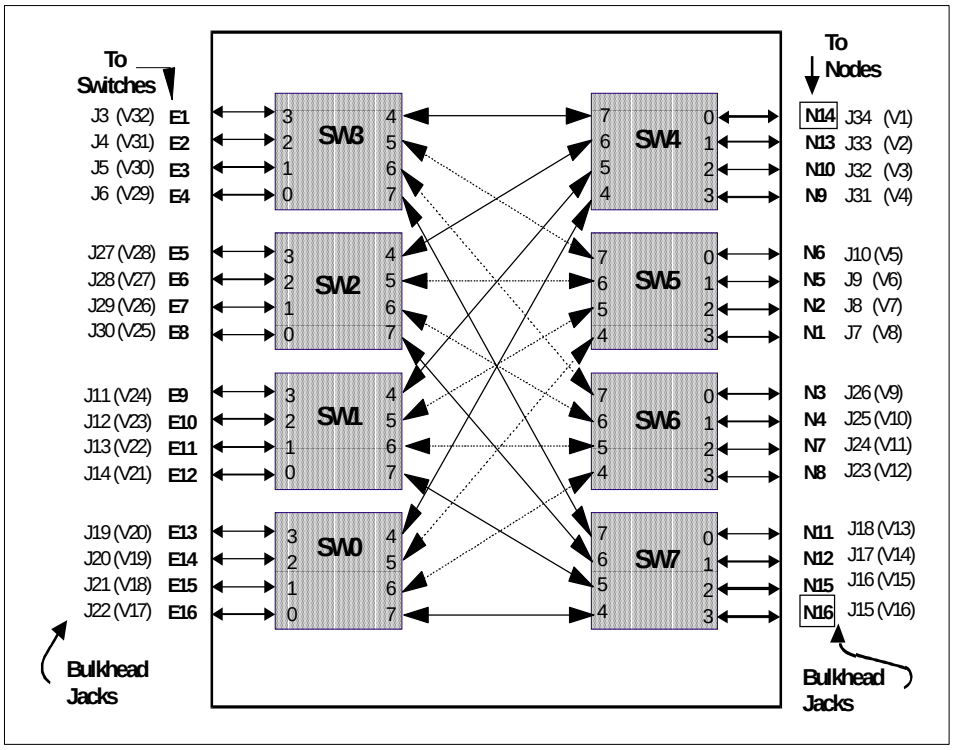


Figure 95. Typical Routing in the SP Switch

9.6 Switch Administration and Management

Since the SP Switch was introduced as part of the RS/6000 SP infrastructure, its management and administration has been predominately a manual task, although many SP system administrators have written scripts to automate the processes specifically for their system environment. Once the Switch and its components have been configured and all nodes within the SP complex are running (that is, all host_responds return "1"), the SP system administrator issues an `Estart` command on the control workstation and the switch network is initialized. If a node was shuts down unexpectedly, without it being flagged to automatically rejoin the switch network, then it has to be manually "re-inserted" into the switch network. This behavior contrasts sharply with other LAN environments, where a node (or server) is immediately available on its networks once the node has been re-booted.

Another concern of many SP system administrators is the difficulty in setting up applications to use the switch network by default, since the switch network

Ex.1010 / Page 249 of 572

is not automatically available on a node reboot. Many SP system administrators have developed their own scripts to circumvent these known limitations.

With the release of PSSP 3.1, many of these SP Switch management and administration issues have been addressed. Specifically, the following changes in the operation of the switch network have been made:

- Switch Admin Daemon

  This daemon will monitor for specific node and switch events to ensure that the switch is re-started automatically in the event that the primary node goes down. Also, if the whole SP complex is powered on, the daemon will automatically start the SP Switch.

- Automatic Node Unfence

  Nodes will rejoin the switch communication fabric without any explicit action by a system administrator. The *autojoin* attribute of the SDR switch_responds class is set whenever nodes join (or rejoin) the switch. This allows the switch network to behave similar to other LANs, since it is immediately available after a node reboot, without any manual intervention.

- Switch-Dependent Application Startup at Boot Time

  For applications that depend on the switch interface being up and available, the switch adapter startup script (rc.switch) has been modified to monitor the status of the switch adapter. After starting the Worm daemon, the script waits (up to 6 minutes) for the css0 adapter to be in an UP status before exiting. Further, the location of the fsd entry in the /etc/inittab file (this entry runs the rc.switch script) has been moved to immediately after the sp entry.

- Centralized Error Logging

  Switch-related AIX error log entries from all nodes installed at PSSP v3.1 are now consolidated into a single file on the control workstation. Entries are ordered by time, and tagged with identifying information, thus providing a summary of switch errors across the entire SP system.

We will review these new features in more detail in the following discussion and in later sections of this chapter.

9.6.1 Switch Admin Daemon

The switch admin daemon was developed to automate the switch startup process by having a daemon monitor for specific node and switch adapter

Ex.1010 / Page 250 of 572

events in all partitions and respond with an automatic Estart whenever appropriate.

Prior to PSSP 3.1, some additional procedures were required before the switch network could be used after the power-on of the system, either in the form of writing a site-specific script to automate `Estart` execution or by having the SP system administrator manually execute the `Estart` command. In PSSP 3.1, there is no need to do this since now the SP Switch network is started automatically by the switch admin daemon.

Attention

Coexistence consideration: The switch admin daemon works with any combination of nodes, since it does not require any modifications to the code in a node. It is required that the event manager daemon running on nodes installed pre-PSSP 3.1 levels be recycled in order to pick up the new resource variables being monitored.

9.6.1.1 Implementation Overview

The cssadm daemon runs under SRC control with the subsystem name of *swtadmd*. It is added to /etc/inittab by the install_cw script during installation, so that it automatically starts up when the control workstation boots.

The configuration file for the cssadm daemon is *cssadm.cfg* in the /spdata/sys1/ha/css directory. By default, this file contains only one non-comment line:

```
Node 1
```

The digit "1" here means that the cssadm daemon will perform the node recovery, that is, it tries to Estart whenever it detects significant node events in the system (for example, the primary node goes down, the primary backup does not comes up). To disable node recovery, change this line from "Node 1" to "Node 0", then stop and restart the cssadm daemon by using the normal SRC commands to stop and start daemons, namely:

```
stopsrc -s swtadmd
startsrc -s swtadmd
```

The cssadm daemon makes use of Event Management to receive notification of events on nodes by registering the following events:

- Node-down-on-switch-adapter event on all nodes
- Node-up-on-host_responds event on all nodes

The logical flow of actions the cssadm daemon takes in response to a node-down-on-switch-adapter event are shown in Figure 96.

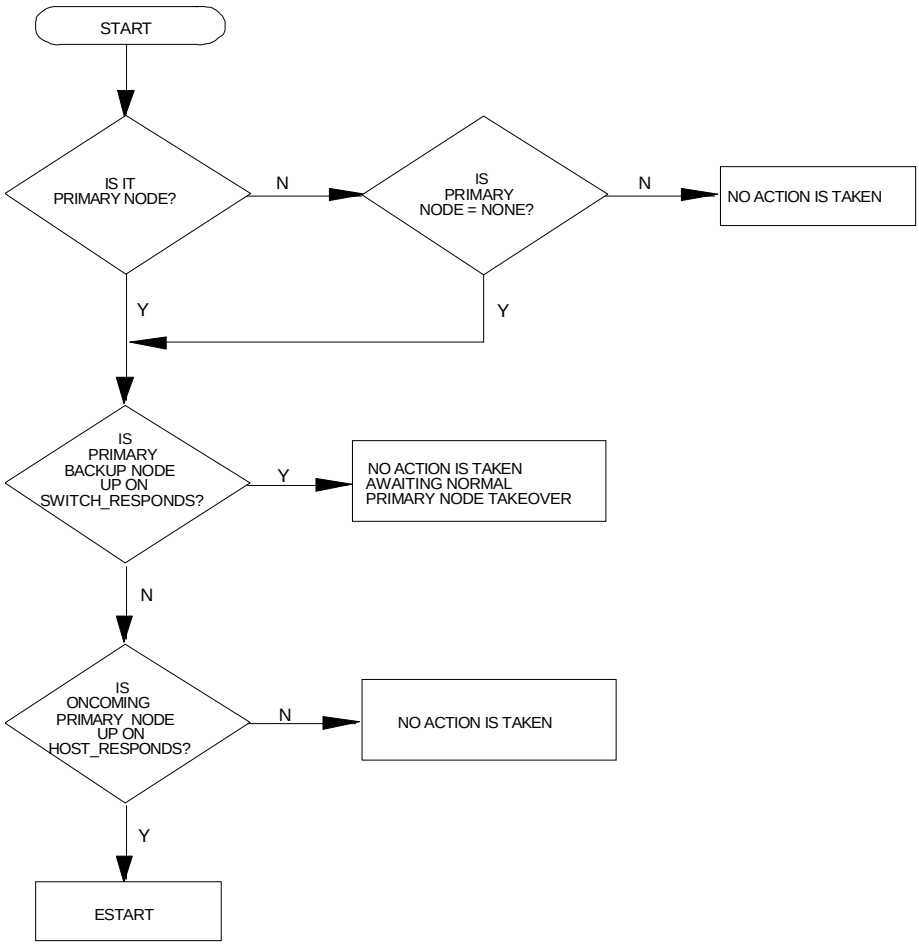


Figure 96. Logic Flow: Node Down on the css0 Interface

Notice that when the primary node is down on the switch but the primary backup node is active, the cssadm daemon takes no action, because the primary backup node will take over the primary node responsibility.

In the situation where both the primary and the primary backup node are down on the switch but the oncoming primary node is not yet up on host_responds, the daemon takes no action but waits until another significant event occurs.

The logical sequence of actions the cssadm daemon takes in response to a node-up-on-host_responds event are shown in Figure 97.

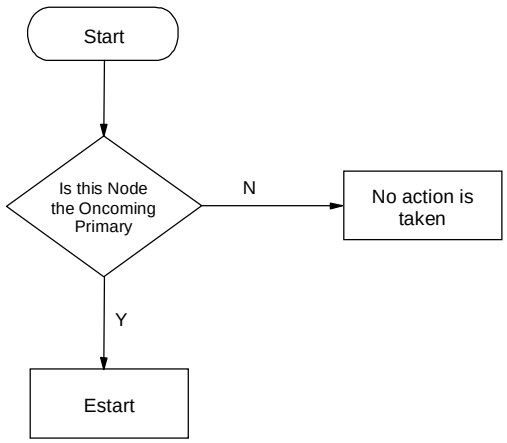


Figure 97. Logic Flow: Node Up on host_responds

Note that the cssadm daemon will only issue an Estart if it determines that the oncoming primary node is up on host_responds.

In all cases, if Estart fails, it logs the errors in the cssadm.stderr log file located in the /var/adm/SPlogs/css directory, but takes no additional recovery actions.

The cssadm daemon generates three output files in the /var/adm/SPlogs/css directory:

- *cssadm.stdout* contains the stdout of the commands executed by the daemon.
- *cssadm.stderr* contains the stderr of the commands executed by the daemon.
- *cssadm.debug* contains entries for each event received, how it was handled, and what the results were.

9.6.1.2 Example Scenarios

In our first example, we simulate the effect of an oncoming primary node becoming available by stopping and restarting the hats daemon on this node. In this example, the primary node is node 14, and the oncoming primary node is node 1. The operation of the cssadm daemon can be determined by reviewing the cssadm.debug file, part of which is shown in Figure 98 on page 230.

```
[sp3en0:/]# Eprimary
14      - primary
1       - oncoming primary
5       - primary backup
14      - oncoming primary backup
[sp3en0:/]# dsh -w sp3n01 stopsrc -s hats
sp3n01: 0513-044 The stop of the hats Subsystem was completed successfully.
[sp3en0:/]# dsh -w sp3n01 startsrc -s hats
sp3n01: 0513-059 The hats Subsystem has been started. Subsystem PID is 17772.
[sp3en0:/]# tail -f /var/adm/SPlogs/css/cssadm.debug
------------------------------------------------------------
Received Event:
------------------------------------------------------------
event_type     = node up
node number     = 1
time = Fri Feb 19 15:01:41 1999

complete        = 2

(i) cssadm: Processing event:
------------------------------------------------------------
Processing Event:
------------------------------------------------------------
event_type     = node up
node number     = 1
time = Fri Feb 19 15:01:41 1999

complete        = 2
(i)  cssadm: Oncoming primary node has come up on Host Responds in partition
sp3en0. Estart will be run
(i) cssadm: Estart successful in partition sp3en0.
^C[sp3en0:/]# Eprimary
1       - primary
1       - oncoming primary
14      - primary backup
14      - oncoming primary backup
[sp3en0:/]#
```

Figure 98. cssadm Daemon Response #1

By observing the contents of the cssadm.debug file we see that the `Estart` command was issued in response to the oncoming primary node (node 1) returning a positive host_responds. The console dialog also shows that node 1 is now the primary node.

In a second example, we simulate the loss of both the primary node (node 5) and the primary backup node (node 15) from the switch network by killing the Worm daemon on both nodes at the same time. The console log output, along with the relevant contents of the cssadm.debug file, are shown in Figure 99 on page 231.

Ex.1010 / Page 254 of 572

```
[sp3en0:/]# Eprimary
5       - primary
1       - oncoming primary
15      - primary backup
14      - oncoming primary backup
[sp3en0:/]# dsh -w sp3n05,sp3n15 ps -ef | grep Worm | grep -v grep
sp3n05:     root 19574     1   0 12:41:09      -  0:09 /usr/lpp/ssp/css/fault_service_Worm_RTG_SP
-r 4 -b 1 -s 5 -p 1 -a TB3 -t 22
sp3n15:     root  6086     1   0 14:24:43      -  0:00 /usr/lpp/ssp/css/fault_service_Worm_RTG_SP
-r 14 -b 1 -s 7 -p 2 -a TB3 -t 22
[sp3en0:/]# dsh -w sp3n05 kill 19574 ; dsh -w sp3n15 kill 6086
[sp3en0:/]# tail -f /var/adm/SPlogs/css/cssadm.debug
Processing Event:
------------------------------------------------------------
event_type     = node down
node number     = 1
time = Fri Feb 19 15:22:34 1999

complete        = 2
(i) cssadm: Primary node is down on switch responds in partition sp3en0.
  Checking primary backup.
(i) cssadm: Primary backup node sp3n14.msc.itso.ibm.com is up.  No action taken, await
ing normal primary node takeover.
------------------------------------------------------------
Received Event:
------------------------------------------------------------
event_type     = node down
node number     = 5
time = Fri Feb 19 15:32:24 1999

complete        = 2

(i) cssadm: Processing event:
------------------------------------------------------------
Processing Event:
------------------------------------------------------------
event_type     = node down
node number     = 5
time = Fri Feb 19 15:32:24 1999

complete        = 2
(i) cssadm: Primary node is down on switch responds in partition sp3en0.
  Checking primary backup.
(i) cssadm: Primary backup is not up on switch responds.  Checking
 if oncoming primary is up on host responds.
(i) cssadm: Oncoming primary up on host responds.  Going to Estart.
^C[sp3en0:/]# Eprimary
1       - primary
1       - oncoming primary
14      - primary backup
14      - oncoming primary backup
[sp3en0:/]#
```

Figure 99. cssadm Daemon Response #2

Though the cssadm daemon detected that both the primary node and the primary backup node are down, it found that the oncoming primary node is up, so it issued an `Estart` to get the switch network back to normal.

9.6.2 Automatic Node Unfence

Prior to PSSP 3.1, the autojoin attribute in the switch_responds class was normally turned off for all nodes. The only exception was that a node that was explicitly fenced with the `Efence -autojoin` command had this attribute turned on.

With PSSP 3.1, the autojoin attribute is now turned on for all nodes. Exceptions are nodes fenced with the Efence command and nodes that the fault service daemon considers to have a persistent problem.

9.6.2.1 Autojoin is Now Default

The possible states that a node's switch_responds object could be in and how they are treated by the primary node are outlined in Table 5.

Table 5. A Node's switch_responds State Table

switch_responds	autojoin	isolated	Description
1	X	X	up on Switch
0	0	1	Fenced & isolated
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 or isolated: it will remain off the switch fabric until it is unfenced or the autojoin attribute is set. Nodes that are fenced with their autojoin attribute set will get unfenced automatically by the switch primary node.

9.6.2.2 Changes to the Switch Scan Function

The switch scan, conducted every two minutes by the primary node, has been modified to unfence any nodes that have the autojoin attribute set when they are "ready" to come up on the switch — that is, the switch adapter microcode has been loaded and the fault service daemon (or its dependent node equivalent) is running. Thus, all nodes that are rebooted or powered off and on will automatically come up on the switch after the primary node determines that they are ready.

However, if a node has an intermittent problem with the switch adapter, it may continually be unfenced and refenced. This causes unnecessary work to be done by the primary node. To avoid this problem, two mechanisms are put into place:

- If the fault service daemon on the failing node reaches an error threshold or detects an unrecoverable error, it puts its TBIC (see the following note) into reset and sets the autojoin attribute to off. Once this occurs, the node will not unfence until the rc.switch script is run to recover the node, and then Eunfence is executed.

Ex.1010 / Page 256 of 572

Note: TBIC stands for *Trail Blazer Interface Chip*, a chip on the switch adapter card that manages the link from the switch adapter to the switch board.

Any node failing to Eunfence signals the administrator that the node has a problem and needs to be diagnosed and recovered.

- If the primary fault service daemon fails to unfence a particular node three consecutive times after its link initializes, its autojoin attribute is set off.

  No further attempt to auto-unfence will be made until the system administrator runs Eunfence.

9.6.2.3 Example Scenarios

The first sample console dialog, in Figure 100, shows a node being fenced with the autojoin option, and then we see that it has been automatically unfenced.

```
[sp3en0:/]# SDRGetObjects switch_responds node_number==7
node_number  switch_responds autojoin     isolated    adapter_config_status
          7            1           1            0 css_ready

[sp3en0:/]# Efence -autojoin 7
All nodes successfully fenced.

[sp3en0:/]# date
Fri Feb 19 19:55:23 EST 1999

[sp3en0:/]# SDRGetObjects switch_responds node_number==7
node_number  switch_responds autojoin     isolated    adapter_config_status
          7            0           1            1 css_ready

[sp3en0:/]# SDRGetObjects switch_responds node_number==7
node_number  switch_responds autojoin     isolated    adapter_config_status
          7            1           1            0 css_ready

[sp3en0:/]# date
Fri Feb 19 19:58:10 EST 1999
```

Figure 100. Console Dialog: Node Fence with Autojoin

The sample console dialog in Figure 101 on page 234 shows the status of the switch_responds class for a node being rebooted.

```
[sp3en0:]# SDRGetObjects switch_responds node_number==8
node_number  switch_responds autojoin     isolated     adapter_config_status
           8            1            1            0 css_ready

[sp3en0:]# date
Fri Feb 19 19:39:23 EST 1999

[sp3en0:/]# cshutdown -Fr -N 8
Progress recorded in /var/adm/SPlogs/cs/cshut.0219193959.46060.
Progress recorded in /var/adm/SPlogs/cs/cstart.0219194224.30326.

[sp3en0:]# date
Fri Feb 19 19:47:05 EST 1999

[sp3en0:/]# SDRGetObjects switch_responds node_number==8
node_number  switch_responds autojoin     isolated     adapter_config_status
           8            1            1            0 css_ready
```

Figure 101. Console Dialog #1: Node Reboot

A periodic review of the status of this node also shows the new method of operation of the automatic unfence function. An example of this is in Figure 102.

```
---------------------------------- Frame 1 -------------------------------------
Frame  Node    Node         Host/Switch  Key     Env   Front Panel   LCD/LED is
Slot   Number  Type  Power   Responds    Switch  Fail    LCD/LED      Flashing

--------------------------------------------------------------------------------
  8       8    thin    on   yes  yes     normal   no   LEDs are blank   no

(node up)
...

                                                  _   _   _
  8       8    thin    on    no  autojn  normal   no   |_  |_   _|      no
                                                    _|  _|  _|
(node being rebooted)
...
  8       8    thin    on   yes  autojn  normal   no   LEDs are blank   no

(node up, but still fenced from switch)
...
  8       8    thin    on   yes  yes     normal   no   LEDs are blank   no

(node up, and running on switch)
```

Figure 102. Console Dialog #2: Node Reboot

Notice that the switch_responds field now informs the system administrator that the autojoin option has been set.

9.6.3 Switch-Dependent Application Startup

In PSSP 3.1, there were a number of simple changes to the switch startup procedures, making it easier to automate the startup of applications that depend on the availability of an operating switch network. These changes are:

- The fsd entry in /etc/inittab has been placed immediately after the sp entry. This entry executes the rc.switch script which starts the Worm daemon. Its new placement ensures that the switch network is available as early as possible in the system startup cycle.
- The action field of the fsd entry can now be changed from *once* to *wait*. The rc.switch script has been modified to recognize this change. If the field is set to *wait*, after starting the Worm, the script will wait for a maximum of 6 minutes before exiting. Then the rest of the /etc/inittab file will be processed.
- Entries in /etc/inittab for applications that depend on the availability of the switch network should be placed after the rc.switch entry.

Attention

By default, the action field is set to *once*, so the behavior of the inittab processing is the same as in previous releases of PSSP. You must make the necessary changes to the inittab file to take advantage of these new features.

9.6.4 SP Switch Management

With the introduction of the switch admin daemon (cssadm), there should be fewer instances when the SP system administrator is required to manually start the SP Switch.

9.6.4.1 Starting the Switch

When an `Estart` command is issued (either manually or via the cssadm daemon), the following steps are followed during the switch initialization process:

1. An `Estart` command is issued for a system partition from the control workstation through SMIT or the command line, and this is relayed to the primary node of the partition as an `Estart_sw`.

2. The primary node consults the SDR file for the current topology file and the current fenced nodes.
3. The primary node distributes the topology file to the bootserver nodes of its partitions. Then, it directs each boot/install server to distribute the topology file to its client nodes.

   Each boot/install server distributes the topology file, tests for success, and returns the resulting status to the primary.

   The primary node logs and evaluates failure data.
4. The primary runs the Worm code to verify the partition's fabric pool and nodes as follows:
   - Each chip is sent a read status service packet to check cabling.
   - Each chip is sent an initialization service packet to set the chip ID and specify routes to the primary.
   - Each non-primary unfenced node is sent an initialization service packet to set its personality and specify the topology file for this partition.
   - Each non-primary unfenced node is sent a read status node service packet to verify the switch address, personality, and readiness.
   - The primary node updates its device database in accordance with the finding.
5. The primary node sets the Time-of-Day (TOD) across the fabric pool as follows:
   - Traffic is stopped, if necessary.
   - The TOD is propagated.
   - Traffic is resumed, if necessary.
6. The primary node downloads its route table to its TB3 adapter.
7. The primary node distributes device database *deltas* to the node over the switch, through the device database service package.
8. On each non-primary, unfenced node:
   - The local device database is updated with the *deltas*.
   - Routes are computed.
   - The routes are loaded to TB3 adapter SRAM.
9. On each node, the communications protocols and LoadLeveler code, as appropriate, are notified that the switch is available.
10. The primary node updates the SDR switch_responds class for its partition. For each node in the partition, the (autojoin, isolated) pair is set to one of the following:

    (0,0) Initial

(0,1)	Fenced
(1,0)	On
(1,1)	Suspended

9.6.4.2 Fencing and Unfencing the Nodes

With the automatic Node Unfence function provided in PSSP 3.1, nodes will rejoin the switch network without any explicit action by the system administrator or operator.

If you need to fence a node off the switch network, and not have it rejoin the switch network, you must fence the node with the `Efence` command, which turns off the automatic rejoin attribute in the switch_responds class. If you do not have the autojoin attribute set, the fault service daemon (Worm) will not automatically unfence it during `Estart`. The node will remain fenced until either it is unfenced using the `Eunfence` command or the autojoin attribute is set in the SDR.

However, in a coexistence environment, if the primary node is at PSSP 2.4 or earlier, the autojoin option enables the nodes in the argument list to be fenced and to automatically rejoin the current switch network if the node is rebooted or the Worm daemon is restarted.

Notes:

1. The primary node and primary backup node for an SP switch cannot be fenced.
2. A node which is Efenced with autojoin will automatically join the switch network within two minutes due to the new automatic unfence function in PSSP 3.1.

9.6.4.3 Monitoring Nodes on a Switch

The new Switch Administration daemon (cssadm) and the automatic unfence options provide the same functions as the Emonitor subsystem that was available in earlier releases of PSSP.

However, if you turn off the Switch Administration daemon functions you may still wish to use the Emonitor subsystem. Furthermore, if you are using a primary node with a code version of PSSP v2.4 or earlier in a coexistence environment, the new automatic unfence functions are not provided by the Worm daemon.

You can monitor 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 partition and is named

Ex.1010 / Page 261 of 572

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 by specifying the -m option with the `Estart` command. Once invoked, it is SRC-controlled and so will restart if halted abnormally. To end monitoring, run the following command, which stops the daemon in the system partition:

```
/usr/lpp/ssp/bin/emonctrl -k
```

9.6.4.4 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 the nodes are shut down or rebooted, they are not fenced from the switch.

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`, and `Etopology` commands are functional after the `Equiesce` command is issued.

An `Estart` command must be issued when the global activity is complete to re-establish switch recovery and primary node takeover.

9.7 Switch Clocks

The switch network is a synchronized network with a clock (oscillator) source, usually within the same frame. For systems with multiple switch boards, there is only one clock source that is distributed. One of the switch boards is designated as Clock Master, which then distributes its internal clock to all other switch boards.

The clock subsystem is critical for the proper functioning of the switch fabric. In order to assure that every switch board is receiving the same clock signal, usually alternative sources are designated in case the primary alternative fails. However, this switchover is not automatic; it requires manual intervention.

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.

Ex.1010 / Page 262 of 572

The Eclock topology files are located in the /etc/SP directory on the control workstation. The filename format is:

```
Eclock.top.<NSBnum>nsb.<ISBnum>isb.0
```

where <NSBnum> is the number of Node Switch Boards, and <ISBnum> is the number of Intermediate Switch Boards installed in the SP complex. For example, a simple installation with three frames and two switches (one possible configuration could be 16 thin nodes in the first frame and 16 wide nodes in the next two frames) would use the Eclock.top.2nsb.0isb.0 file.

If `Eclock` is run to change the clock multiplexor settings while the switch is operational, you will experience switch outages until a subsequent `Estart` is completed. If you run `Eclock` and specify the -f, -a, -r or -d flags, you do not need to run `Estart` if the switch admin daemon (swtadmd) subsystem is active. In this case the subsystem runs `Estart` for you. These flags perform the following operations:

-f *Eclock_topology file*
: Specifies the file name of the clock topology file containing the initial switch clock input values for all switches in the system.

-a *Eclock_topology file*
: Uses the alternate Eclock topology specified in the given clock topology file.

-r
: Extracts the clock topology file information from the System Data Repository (SDR) and initializes the switch clock inputs for all switches in the system.

-d
: Detects the switch configuration, automatically selects the clock topology file, and initializes the switch clock inputs for all switches in the system.

Since Eclock operates across system partitions, if you specified the -f, -a, -r or -d flag, you must run the `Estart` command in *all* system partitions unless the swtadmd subsystem is active. In this case the subsystem runs `Estart` for you. If you use the -s flag (which, together with the -m flag, sets the specific clock source for an individual switch), then the `Eclock` command operates just on the specified switch board. In this case, you need to run `Estart` only in the partitions which share that switch board. However, if you used the -s flag to reset the master switch board, the effect is the same as having issued a global `Eclock` command and you must run `Estart` in all partitions. The -s flag will recycle the Worm daemons only on the nodes connected to the target switch boards.

9.8 SP Switch Error Logging

The primary impetus for developing a consolidated error logging mechanism in PSSP 3.1 was to provide a centralized location for viewing error activity generated by the SP Switch. Switch-related AIX error log entries from all nodes are now consolidated into a single file on the control workstation. This mechanism provides a summary of all significant switch-related events that occur in the entire SP complex.

In previous releases of PSSP, system administrators were required to collect data and log file information from the primary node, and the nodes which were suspected to cause the switch error. Further, they were required to correlate in time the gathered information from the various sources to generate the sequence of events that led up to the switch or switch adapter error.

9.8.1 Consolidated Switch Error Logging

Using the new error logging daemon provided in PSSP 3.1, entries are ordered by time, and tagged with identifying information, thus providing a summary of switch errors across the entire SP system. The new log file /var/adm/SPlogs/css/summlog (a symbolic link to the same file in directory /spdata/sys1/ha/css) is written to by a new daemon called css.summlog. This daemon runs only on the control workstation and listens for log change events generated by the ODM method running on the nodes that make up the SP complex.

Additionally, some switch events not only create a summary record on the control workstation, but also trigger a snapshot, generated by the css.snap utility on the node logging the error. This utility captures the data necessary for further problem determination at the most appropriate time, when the error happened, and thus helps shorten problem resolution time.

The new summary log file has entries with the following format:

- Timestamp - in the form of MMDDhhmmYYYY
- Nodename - short reliable hostname
- Snap - "Y" or "N", indicates whether a snap was taken
- Partition name - system partition name or global
- Index - the sequence number field in the AIX error log
- Label - the label field in the AIX error log

When the summlog file size is greater than 3MB, it is renamed to summlog.old and a new summlog file is opened.

Ex.1010 / Page 264 of 572

Attention

Coexistence considerations: This new function operates only on nodes which have been installed at PSSP 3.1 or later. Earlier versions of PSSP will not trigger the logging of switch-related error record entries into the new log file on the control workstation. In SP systems with nodes at different levels of PSSP, only PSSP 3.1 nodes will have summary records in the consolidated log on the control workstation.

9.8.1.1 Implementation Overview

A new daemon, css.summlog was implemented on the control workstation to monitor for a switch-related event added to the AIX error log on any nodes. Once such an event occurs, a summary log record is generated in a new log file, summlog, in the /var/adm/SPlogs/css directory on the control workstation. (This file is actually a symbolic link to the summlog file in the /spdata/sys1/ha/css directory.)

The daemon css.summlog connects to the Event Manager daemon in each partition and generates a summary log record in the summlog file for every event it receives. It operates under SRC control with the subsystem name of *swtlog*, and is added to /etc/inittab by the install_cw script during SP system installation so that it automatically starts up when the control workstation boots.

Other changes to support this function are:

- The `css.snap` command was modified to accept a new flag, -s, to indicate that this is a soft dump and so the tb<x>dump command will not be invoked. This utility now also extracts the switch adapter information from ODM.
- The haemloadlist script (part of the rsct.basic.sp fileset) was modified to include the new resource variables.

9.8.1.2 New Resource Variables

A new resource variable called IBM.PSSP.CSSlog.errlog is defined that is specific to CSS log consolidation function. The intent is to support the generation of events reflecting that a new switch-related entry was made to the local AIX error log. It provides the index of the entry within the error log, the label and the time of the entry.

This resource variable is made part of the new IBM.PSSP.CSSlog resource class since its resource monitor is IBM.PSSP.CSSlogMon.

Ex.1010 / Page 265 of 572

The detail of each field in this resource variable is shown in Table 6.

Table 6. IBM.PSSP.CSSlog.errlog is a Structured Byte String with These Fields

Name	Type	Description
lastupdt	long	time of last update in seconds
entrynum	long	entry number in log
version	cstring	version number (for future use)
symptom	cstring	symptom string

9.8.1.3 Modified Error Notification Objects

Sixty-eight ODM error notification objects for switch error log entry are modified to use a new command, css.logevnt as the error notification method (instead of errlog_rm as in previous releases).

All switch-related AIX error log entries that occur on a node and that generate a summary log record on the control workstation are shown in Table 7.

Table 7. AIX Error Log Entries

Error Label	Description
HPS_FAULT6_ER	Switch Fault Service Daemon Terminated
HPS_FAULT9_ER	Switch Adapter - Bus error
SP_CLCK_MISS_RE	Switch (non-master) lost clock
SP_CSS_IF_FAIL_ER	Switch adapter i/f system call failed
SP_MCLCK_MISS_RE	Switch (master oscillator) lost clock
SP_PROCESS_KILLD_RE	Process killed due to link outage
SP_SW_ACK_FAILED_RE	Switch daemon ACK of svc command failed
SP_SW_BCKUP_TOVR_RE	Switch primary backup node takeover
SP_SW_CBCST_FAIL_RE	Switch daemon command broadcast failed
SP_SW_CRC_SVCPKT_RE	Switch service logic incorrect CRC
SP_SW_DNODE_FAIL_RE	Switch daemon dependent node svc failure
SP_SW_ECLOCK_RE	Eclock command issued by user
SP_SW_EDCTHRSHLD_RE	Switch rcvr EDC errors exceed threshold
SP_SW_EDC_ERROR_RE	Receiver EDC-class error

Ex.1010 / Page 266 of 572

Error Label	Description
SP_SW_ESTRT_FAIL_RE	Estart failed
SP_SW_FENCE_FAIL_RE	Fence of node failed
SP_SW_FIFOOVRFLW_RE	Switch receiver FIFO overflow error
SP_SW_GET_SVCREQ_ER	Switch daemon could not get svc request
SP_SW_INIT_FAIL_ER	Switch daemon initialization failed
SP_SW_INVALD_RTE_RE	Switch sender invalid route error
SP_SW_IP_RESET_ER	Switch daemon could not reset IP
SP_SW_LNK_ENABLE_RE	Switch svc logic invalid link enable
SP_SW_LOGFAILURE_RE	Error writing switch log files
SP_SW_LST_BUP_CT_RE	Primary backup node not responding
SP_SW_MISWIRE_ER	Switch - cable mis-wired
SP_SW_NCLL_UNINT_RE	Switch central queue NCLL uninitialized
SP_SW_NODEMISW_RE	Switch node miswired
SP_SW_OFFLINE_RE	Node fence request received
SP_SW_PE_INBFIFO_RE	Switch svc logic bad parity - inFIFO
SP_SW_PE_ON_DATA_RE	Switch sender parity error on data
SP_SW_PE_ON_NCLL_RE	Switch central queue parity error - NCLL
SP_SW_PE_ON_NMLL_RE	Switch central queue parity error - NMLL
SP_SW_PE_RTE_TBL_RE	Switch svc logic bad parity - route tbl
SP_SW_PRI_TAKOVR_RE	Switch primary node takeover
SP_SW_RCVLNKSYNC_RE	Switch receiver link sync error
SP_SW_RECV_STATE_RE	Switch receiver state machine error
SP_SW_REOP_WIN_ER	Switch daemon reopen windows failed
SP_SW_ROUTE_VIOL_RE	Recvr Route Violation Error
SP_SW_RSGN_BKUP_RE	Resigning as switch primary backup
SP_SW_RSGN_PRIM_RE	Resigning switch primaryship
SP_SW_RTE_GEN_RE	Switch daemon failed to generate routes

Error Label	Description
SP_SW_SCAN_FAIL_ER	Switch scan failed
SP_SW_SDR_FAIL_RE	Switch daemon SDR communications failed
SP_SW_SEND_TOD_RE	Switch svc logic send TOD error
SP_SW_SIGTERM_ER	Switch daemon received SIGTERM
SP_SW_SNDLNKSYNC_RE	Switch sender link sync error
SP_SW_SNDLOSTEOP_RE	Sender Lost EOP Error
SP_SW_SNDTKNTHRS_RE	Switch snd token errors exceed threshold
SP_SW_SND_STATE_RE	Switch sender state machine error
SP_SW_STIDATARET_RE	Recvr STI Data Re-Time Request
SP_SW_STITOKN_RT_RE	Sender STI Token Re-Time Request
SP_SW_SVC_PKTLEN_RE	Switch svc logic saw bad packet length
SP_SW_SVC_Q_FULL_RE	Switch service send queue full
SP_SW_SVC_STATE_RE	Switch svc logic state machine error
SP_SW_UBCST_FAIL_RE	Switch daemon DBupdate broadcast failed
SP_SW_UNINI_LINK_RE	Links not initialized during Estart
SP_SW_UNINI_NODE_RE	Nodes not initialized during Estart
TB3_BAD_PACKET_RE	Bad packet received
TB3_CONFIG1_ER	Failed to update ODM during CSS config
TB3_HARDWARE_ER	Switch adapter hardware/microcode error
TB3_LINK_RE	Switch adapter link outage
TB3_MICROCODE_ER	Switch adapter microcode error
TB3_PIO_ER	I/O error
TB3_SLIH_ER	Switch adapter interrupt handler error
TB3_SVC_QUE_FULL_ER	Switch adapter svc interface overrun
TB3_THRESHOLD_ER	Switch adapter error threshold exceeded
TB3_TRANSIENT_RE	Switch adapter transient error
TBS_HARDWARE_ER	Switch board hardware error

Ex.1010 / Page 268 of 572

9.8.1.4 New Error Notification Method

The new error notification method, css.logevnt, does the following:

1. Invokes errlog_rm if ssp.pman is installed.

   errlog_rm is a resource monitor that generates an event indicating that the AIX error log has been updated.

2. Optionally takes a full or soft css.snap run.

   Example of errors that cause a full css.snap run to be taken:

   - Switch adapter link outage
   - Switch adapter error threshold exceeded
   - Switch adapter service interface overrun

   Example of errors that cause a soft css.snap to be taken:

   - Switch sender parity error on data
   - Switch central queue parity error
   - Switch adapter device driver I/O error

3. Generates log change event to trigger summary record creation on the control workstation.

4. Ensures that its stdout/stderr file logevnt.out, located in directory /var/adm/SPlogs/css, is not larger than 1 MB. When it is more than 1 MB, it is renamed to logevnt.out.old and a new logevnt.out file is opened.

9.8.1.5 Example Scenario

In the following example, there is a single frame SP, with the primary node being node 1, and the primary backup node being node 7. The loss of the primary node was simulated by killing the Worm daemon, and the results of this activity can be seen by monitoring the summlog file on the control workstation and the standard AIX error logs on the nodes themselves. The dialog generated at the control workstation is shown in Figure 103 on page 246.

Ex.1010 / Page 269 of 572

```
[sp3en0:/]# date
Thu Feb 18 16:07:34 EST 1999

[sp3en0:/]# Eprimary
1       - primary
1       - oncoming primary
7       - primary backup
14      - oncoming primary backup

[sp3en0:/]# dsh -w sp3n01 ps -ef | grep Worm | grep -v grep
sp3n01:     root 17550     1   0 11:33:05      -  0:35
/usr/lpp/ssp/css/fault_service_Worm_RTG_SP -r 0 -b 1 -s 5 -p 3 -a TB3 -t 22

[sp3en0:/]# dsh -w sp3n01 kill 17550

[sp3en0:/]# tail -f /var/adm/SPlogs/css/summlog
021816071999 sp3n01 N sp3en0 64 SP_SW_SIGTERM_ER
021816071999 sp3n01 N sp3en0 65 HPS_FAULT6_ER
021816101999 sp3n07 N sp3en0 72 SP_SW_UNINI_NODE_RE
021816101999 sp3n07 N sp3en0 73 SP_SW_UNINI_LINK_RE
021816101999 sp3n07 N sp3en0 74 SP_SW_SDR_FAIL_RE
021816101999 sp3n07 N sp3en0 75 SP_SW_PRI_TAKOVR_RE

[sp3en0:/]# Eprimary
7       - primary
1       - oncoming primary
5       - primary backup
14      - oncoming primary backup
```

Figure 103. Control Workstation Dialogue

By observing the contents of the /var/adm/SPlogs/css/summlog file we can see that the Worm daemon on sp3n01 was killed at about 16:07 and two entries were recorded into the log file. About 3 minutes later, four additional entries from sp3n07 were also placed into the log file. Note that the snap field value is "N" for all entries, and so the css.snap utility was not required to be executed.

The AIX error log entries on nodes 1 and 7 are shown in Figure 104 on page 247.

```
[sp3en0:/]# dsh -w sp3n01 errpt | more
sp3n01: IDENTIFIER TIMESTAMP  T C RESOURCE_NAME  DESCRIPTION
sp3n01: 89F34F83   0218160799 P S Worm           Switch Fault Service
Daemon Terminated
sp3n01: 5A38E3B0   0218160799 P S Worm           Switch daemon received
SIGTERM
...
...
[sp3en0:/]# dsh -w sp3n07 errpt | more
sp3n07: IDENTIFIER TIMESTAMP  T C RESOURCE_NAME  DESCRIPTION
sp3n07: D70626E9   0218161099 I H Worm           Switch primary node
takeover
sp3n07: EC771C6B   0218161099 T S css            Switch daemon SDR
communications failed
sp3n07: A388E1C3   0218161099 I S Worm           Links not initialized
during Estart
sp3n07: B1531818   0218161099 I S Worm           Nodes not initialized
during Estart
```

Figure 104. Individual Node Error Log Entries

We can see that the entries from the AIX error logs on the nodes have been conveniently recorded in the summlog file, making the initial problem determination process easier.

9.8.2 Switch Log Files

There are numerous log files associated with the switch, some of which can provide essential information for diagnosing and resolving switch problems. It is beyond the scope of this book to provide details about each log file together with enough detail on how they can be read and interpreted. However, we will explore the contents of the out.top file. Refer to *IBM Parallel System Support Programs for AIX: Diagnosis Guide*, GA22-7350*, IBM Parallel System Support Programs for AIX: Messages Reference*, GA22-7352 and *RS/6000 SP: PSSP 2.2 Survival Guide,* SG24-4928 for more detailed information. The css.snap utility (located in /usr/lpp/ssp/css) creates a compressed tar file of all the switch log files and switch configuration information. This file can be forwarded to local IBM Support Centers for further analysis, if required.

The various log files associated with the SP Switch are located as follows:

On the Control Workstation:

- /var/adm/SPlogs/css/Ecommands.log
- /var/adm/SPlogs/css/cssadm.debug

- /var/adm/SPlogs/css/summlog

On the Primary Node ONLY:

- /var/adm/SPlogs/css/css.snap.log
- /var/adm/SPlogs/css/cable_miswire
- /var/adm/SPlogs/css/dist_topology.log
- /var/adm/SPlogs/css/dtbx.trace.failed
- /var/adm/SPlogs/css/flt

On all Nodes:

- /var/adm/SPlogs/css/Ecommands.log
- /var/adm/SPlogs/css/daemon.stderr
- /var/adm/SPlogs/css/daemon.stdout
- /var/adm/SPlogs/css/dtbx.trace
- /var/adm/SPlogs/css/flt
- /var/adm/SPlogs/css/fs_daemon_print.file
- /var/adm/SPlogs/css/logevnt.out
- /var/adm/SPlogs/css/out.top
- /var/adm/SPlogs/css/rc.switch.log
- /var/adm/SPlogs/css/router.log
- /var/adm/SPlogs/css/worm.trace

9.9 The out.top File

The /var/adm/SPlogs/css/out.top file is a dump of the topology file with errors and fault information, created when the switch is initialized. In this file you can find, at a glance, some common wiring and node problems. An extract from this file is shown in Figure 105 on page 249.

Ex.1010 / Page 272 of 572

```
# Node connections in frame L01 to switch 1 in L01
s 15 3  tb3 0 0            E01-S17-BH-J7 to E01-N1
s 15 2  tb3 1 0            E01-S17-BH-J8 to Exx-Nxx   -4 R: device has
been removed from network - faulty (link has been removed from network -
fenced)
s 16 0  tb3 2 0            E01-S17-BH-J26 to Exx-Nxx   -4 R: device has
been removed from network - faulty (link has been removed from network -
fenced)
s 16 1  tb3 3 0            E01-S17-BH-J25 to Exx-Nxx   -4 R: device has
been removed from network - faulty (link has been removed from network -
fenced)
s 15 1  tb3 4 0            E01-S17-BH-J9 to E01-N5
s 15 0  tb3 5 0            E01-S17-BH-J10 to E01-N6
s 16 2  tb3 6 0            E01-S17-BH-J24 to E01-N7
s 16 3  tb3 7 0            E01-S17-BH-J23 to E01-N8
s 14 3  tb3 8 0            E01-S17-BH-J31 to E01-N9
s 14 2  tb3 9 0            E01-S17-BH-J32 to E01-N10
s 17 0  tb3 10 0           E01-S17-BH-J18 to E01-N11
s 17 1  tb3 11 0           E01-S17-BH-J17 to E01-N12
s 14 1  tb3 12 0           E01-S17-BH-J33 to E01-N13
s 14 0  tb3 13 0           E01-S17-BH-J34 to E01-N14
s 17 2  tb3 14 0           E01-S17-BH-J16 to E01-N15
s 17 3  tb3 15 0           E01-S17-BH-J15 to Exx-Nxx   -4 R: device has
been removed from network - faulty (link has been removed from network -
fenced)
```

Figure 105. Extract from out.top File

We discussed earlier in this chapter the naming conventions that are used to interpret the various fields in this file (see 9.4, “Switch Topology File” on page 218). Those conventions still apply. After the logical and physical information is displayed, other switch network information is provided with a return code. This additional information could be an indication of an error, or simply, an observation that there is no node in a slot to receive a connection from the switch board. In this example, there are no nodes in the slots 2, 3, 4 and 16. The likely explanation is that there is a high node in slot 1 (it occupies 4 slots), and a wide node in slot 15 (it occupies 2 slots).

9.10 System Partitioning and the SP Switch

As discussed in 4.2, “Partitioning the SP System” on page 95, system partitioning is a method for organizing the SP into non-overlapping groups of nodes for various purposes, such as testing new software and creating multiple production environments. In this section, we consider the specific aspects of partitioning on the SP Switch.

A system partition is a fairly static entity that contains a specified subset of SP nodes with a consistent software environment. System partitioning lets you divide system resources to make your SP system more efficient and more tailored to your needs while keeping system management simple and

Ex.1010 / Page 273 of 572

efficient. At the most elementary level a system partition is a group of nodes working together.

The control workstation does not belong to any system partition. Instead, it provides a central control mechanism for all partitions that exist on the system.

9.10.1 Considerations for Partitioning

Some basic rules should always be followed when you set up system partitions. These rules impose some limitations on the configurations that can be selected for the partitions.

- Nodes in a partition must not cross the switch chip boundaries. Putting it another way, all nodes connected to the same switch chip must be in the same partition. The nodes have predetermined fixed connectivity with each switch board. They are:

Node slots 1, 2, 5, and 6	connected to chip 5
Node slots 3, 4, 7, and 8	connected to chip 6
Node slots 9, 10, 13, and 14	connected to chip 4
Node slots 11, 12, 15, and 16	connected to chip 7

  For example, in a single-frame, single-switch system, with a high node in slot 1 and 5 (so they are connected to the same switch chip), it would not be possible to have these two nodes in *different* partitions.

- Virtual Shared Disks (VSDs) cannot span partitions.

Figure 106 on page 251 shows how a single-switch system with 16 thin nodes could be partitioned using one of the layouts for the 8-8 configurations.

Ex.1010 / Page 274 of 572

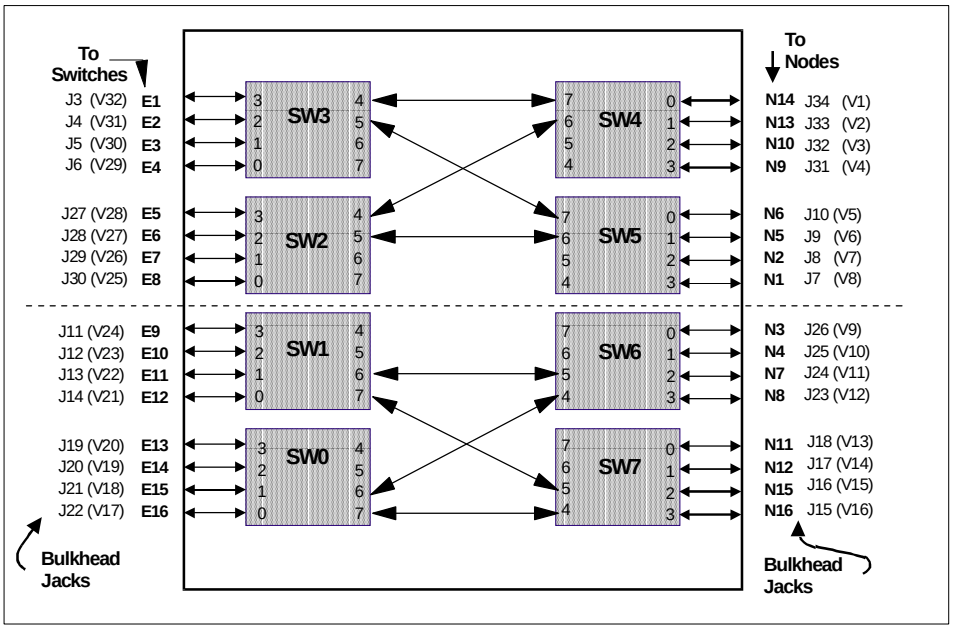


Figure 106. Example of an 8_8 Configuration

Where:

J3 to J34 Bulkhead jacks for data cables

SW0 to SW7 Logical identification of switch chips

- In output files, add 1000 to the logical number.
- The frame number is used to build the logical number. Multiply it by 10.

N1 to N16 Node ports (physical numbering)

E1 to E16 Switch-to-switch ports

9.10.2 System Partitions Directory Structure

The typical directory structure for a single-frame, 16-node system that has two partitions is shown in Figure 107 on page 252.

Ex.1010 / Page 275 of 572

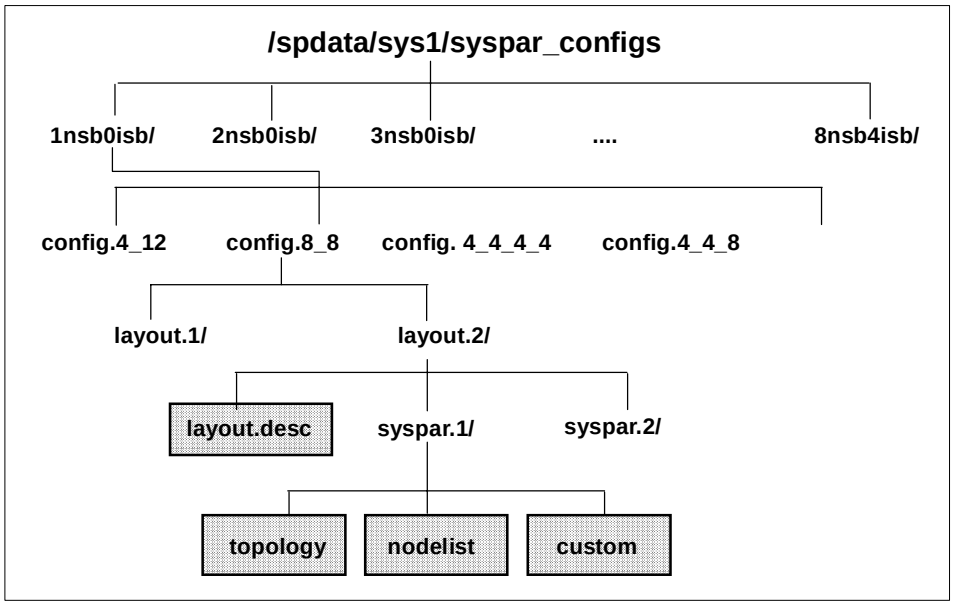


Figure 107. Directory Structure for Topology Files

The /spdata/sys1/syspar_configs directory is at the top-most level and holds all the configuration files and directories that contain information about RS/6000 SP partitions. These are:

- The /spdata/sys1/syspar_configs/<NSBnum>nsb<ISBnum>isb directory is named for the number of Node Switch Boards (NSB) and Intermediate Switch Boards (ISB) used in your configuration. In this example it is /spdata/sys1/syspar_configs/1nsb0isb. Under this directory you will find system-supplied configuration directories.
- The /spdata/sys1/syspar_configs/config.8_8 directory contains the configuration that we have selected in our example. Here config.8_8 means that we have selected a configuration that will make two partitions of 8 nodes each.
- The /spdata/sys1/syspar_configs/config.8_8/layout.3 directory determines the selected layout. This layout determines which nodes are selected for a particular partition. The file layout.desc (also in this directory), describes the actual layout: it lists the switch node numbers of the nodes in each partition.
- /spdata/sys1/syspar_configs/config.8_8/layout.3/syspar.1 and /spdata/sys1/syspar_configs/config.8_8/layout.3/syspar.2 are the syspar directories, one for each partition.

- In each of the syspar.n directories, there are three files:
  - **topology**

    This file contains the necessary topology information for the switch. Sometimes the topology files provided with PSSP do not meet your needs. In such cases, the system partitioning aid can be used to create custom topology files.
  - **nodelist**
    The nodelist file is similar to the layout.desc file, except that it describes the nodes that only belong to its partition.
  - **custom**
    The custom file is created at the time of partition creation. The other files are selected as supplied to match your partitioning requirement. This file contains information on partition name, PSSP level, default mksysb name for the partition, primary node for the partition, backup primary node for the partition, and topology file name.

Part 3. System Management

© Copyright IBM Corp. 1999

Chapter 10. Installation and Configuration

The first and foremost task an administrator has to perform after purchasing an RS/6000 SP is to set it up. Setting up an RS/6000 SP system requires a fair amount of planning. This chapter explains the processes involved in installing and configuring the RS/6000 SP system from the software perspective. Hardware installation is planned with the help of your IBM hardware engineers. They will be able to assist you in developing a satisfactory design.

10.1 Concept of SP Installation

The implementation of an RS/6000 SP system involves two types of installations: the hardware and the software. For the hardware installation, an IBM hardware engineer will perform the initial set up. However, you need to understand how the various hardware components interact with one another in order to understand how the software works. The software installation includes two main components: the control workstation and the processor nodes. The installation of the control workstation plays the primary role. Only when this is in place can the rest of the system (the processor nodes) be installed. Earlier chapters explain how the various PSSP components work together to form the RS/6000 SP.

In order to achieve a successful implementation, you will need to properly design your final setup. Aspects that you will have to consider are:

- Network design
- The physical equipment and its operational software
- Operational environments
- System partitions
- Migration and coexistence on existing systems
- Security and authentication
- Defining user accounts
- Backup procedures

If you are new to the RS/6000 SP, we recommend that you read *IBM RS/6000 SP: Planning, Volume 1, Hardware and Physical Environment*, GA22-7280 and *IBM RS/6000 SP: Planning, Volume 2, Control Workstation and Software Environment*, GA22-7281 for complete information regarding system planning.

© Copyright IBM Corp. 1999

10.1.1 Role of Control Workstation and Boot/Install Servers

The control workstation plays the main role in an RS/6000 SP installation. It is from this system that the rest of the nodes are installed. The control workstation is an RS/6000 system running AIX. In order to perform its role as a control workstation, it has to be loaded with the IBM Parallel System Support Programs for AIX (PSSP) software package. With this software installed, it serves as a point of control for installing, managing, monitoring and maintaining the frames and nodes. In addition to these functions, the control workstation is normally set up as an authentication server.

After installing at least one node, it can configure one or more of the nodes to become boot/install servers. These nodes, serving as boot/install servers, can be used to boot up and install other nodes, offloading from the control workstation this time-consuming and process-hungry task.

10.1.2 NIM Concept

The core of the SP node installation centers around the Network Installation Management (NIM) in AIX.

With NIM, you can manage standalone, diskless, and dataless systems. In a broad sense, an SP node can be considered a set of standalone systems: each node has the capability of booting up on its own because that the RS/6000 SP system is based on a share-nothing architecture. Each node is basically a standalone system configured into the SP frame.

Working together with the System Data Repository (SDR), NIM allows you to install a group of nodes with a common configuration or individually customize to each node's requirements. This helps to keep administrative jobs simpler by having a standard rootvg image and a standard procedure for installing a node. The rootvg volume groups are consistent across nodes (at least when they are newly installed). You also have the option to customize an installation to cater to the specific needs of a given node if it differs from the rest.

As NIM installations utilize the network, the number of machines you can install simultaneously depends on the throughput of your network (namely, Administrative Ethernet). Other factors that can restrict the number of installations at a time are the disk access throughput of the installation servers, and the processor type of your servers.

The control workstation and boot/install servers are considered NIM masters. They provide resources (like files, programs and booting capability) to nodes. Nodes are considered NIM clients, as they are dependent on the masters for

Ex.1010 / Page 282 of 572

services. Boot/install servers are both masters and clients since they serve other nodes but are in turn dependent on the control workstation for resources. Not considering NIM masters outside of the SP complex, the control workstation is configured only as a NIM master. The master and clients make up a NIM environment. Each NIM environment can have only one NIM master. In the case of the control workstation and the boot/install server, only the control workstation is considered the master. In another NIM environment consisting of the boot/install server and its client nodes, only the boot/install server is the master.

A NIM master makes use of the Network File System (NFS) utility to share resources with clients. As such, all resources required by clients must be local file systems on the master.

Attention

Administrators might be tempted to NFS-mount file systems from another machine to the control workstation as /spdata/sys1/install/images to store node images, and as /spdata/sys1/install/<name>/lppsource to store AIX filesets due to disk space constraints. This is strictly not allowed, as NIM will not be able to NFS export these file systems.

The way NIM organizes itself is by object classes, object types and object attributes. Table 8 shows how each NIM object is related.

Table 8. NIM Objects Classification

Object Class	Object Type	Object	Object Attributes
Machines	standalone dataless diskless	machines	TCP/IP hostname hardware address name of network type
Resources	spot lppsource mksysb scripts bosinst_data other resources	file directory	location
Networks	ethernet token ring FDDI ATM	subnet	subnet mask

Ex.1010 / Page 283 of 572

Not all of NIM's functionality is used in the SP complex. Only the minimum to install a standalone system is utilized. Table 9 on page 260 shows the required information used during installation of an SP node.

Table 9. NIM Objects Classification Used By PSSP

Object Class	Object Type	Object	Object Attributes
Machines	standalone	machines	TCP/IP hostname hardware address name of network type
Resources	spot lppsource mksysb scripts bosinst_data	file directory	location
Networks	ethernet	subnet	subnet mask

NIM allows two modes of installation: pull or push. In the pull mode, the client initiates the installation by pulling resources from the master. In the push mode, the master initiates the installation by pushing resources to the client. The RS/6000 SP technology utilizes the pull mode. Nodes request resource like mksysb image, SPOT, AIX source files and so on from the master (by default, the control workstation).

During the initial booting of a node, a bootp request from the node is issued over the network specifying its en0 hardware address. This request reaches the boot/install server or control workstation. The boot/install server or control workstation verifies the node's hardware address against the /etc/bootptab file. If the node is registered in the /etc/bootptab file, the boot/install server or control workstation sends the node's IP address back to the node.

On receiving the IP address, the node's Initial Program Load (IPL) Read-Only Storage (ROS) requests a boot image from the boot/install server or control workstation through tftp. Once the boot image is run, the rootvg gets created, and NIM starts the mksysb image installation.

Upon completing the installation of the mksysb image, NIM invokes the `pssp_script` script to customize the node.

If you need further details about the way NIM works, refer to *AIX Version 4.3 Network Installation Management Guide and Reference*, SC23-2627.

Ex.1010 / Page 284 of 572

10.1.3 Installation Procedure

The installation of the RS/6000 SP system involves the following steps:

1. Install and configure AIX on the control workstation.

   Note that the control workstation must be at the highest AIX and PSSP level for the whole system.

2. Setting up the SDR information for the frames, nodes and switches.
3. Installation of the nodes.

The rest of this chapter describes in detail the procedure to set up the RS/6000 SP system. For further details, refer to *PSSP: Installation and Migration Guide*, GA22-7347.

10.2 Control Workstation Installation

Setting up and preparing the control workstation takes up most of the installation time. This is also the most important part of the installation. All conditions must be properly laid out during the planning session.

In a broad sense, the set up of the control workstation involves the following four steps:

1. Install AIX
2. Set up the AIX environment
3. Set up the PSSP environment
4. Set up SDR information

10.2.1 AIX Installation

The installation of AIX on the control workstation follows the standard AIX installation on any other RS/6000 machine of the same architecture. Shipped together with the system are mksysb tapes and AIX CDs. You can make use of these sources for installing your control workstation. However, if you choose to install with your own image, make sure the AIX level is compatible with the PSSP software and that the latest AIX fixes are applied to your system. Table 10 shows the correct AIX level for each PSSP level.

Table 10. PSSP Levels and Their Corresponding AIX Levels

PSSP Level	AIX Level
PSSP 3.1	AIX 4.3.2
PSSP 2.4	AIX 4.2.1 or AIX 4.3
PSSP 2.3	AIX 4.2.1 or AIX 4.3

Ex.1010 / Page 285 of 572

PSSP Level	AIX Level
PSSP 2.2	AIX 4.1.5 or AIX 4.2.1

10.2.2 Setting Up The AIX Environment

There are many steps to set up the AIX environment.

Ensure the root user PATH environment variable includes /usr/lpp/ssp/bin, /usr/lib/instl, /usr/sbin, /usr/lpp/sp/kerberos/bin. These paths are required, as commands issued during installation reside in these directories.

There must be enough disk space on the control workstation since it is the default boot/install server. Backup images, AIX filesets, and SPOT all require substantial amount of disk space.

Ensure that bos.net (TCP/IP and NFS) is installed on your system. Another pre-requisite file set that comes with AIX 4.3.2 is the perfagent.tools 2.2.32.*. Refer to Table 11 for the required perfagent file sets.

Table 11. Perfagent File Sets

AIX Level	PSSP Level	Required Perfagent File Sets
4.1.5	2.2	perfagent.server 2.1.5.x
4.2.1	2.2	perfagent.server 2.2.1.x or greater, where x is greater than or equal to 2
4.2.1	2.3	perfagent.server 2.2.1.x or greater, where x is greater than or equal to 2
4.2.1	2.4	perfagent.server 2.2.1.x or greater, where x is greater than or equal to 2
4.3.1	2.3	perfagent.server 2.2.31.x
4.3.1	2.4	perfagent.server 2.2.31.x
4.3.2	2.3	perfagent.tools and perfagent.server 2.2.32.x
4.3.2	2.4	perfagent.tools and perfagent.server 2.2.32.x
4.3.2	3.1	perfagent.tools 2.2.32.x

Next you will need to configure the RS-232 tty connection to the frame. Select the port to which the RS-232 frame supervisor cable is connected. The default baud rate of 9600 can be used since hardmon changes it to 19200 when it accesses the line. Each SP frame has one RS-232 line connected, except in HACWS configurations, where two RS-232 lines are attached for

Ex.1010 / Page 286 of 572

high availability. For each S70 attachment two RS-232 lines are required. For each Netfinity server one RS-232 line is required. Each of these RS-232 lines must be configured.

The transmit queue size for the SP administrative ethernet has to be tuned accordingly for better performance. Refer to Table 12 for the various settings.

Table 12. Adapter Type and Transmit Queue Size Settings

Adapter Type	Transmit Queue Size
PCI adapters	256
MCA adapters AIX 4.2.0 or earlier	Maximum possible
MCA adapters AIX 4.2.1 or later	512

After changing the transmit queue size for the appropriate adapter, you can proceed to configure the SP administrative ethernet IP address based on your planning sheet.

The maximum number of processes allowed per user must be increased from the default value of 40 to a recommended value of 256. This is done to accommodate the numerous processes spawned off during the installation.

The default network options have to be tuned for optimal performance over the network. The recommended values are stated in Table 13. To make the changes effective every time the control workstation is rebooted, these values must be set in the /etc/rc.net file. For immediate effect, use the command line. For example, to change thewall value, type:

```
# no -o thewall=16384
```

Table 13. Recommended Network Option Tunables

Parameters	Values
thewall	16384
sb_max	163840
ipforwarding	1
tcp_sendspace	65536
tcp_recvspace	65536
udp_sendspace	32768
udp_recvspace	65536
tcp_mssdflt	1448

Ex.1010 / Page 287 of 572

A directory (preferably a file system) /tftpboot has to be created to store all the NIM boot images. Its size should be about 25 MB for each AIX level supported on the nodes. Having too little storage space in this directory will cause the `setup_server` command to fail.

Next comes the /spdata directory. We recommend that this directory be created as a file system on a volume group of its own because of the amount of disk space it requires. A minimum of 2 GB disk space is required to support one lppsource. Figure 108 shows the directory structure for /spdata.

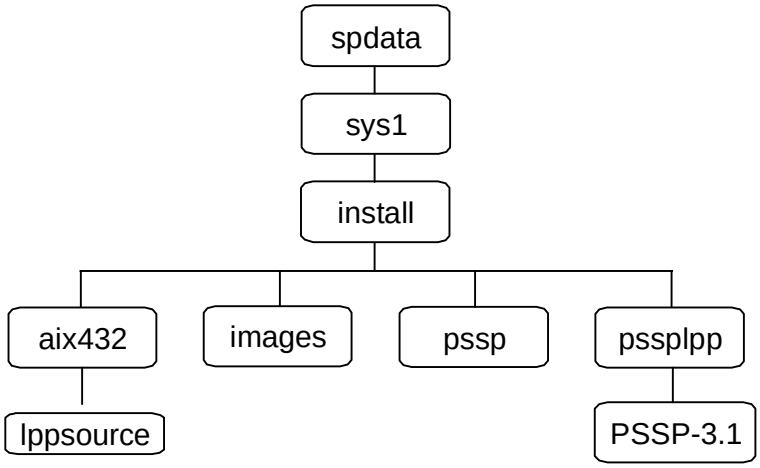


Figure 108. /spdata Directory Structure

Each one of the directories must be created. If you have to install nodes of other AIX levels, you must create the relevant lppsource directories under their own AIX directories. Here, only the aix432 directory and its related lppsource directory are created. If the other AIX levels require different PSSP codes, then the relevant PSSP directories must be created under the pssplpp directory. This shown in Figure 109 on page 265.

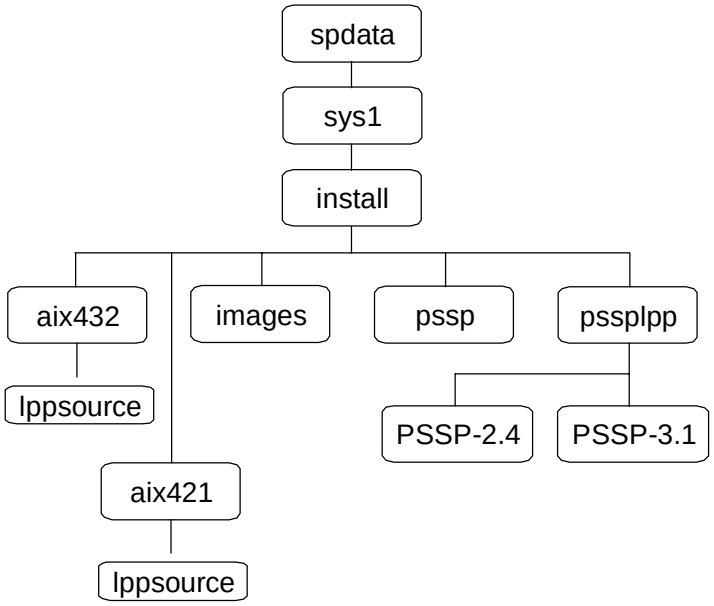


Figure 109. Multiple AIX And PSSP Levels

The /spdata/sys1/install/aix432/lppsource directory contains all the necessary AIX file sets for node installation. We recommend that you download all the AIX file sets. You must use the `bffcreate` command to copy these file sets from the source CDs or tapes to this directory. The command to copy from an AIX CD is as follows:

```
# /usr/sbin/bffcreate -v -d /dev/cd0 -t \
/spdata/sys1/install/aix432/lppsource -X all
```

The SMIT fastpath is:

```
# smitty bffcreate
```

Also required in the lppsource directory are the perfagent.* file sets. The perfagent.server file sets are part of the Performance Aide for AIX (PAIDE) feature of the Performance Toolbox for AIX (PTX). The perfagent.tools file sets are part of the AIX packaging. See Figure 11 on page 262 for the correct level for your installation.

Copy the mksysb image that you want to use for installing your nodes into the /spdata/sys1/install/images directory. Each mksysb must have an equivalent AIX lppsource to support it. You can use the minimum image that is shipped with the RS/6000 SP system, or you can create your own mksysb image. By

creating your own, you can install nodes that have the same file sets requirement instead of individually loading each one of them with the same file sets after the mksysb installation. The recommended method is to install a node with the default image and then install all the other required file sets. Next, take a mksysb image of that node and install the rest of the nodes that have the same requirements. You can do likewise for other groups of nodes with other specific needs.

To download the minimum image from the tape media, type:

```
# installp -a -d /dev/rmt0.1 -X spimg
```

Next, copy the PSSP images from the product tape into the /spdata/sys1/install/pssplpp/PSSP-3.1 directory:

```
# bffcreate -d /dev/rmt0 -t /spdata/sys1/install/pssplpp/PSSP-3.1 -X all
```

The ssp.usr.3.1.0.0 file set must be renamed to pssp.installp. After renaming the file, a new table of contents has to be generated as follows:

```
# mv /spdata/sys1/install/pssplpp/PSSP-3.1/ssp.usr.3.1.0.0
/spdata/sys1/install/pssplpp/PSSP-3.1/pssp.installp

# inutoc /spdata/sys1/install/pssplpp/PSSP-3.1
```

10.2.3 Setting Up The PSSP Environment

After installing AIX and preparing its environment, the PSSP software is installed. On the control workstation, the minimum required file sets are:

- rsct.* (all rsct file sets)
- ssp.basic
- ssp.authent (if control workstation is Kerberos authentication server)
- ssp.clients
- ssp.css (if SP switch is installed)
- ssp.top (if SP switch is installed)
- ssp.ha_topsvcs.compat
- ssp.perlpkg
- ssp.sysctl
- ssp.sysman

Attention

If you are adding dependent nodes, you must also install the ssp.spmgr fileset.

Ex.1010 / Page 290 of 572

The ssp.docs and ssp.resctr.rte file sets are recommended as part of the installed file sets. The ssp.docs file set provides online PSSP publications while the ssp.resctr.rte file set provides the front end interface to all online documentation and resources including SP-related Web site links.

The authentication services you want to implement on your system include RS/6000 SP, AFS, or MIT Kerberos Version 4. You can choose to implement one of them. The `/usr/lpp/ssp/bin/setup_authent` script will initialize the authentication services.

To finish off with the installation of the control workstation, the `install_cw` script is run. This command configures the control workstation as follows:

- It adds the PSSP SMIT panels to the ODM.
- It creates an SP object in the ODM which contains the node_number for the control workstation, which is always 0.
- It calls the script `/usr/lpp/ssp/install/bin/post_process` to do post-installation tasks such as:
  - Starting the hardmon daemon, SDR daemons
  - Calling the setup_logd script to start the monitor's logging daemon
  - Updating the /etc/services and /etc/inittab files.
  - It also ensures that /tftpboot/tuning.cust exists. If the file does not exist, it copies from /usr/lpp/ssp/install/config/tuning.default.
  - Finally, it sets the authentication server attribute in the SDR to reflect the kind of authentication server environment.

To verify that the SDR is properly installed, type:

```
# SDR_test
```

To verify if system monitor is properly installed, type:

```
# spmon_itest
```

10.3 Frame, Node And Switch Installation

After completing the control workstation installation and set up, the next step is to configure the frames, nodes and switches.

10.3.1 Site Environment And Frame Information

SDR information can only be entered on the control workstation. You can use the SMIT panels or the command line. If you are already using the SP Perspectives, there are icons which will lead you to the SMIT panels. You are

Ex.1010 / Page 291 of 572

advised to use the SMIT panels, as they are less prone to errors and more user-friendly. The SMIT fastpath is:

```
# smitty enter_data
```

The first task is to set up the Site Environment Information. Select **Site Environment Information** on the SMIT menu. Figure 110 shows this SMIT screen. Enter the following information:

- Default network install image name (directory path not required)
- NTP information
- Automounter option
- User administration information
- File collection information
- SP accounting information
- AIX lppsource name (directory path not required)



Figure 110. Site Environment Information SMIT Panel

Next, enter the SP Frame Information as shown in Figure 111 on page 269. In this panel, specify the starting frame number, the frame count (the number of consecutive frames), the starting frame tty port and whether you want to re-initialize the SDR. Do not enter non-SP frames in this panel. The re-initialization of the SDR is required only when the last set of frames information is entered. Take note that if your tty numbers do not run consecutively, you need to enter the frames information in separate sessions.



Figure 111. SP Frame Information SMIT Panel

If you have non-SP frames attached, enter their information in the Non-SP Frame Information SMIT panel as shown in Figure 112 on page 270. If the non-SP frame is an S70 or S7A, make sure that the starting frame tty port you specify is the one connected to the operator panel on the server. The s1 tty port field is the tty connected to the serial port on the server. The starting switch port number is the switch node number. The frame hardware protocol is where you specify if the non-SP frame is an S70/S7A server or a Netfinity server. For an S70/S7A server, select SAMI; for a Netfinity server, select SLIM. Re-initialize the SDR if the information you entered is for the last set of frames.

Ex.1010 / Page 293 of 572



Figure 112. Non-SP Frame Information SMIT Panel

At this point, your frames must already be powered up. Verify that the System Monitor and Perspectives have been correctly installed. The SMIT fastpath is:

```
# smitty SP_verify
```

Select the System Monitor Configuration option. Figure 113 on page 271 shows the output if everything is installed properly. If it fails, check that your tty configuration is correct. Also, ensure that the physical RS-232 lines are properly connected. Delete the frame information and reconfigure again. Make sure that you do not remove or change the RS-232 connections before you delete the frame or you will end up having to hack the SDR to salvage the situation. Other causes of error include Kerberos authentication problems, SDR or hardmon daemons not running and so on. Rectify the errors before proceeding.

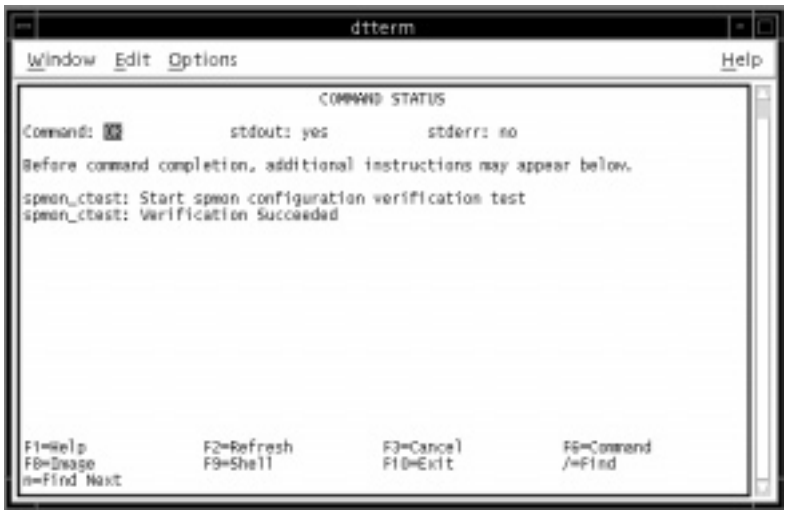


Figure 113. Successful Verification With spmon_ctest

Verify the frame information using the following command:

```
# spmon -d
```

If any error is found, check the RS-232 cables. You should not have this problem, though, as the previous test will have detected it.

Next, verify the supervisor microcode of your system. The SMIT fastpath is:

```
# smitty supervisor
```

Figure 114 on page 272 shows the result of selecting the List Status of Supervisors (Report Form) option. Any item that has the status Upgrade needs the microcode updated. Choose the **Update *ALL* Supervisors That Require Action (Use Most Current Level)** option to update microcode on all the necessary supervisor cards.

Ex.1010 / Page 295 of 572

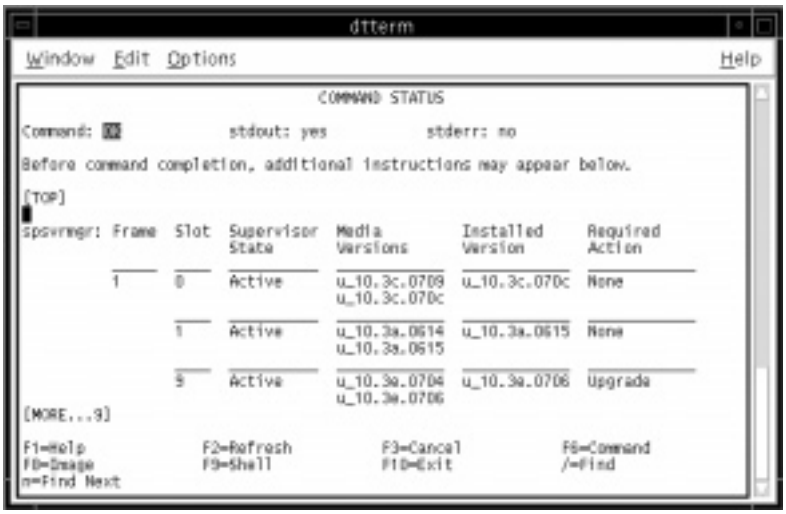


Figure 114. Supervisor Microcode Status

10.3.2 Node Database Information

The next information required is the node database information. The fastpath for this SMIT menu is:

```
# smitty node_data
```

Select the SP Ethernet Information. Provide the information required: the node you are installing, the starting node's en0 host name or IP address, the netmask, default route hostname or IP address, the type of ethernet and its characteristics and whether you want to skip IP address for unused slots. When specifying the starting node's en0 hostname, make sure the hostname is fully qualified. To be on the safe side, use the IP address. Ensure all IP addresses and hostnames are resolvable. You can issue the following command to ensure the correct hostname is specified:

```
# host <node's en0 IP address>
```

If you intend to use the node's slot number as a reference for assigning an IP address, set the Skip IP Address for Unused Slots to yes. This will skip the next sequential IP addresses when it encounters Wide or High nodes. Figure 115 on page 273 shows the SMIT panel available for setting up SP Ethernet addresses.



Figure 115. Setting Up The SP Ethernet Information

In order to establish bootp communication between the control workstation and the nodes, the hardware addresses of the nodes must be made known to the control workstation in the /etc/bootptab file. Select the **Get Hardware Ethernet Addresses** menu. Figure 116 shows the SMIT panel available for getting Ethernet hardware addresses.



Figure 116. Acquiring The SP Ethernet Hardware Addresses

However, if you already have the hardware addresses available, enter them into the /etc/bootptab.info file. This speeds up the installation process since acquiring the hardware addresses can take up a significant amount of time. We recommend that you let the system acquire the hardware addresses. The small amount of time spent doing so will probably save time later by avoiding corrections if the wrong hardware addresses are specified in the /etc/bootptab.info file. The format of the /etc/bootptab.info file is:

```
<node number> <en0 hardware address>
<node number> <en0 hardware address>
```

Attention

Acquiring hardware addresses for nodes will power those nodes down. Do not use this step on nodes running in a production environment.

If you have a switch or other network adapters (ethernet, FDDI, token ring) in your system and you want them to be configured during installation, you need to perform the this step. To do so, select the Additional Adapter Information menu. To configure the SP Switch adapter, use css0 for the Adapter Name. Specify the IP addresses and netmask just like for the en0 adapter.

Attention

- To skip IP addresses for unused slots, do not use the switch node numbers for css0 IP addresses.
- If you do not use switch node numbers for css0 IP addresses, you must ensure ARP is enabled.

In addition, you can specify additional IP addresses for the adapter if you have IP aliasing. Figure 117 on page 275 shows the SMIT panel available for setting up additional network adapters.

Ex.1010 / Page 298 of 572



Figure 117. Setting Up Additional Adapter Information

Next, configure the default hostname of the nodes. By default, the en0's long hostname is used. You can use another adapter's hostname or change to short hostname. Figure 118 shows the SMIT panel for setting up hostnames.



Figure 118. Changing Default Hostname Information

The next step creates the appropriate authorization files for the use of remote commands. The available methods are k4 (Kerberos Version 4) and std

(Standard AIX). k4 must be selected if std is an optional choice. We recommend that you enable all if you are not sure. The SMIT fastpath is:

```
# smitty spauth_config
```

Go to the Select Authorization Methods menu. Indicate the system partition name and the methods. Figure 119 shows the SMIT panel for setting up authorization methods.



Figure 119. Selecting The Authorization Method

Next, enable the selected authentication methods for use in conjunction with System Management tasks. The default is to enable authentication on all nodes as well as the control workstation if the Force change on nodes option is set to yes, it will force the authentication method to change to whatever you set, even though the information in the node may be the same. The available methods are k5 (Kerberos Version 5), k4 and std; k4 is required while k5 and std are optional. If you intend to use ftp, rlogin and telnet commands, k5 or std must be enabled as well. It is recommended that you enable all. Figure 120 on page 277 shows the SMIT panel available for enabling the authentication methods.



Figure 120. Enabling Authentication Methods

If you have a dependent node, add it at this point. First, ensure that you have the ssp.spmgr fileset installed and that the UDP port 162 it uses for SNMP traffic does not clash with other applications that use the same port. If a conflict arises, modify the spmgrd-trap entry in /etc/services to use another port number.

To define the dependent node, use the SMIT fastpath:

```
# smitty enter_extnode
```

The required entries are administrative hostname, SNMP community name, extension node identifier, resolvable SNMP agent hostname, unused node number. Keep the administrative hostname and SNMP agent hostname the same to avoid confusion. Figure 121 on page 278 shows the SMIT panel available for defining dependent node information.



Figure 121. Defining Dependent Node Information

After adding the dependent node information, each node adapter for the dependent node must be defined. The SMIT fastpath is:

```
# smitty enter_extadapter
```

Specify the network address, netmask and the node number as shown in Figure 122 on page 278.



Figure 122. Defining Dependent Node Adapter Information

Ex.1010 / Page 302 of 572

On the dependent node, you can choose to define the dependent node and dependent node adapter. You must configure the SNMP agent on the dependent node to communicate with the dependent node SNMP manager on the control workstation. Next, install the SP Switch Router and all IP interfaces connected to it.

At this point, the system partition-sensitive subsystems like hats, hags, haem and hr must be added and started. To do so, issue the following command:

```
# syspar_ctrl -A
```

Verify that all the system partition-sensitive subsystems have been properly added by issuing the following command:

```
# syspar_ctrl -E
```

Check that the subsystems are active with the following command:

```
# lssrc -a | grep <partition_name>
```

You may have to wait a few minutes before some of the subsystems become active.

The volume group information is required next. Go back to the Node Database Information SMIT menu and select the **Create Volume Group Information** option. In this screen, specify the node you are installing, the volume group name, the target disk, whether you want mirroring, the boot/install server, the network install image name, the lppsource name, the PSSP code version and whether you want to turn on quorum. Make sure that the image you use corresponds to the correct lppsource name. Mismatch of AIX levels can cause unpredictable results. Ensure also the PSSP level is supported for that level of AIX. Figure 123 on page 280 shows the SMIT panel for defining alternate volume groups.

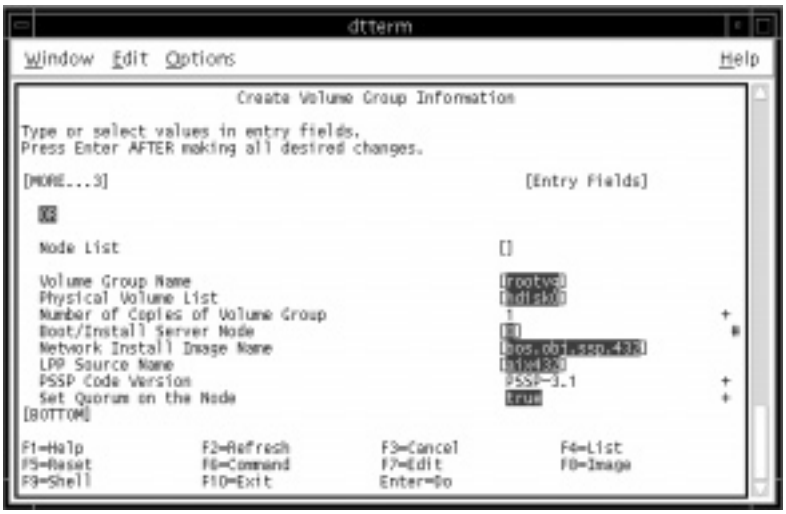


Figure 123. Setting Up Volume Group Information

With PSSP 3.1, you can install the rootvg on external SSA or SCSI disks. Refer to Table 14 for a list of supported SSA boot devices and Table 15 on page 281 for a list of supported SCSI boot devices.

Table 14. Supported External SSA Boot Devices

Node Type	External Boot Supported?	SSA Adapter Feature Codes			
		6214	6216	6217	6219
62 MHz Thin	N				
66 MHz Wide	N				
66 MHz Thin 2	N				
77 MHz Wide	Y	X	X	X	X
112 MHz 604 High	Y	X	X	X	X
120 MHz Thin	Y	X	X	X	X
135 MHz Wide	Y	X	X	X	X
200 MHz 604e High	Y	X	X	X	X
160 MHz Thin	Y	X	X	X	X
332 MHz SMP Thin	N				
332 MHz SMP Wide	N				

Node Type	External Boot Supported?	SSA Adapter Feature Codes			
		6214	6216	6217	6219
POWER3 Thin	N				
POWER3 Wide	N				

The supported external SSA subsystems are:

- 7133-010
- 7133-020
- 7133-500
- 7133-600

Table 15. Supported External SCSI Boot Devices

Node Type	External Boot Supported?	SCSI Adapter Feature Codes			
		2412	2416	6207	6209
62 MHz Thin	Y	X	X		
66 MHz Wide	Y	X	X		
66 MHz Thin 2	Y	X	X		
77 MHz Wide	Y	X	X		
112 MHz 604 High	Y	X	X		
120 MHz Thin	Y	X	X		
135 MHz Wide	Y	X	X		
200 MHz 604e High	Y	X	X		
160 MHz Thin	Y	X	X		
332 MHz SMP Thin	Y			X	X
332 MHz SMP Wide	Y			X	X
200 MHz POWER3 Thin	Y			X	X
200 MHz POWER3 Wide	Y			X	X

The supported external SCSI disk subsystems are:

- 7027-HSD IBM High Capacity Drawer
- 7131-105 SCSI Multi-Storage Tower

When specifying the Physical Volume List (target disk for rootvg), three types of format can be used. The first format specifies the device name such as hdisk0 or hdisk1. The second format specifies the hardware location of the disk (SCSI only), for example 00-00-00-0,0. When specifying more than one disk using this format, separate the location addresses by colons:

00-00-00-0,0:00-00-00-1,0

The third format specifies the parent-connwhere attribute (SSA only), for example ssar//0004AC5052B500D. To indicate more that one disk, separate the disks by colons:

ssar//0004AC5052B500D:ssar//0004AC5150BA00D

You can always go back to make changes to this volume group by selecting the **Change Volume Group Information** option.

Before continuing with the installation, perform a check on all the information you have entered into the SDR using the `splstdata`. command.

To list the site environment settings, use the following command:

```
# splstdata -e
List Site Environment Database Information

attribute               value
-----------------------------------------
control_workstation     sp4en0
cw_ipaddrs              9.12.0.4:192.168.4.140:
install_image           bos.obj.ssp.432
remove_image            false
primary_node            1
ntp_config              consensus
ntp_server              ""
ntp_version             3
amd_config              false
print_config            false
print_id                ""
usermgmt_config         true
passwd_file             /etc/passwd
passwd_file_loc         sp4en0
homedir_server          sp4en0
homedir_path            /home/sp4en0
filecoll_config         true
supman_uid              102
supfilesrv_port         8431
spacct_enable           true
spacct_actnode_thresh   80
spacct_excluse_enable   false
```

```
acct_master             0
cw_has_usr_clients      false
code_version            PSSP-3.1
layout_dir              ""
authent_server          ssp
backup_cw               ""
ipaddrs_bucw            ""
active_cw               ""
sec_master              ""
cds_server              ""
cell_name               ""
cw_lppsource_name       aix432
cw_dcehostname          ""
```

To list the frame information, use the following command:

```
# splstdata -f
```

To list the node information, use the following command:

```
# splstdata -n
```

To list the adapter information, use the following command:

```
# splstdata -a
```

To list the boot/install information, use the following command:

```
# splstdata -b
```

To list the switch information, use the following command:

```
# splstdata -s
```

To list the node information for dependent nodes use the following command:

```
# splstnodes -t dependent node_number reliable_hostname \
management_agent_hostname extension_node_identifier snmp_community_name
```

To list the node adapter information, use the following command:

```
# splstadapters -t dependent node_number netaddr netmask
```

You can customize your nodes during the installation process. There are three files where you can specify your customization requirements. The three files are:

- **tuning.cust -** It is used to set the initial network tuning parameters. This file is called by pssp_script after installing the node. It must be placed in the /tftpboot directory for it to be effective. If during the installation process, the boot/install server cannot locate this file, the default

/usr/lpp/ssp/install/config/tuning.default file is copied to /tftpboot/tuning.cust. To help you with an initial setting, IBM provides three sets of values for three types of environment:

  - /usr/lpp/ssp/install/config/tuning.commercial for typical commercial environment.
  - /usr/lpp/ssp/install/tuning.scientific for typical engineering/scientific environment. This file must not be used for tuning the S70/S7A servers.
  - /usr/lpp/ssp/install/config/tuning.development for development environment.

- **script.cust** - This file is also called by pssp_script just after the node is installed but before it reboots for the first time. This script is run in a limited environment, so many functions are not available yet. You can copy the /usr/lpp/ssp/samples/script.cust to /tftpboot as an initial reference.
- **firstboot.cust** - This file is called the first time a node is rebooted after a network installation. A sample copy of this file resides in /usr/lpp/ssp/samples/firstboot.cust. Copy this sample file to /tftpboot/firstboot.cust and modify the contents to suite your requirement.

The next step will configure the control workstation as a boot/install server. Prior to performing this step, ensure that the /usr file system or its related directories are not NFS-exported. The /spdata/sys1/install/images directory also must not be NFS-exported.

The command to set up the control workstation as a boot/install server is `setup_server`. The first time `setup_server` is run, it takes a considerable amount of time.

The `setup_server` Perl script is run on the control workstation when explicitly called, and on every node when they reboot. On the control workstation, or on a node set to be a boot/install server (stated in the SDR), this script configures it as a boot/install server. This command requires a ticket-granting-ticket to run. It performs the following functions:

- Defines the boot/install server as a Network Installation Management (NIM) master
- Defines the resources needed for the NIM clients
- Defines each node that this server installs as a NIM client
- Allocates the NIM resources necessary for each NIM client
- Creates the node.install_info file containing netinstall information
- Creates the node.config_info file containing node-specific configuration information be used during network boot

- Creates server key files containing the service keys for the nodes
- Copies the install images from the control workstation for nodes which are boot/install servers

Creation of the Network Installation Management (NIM) lppsource resource on a boot/install server will result in `setup_server` creating a lock in the lppsource directory on the control workstation. This lock is created by the `mknimres` command when the `setup_server` command calls it.

The general flow for `setup_server` is listed in Figure 124

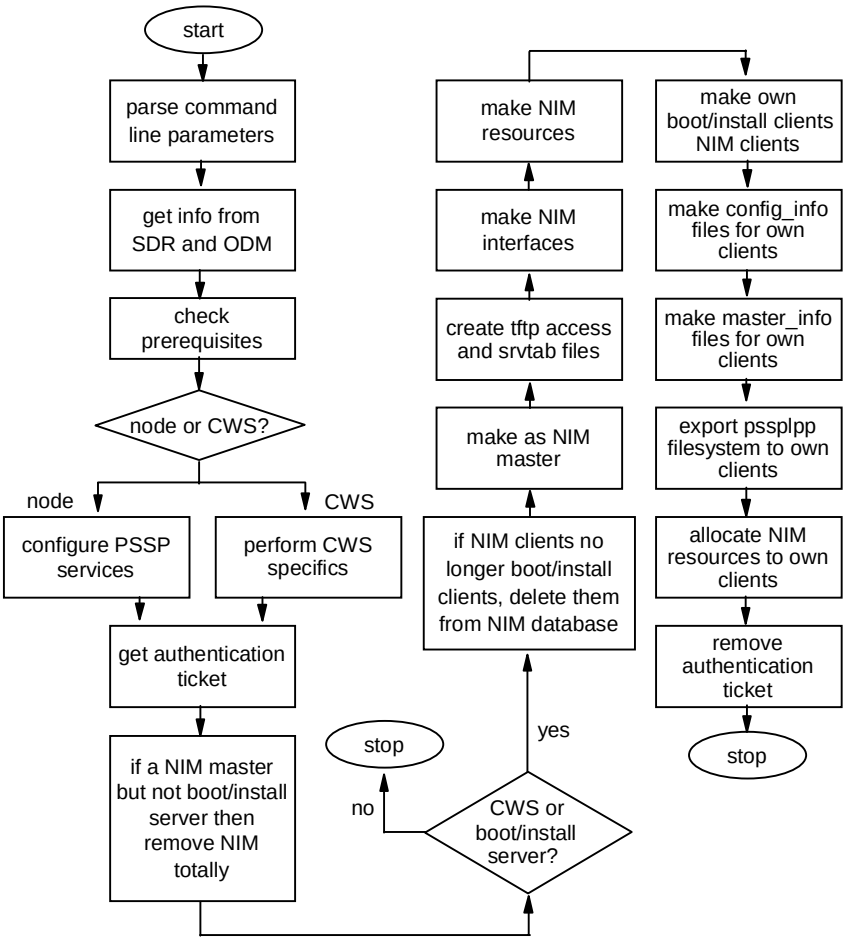


Figure 124. General Flow Of setup_server Command

Ex.1010 / Page 309 of 572

After running setup_server, do a check to see if the System Management tools are properly set up using the following command:

```
# SYSMAN_test
```

10.3.3 Setting Up The Switch

If you have a switch, you must perform the next few steps. Depending on the number of switches, you need to select a correct switch topology file for your system. Refer to 9.4, “Switch Topology File” on page 218 for a description on this file and how to select the proper file. After determining the correct switch topology file to use, annotate the file. The SMIT fastpath for Eannotator is:

```
# smitty annotator
```

Enter the topology file name, specifying the full directory path. Also enter a fully-qualified name of a file in which to store the annotated topology file. Store the annotated file in the SDR. If the system cannot find a /etc/SP/expected.top file, it will read from the SDR to get the required information.



Figure 125. Annotating A Switch Topology File

The primary and primary backup nodes will already be defined. Verify this by running the Eprimary command. You will see an output like this:

```
1       - primary
1       - oncoming primary
```

```
15      - primary backup
15      - oncoming primary backup
```

The nodes assigned may be different in your environment, but all four roles are defined.

You must next set the switch clock source. Refer to 9.7, “Switch Clocks” on page 238 for details on selecting the correct Eclock topology file. To initialize the clock setting in SMIT, use the fastpath:

```
# smitty chclock_src
```

Important

If you have a running switch network, this command will bring the whole switch network down. The moment you select the topology file, it executes the `Eclock` command immediately. There is no warning message given.

You can partition your system now or you can do it after the installation. Refer to *PSSP: Administration Guide*, SA22-7348 for details on partitioning the system.

10.3.4 Node Installation

With all settings in place, the nodes can be installed. If your configuration has many nodes, we recommend that you perform installation of the nodes in batches of eight nodes or less at a time due to the performance issue. Perform installation on one node and assess the outcome. If the installation succeeds, you can then perform installation for the rest of the nodes. Install the node by netbooting it (a term used for network boot). Use the SP Hardware Perspective to perform the netboot function. From the Nodes pane, select the node to install by clicking it once. Select **Network Boot** from the Action menu bar. Click on the **Apply** button to perform the netboot function. Figure 126 on page 288 shows how this is done.

10.3.4.1 Network Boot

When netboot is performed, the nodecond command is issued. This command checks for the architecture of the node and invokes the appropriate scripts. If the node is an MCA node, it calls the /usr/lpp/ssp/bin/nodecond_mca script. If the node is a CHRP node, it calls the /usr/lpp/ssp/bin/nodecond_chrp script.

Ex.1010 / Page 311 of 572

The nodecond_chrp script gets information from the SDR on the network type and attributes. Next, a serial port is opened via the `s1term` command. The node then gets powered off and then powered on again. When the RS/6000 logo appears, the script obtains the network address and performs a network boot by sending bootp packets to the node. When all these steps are done, NIM takes over with the installation process.

The nodecond_mca script does things a little differently. It first gets the network type and attributes from the SDR. It determines the node type to see whether it is a Thin, Wide or High node. Next, it powers off the node and opens up a serial port. If the node type is either Thin or Wide, it sets the key mode to secure. Next, it initiates a hmmon process to monitor the LED status.

The node is then powered on. When the LED reaches 200, the key mode is changed to service. On detecting an LED of 262, it lets the node know that the serial port is available. The IP address is determined and the network boot proceeds. NIM does the rest of the installation. For the High node, the key mode is switched to service. The BUMP processor is awaken and is set up. The node is then powered on. The script will then set the node up for network boot. The IP address is determined and the network boot proceeds. NIM takes over the rest of the installation.

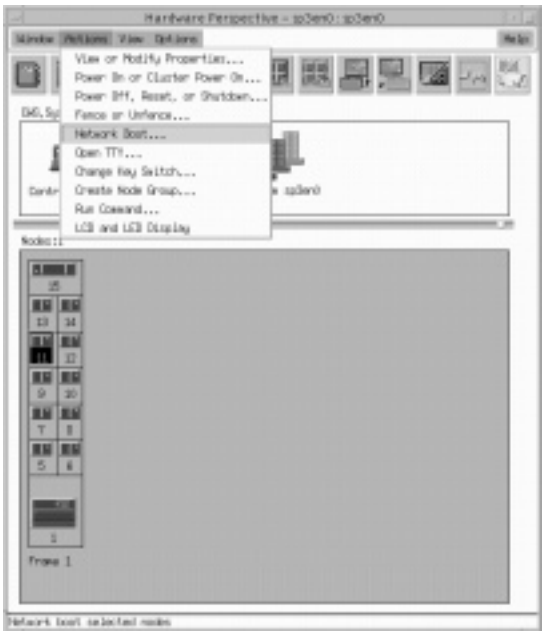


Figure 126. Network Boot Option

10.3.4.2 Customization

Once NIM finishes installing the AIX software on the node, and before the node is rebooted, it invokes the customization script `pssp_script`. This script performs the following functions:

- Creates the necessary directories /var/adm/SPlogs/sysman and /tftpboot
- Creates the log files to record all activities that occur during the customization phase
- Sets up environment, for example getting the Ethernet (en0) hardware address and node number for both the node as well as the boot/install server
- Configures the node's ODM without including the adapters at this point.
- Creates the /etc/ssp files
- Updates the /etc/hosts file with the latest information
- Transfers the SDR_dest_info, script.cust, tuning.cust, spfbcheck and psspfb_script files from the control workstation
- Sets up the Kerberos information
- Updates the /etc/inittab file to include setting up of default route and en0 is brought up.
- Adds entries to the /etc/inittab file to include the spfbcheck and psspfb_script scripts
- Installs all necessary PSSP files and adds the switch adapter information (if available) to the ODM
- Creates the dump device
- Initiates mirroring (if specified)
- Calls the `tuning.cust` script to tune the network parameters
- Calls the `script.cust` script

When `script.cust` finishes, the `psspfb_script` script is initiated. When everything is done, the `pssp_script` script exits. The first time a node is rebooted after installation, the `/etc/firstboot` script is called by the `fbcheck` script. The `/etc/firstboot` script contains information you specified earlier in the firstboot.cust file. The `fbcheck` script next renames the `/etc/firstboot` script so that it will not be executed the next time the node reboots.

Once the node is installed and set up, run the SYSMAN_test command again to verify that System Management tools are properly installed on the node. When this verification check succeeds, the rest of the nodes can be installed.

If you have a switch installed, you can start the switch network by issuing the `Estart` command. Run the `CSS_test` command to test the integrity of the switch. Verify that the host_responds and switch_responds are up and running. You can use the SP Hardware Perspectives to monitor this, or use the following command:

```
# spmon -d -G
```

10.3.4.3 Optional Tasks

The nodes are now installed. You may choose to perform additional tasks like mirroring the rootvg (if not already done), or install another rootvg onto another disk (if you have more than one disk drive).

The procedure to create mirroring after the rootvg is installed is simplified by the PSSP codes; see Figure 127. To create a mirrored copy of the rootvg from hdisk0 to hdisk1, use the SMIT fastpath:

```
# smitty changevg_dialog
```

Specify the node on which you want mirroring to be performed. Enter rootvg as the Volume Group Name and specify the hard disks that rootvg will occupy (including hdisk0). Change the number of copies to two or three, depending on the mirroring copies you want.



Figure 127. Changing rootvg For Mirroring

After changing the rootvg characteristics, begin the mirroring process using the SMIT fastpath:

```
# smitty start_mirroring
```

Select the node and use Forced Extending the Volume Group in case the hard disk contains unwanted volume group information; see Figure 128. The mirroring process starts and takes about 30 minutes or more depending on the size of your volume group.



Figure 128. Initiate Mirroring

You can choose to install an alternate rootvg on another hard disk (maybe for testing purposes). You need to create the new volume group, naming it anything except rootvg (an example is rootvg1). The installation procedure is the same as for rootvg installation. With two rootvg residing on the two hard disks, you need to know how to switch them back and forth. The `spbootlist` command assists you in performing this task.

You will first change the Volume Group Information to the volume group from which you want to boot up. Use the following command if you want to use command line:

```
# spbootins -l <node number> -c <volume group name> -s no
```

Next, use the `spbootlist` command to change to bootlist on the node as follows:

```
# spbootlist -l <node number>
```

Perform a check on the bootlist of the node using the following command:

```
# dsh -w <node name> bootlist -m normal -o
```

If the change is correct, reboot the node. Your node will boot up from the rootvg you specified.

Ex.1010 / Page 316 of 572

Chapter 11. System Monitoring

In order to successfully manage and administer an RS/6000 SP complex, the system administrator needs information. The type of information you require can be summarized in the responses to a series of hypothetical questions:

- Which nodes are up and available?
- How can I minimize application downtime to my users?
- Are any nodes experiencing environmental stresses?
- Is the SP Switch available to dependent applications?
- My manager wants to charge individual departments for their use of system resources — how can I generate the information he requires?
- A node has just failed on the switch network — how do I collect data to assist in problem determination?

The information presented in this chapter will help you answer these questions and guide you to other resources that will assist in system management activities. We will be covering the following topics in this chapter:

- SP Perspectives – the graphical user interface for system management on the RS/6000 SP
- SP Tuning and Performance Monitoring
- SP Accounting
- Problem Management

11.1 SP Perspectives

Scalable POWERparallel Perspectives for AIX (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.

By simply selecting an SP system object (a managed SP system resource such as a frame, node, or switch) by clicking on it with the mouse, you can select an action to perform on that object from the menu bar or tool bar. SP Perspectives uses this pattern of selecting an object, and then selecting an action to accomplish numerous system management tasks.

The AIX command `perspectives` starts the Launch Pad, from which you can launch the following applications:

- Hardware Perspective, for monitoring and controlling hardware

© Copyright IBM Corp. 1999

- Event Perspective, for managing system events and taking actions when events occur
- IBM Virtual Shared Disk Perspective, for managing shared disks
- The System Partitioning Aid
- The Performance Monitor Perspective
- access to other miscellaneous applications such as a set of pre-defined SP Perspectives applications, SMIT, the partition-sensitive subsystems command `syspar_ctrl`, and the SP Resource Center which gains access to the SP documentation through a Web browser interface.

The Launch Pad is shown in Figure 129.

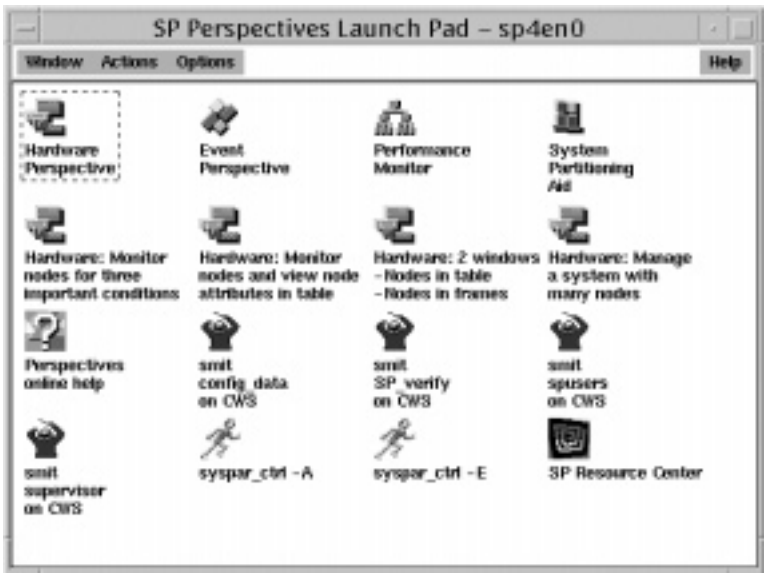


Figure 129. SP Perspectives Launch Pad

You can run the individual applications outside of the Launch Pad. The full pathnames of the individual applications are shown in Table 16.

Table 16. SP Perspective Application Pathnames

SP Perspective Application	Pathname
Hardware Perspective	/usr/lpp/ssp/bin/sphardware
Event Perspective	/usr/lpp/ssp/bin/spevent

SP Perspective Application	Pathname
IBM Virtual Shared Disk Perspective	/usr/lpp/ssp/bin/spvsd
Performance Monitor	/usr/lpp/ssp/bin/spperfmon
System Partitioning Aid	/usr/lpp/ssp/bin/spsyspar
SP Resource Center	/usr/lpp/ssp/bin/resource_center

In previous releases of PSSP, another graphical interface (the original spmon GUI) was available and this was invoked by the `spmon -g` command. With the availability of PSSP v3.1, the -g flag is removed and the functionality that the original spmon graphical monitor provided is now available in SP Perspectives. (All other flags for spmon can still be used. For example, you are still able to issue the command `spmon -G -d`.)

A comprehensive review of SP Perspectives applicable to PSSP v3.1 can be found in the redbook *SP Perspectives: A New View of Your SP System*, SG24-5180. Refer to this book for extensive details on configuring and customizing the individual SP Perspective applications.

11.1.1 Hardware Perspective

The hardware perspective has two functions:

- Controlling hardware, for example, selecting a node, or several nodes, and performing an action on them such as power on/off, fence/unfence or network boot.
- Monitoring hardware, for example, opening a window which monitors important system conditions such as host_responds, switch_responds and individual node power LEDs.

Before we review these functions and how to perform them, we will look at the structure of the Hardware Perspective window as it appears when it is run either from the command line with the `sphardware` command or by double clicking the top leftmost icon **Hardware Perspective** on the launch pad.

The hardware perspective uses a concept of system objects. The system objects are defined as:

- Control workstation
- System
- System Partitions
- Nodes
- Frames and Switches

- Node Groups

These objects are displayed by default as icons, which are placed inside panes in the perspective window (icon view).

The hardware perspective allows four different kinds of pane. It refers to them as:

- CWS, System and Syspars (contains the control workstation, system and system partition objects)
- Nodes (contains node objects)
- Frames and Switches (contains frame and switch objects)
- Node Groups (contains node group objects)

Additional information on the features of the Hardware Perspective can be found in 5.3.2, “SP Perspectives” on page 119.

11.1.2 Event Perspective

Event Management is a distributed subsystem, a component of the RS/6000 Cluster Technology (RSCT). It matches 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.

More detailed information about Event Management can be found in 8.6, “Event Management (EM)” on page 200. In that section we focus on the use of the Event Perspective to define and take action on events.

The Event Perspective allows you to define and manage event definitions within a system partition. In this sense it is not a GUI for displaying information; rather, it allows you to define monitors and triggers for other perspectives to display.

An *event definition* allows you to specify under what condition the event occurs and what actions to take in response. Using the Event Perspective, you can:

- Create an event definition
- Register or unregister an event definition
- View or modify an existing event definition
- Create a new condition

An event is triggered when a boolean expression involving a resource evaluates to true. A *resource* can be any entity in the system that can be observed, for examples processors, disk drives, memory, adapters, database

applications, processes, and file systems. A more detailed description of the Event Perspective, along with examples including defining and monitoring events, is given in *RS/6000 SP Monitoring: Keeping it Alive*, SG24-4873.

11.1.2.1 Getting Started

To use the event perspective to register event definitions with the problem management system, you must add your kerberos principal to the problem management access control list file /etc/sysctl.pman.acl. If you do not have an entry in this file, the Event Perspective will issue a warning as it initializes and certain actions cannot be performed. The file must exist on all nodes where an event definition is registered to a principal other than root.admin. A sample file is shown here.

```
#acl# /etc/sysctl.pman.acl
#
# These are the kerberos principals for the users that can configure
# Problem Management on this node.  They must be of the form as indicated
# in the commented out records below.  The pound sign (#) is the comment
# character, and the underscore (_) is part of the "_PRINCIPAL" keyword,
# so do not delete the underscore.

#
_PRINCIPAL root.admin@MSC.ITSO.IBM.COM
```

You can initialize the Event Perspective from the launch pad, or alternatively from the command line using the spevent command. The main window of the Event Perspective is shown in Figure 130 on page 298.

Ex.1010 / Page 321 of 572

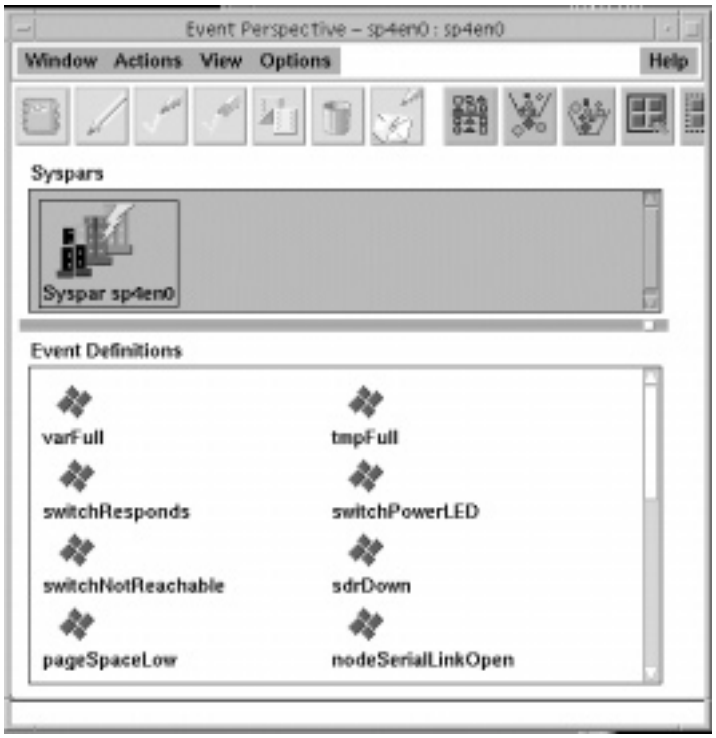


Figure 130. Event Perspective - Main Window

The bottom pane of the initial event perspective window shows some of the 19 pre-defined Event Definitions that are displayed when the Event Perspective is started. These definitions are summarized in Table 17.

Table 17. Pre-Defined Event Definitions

Event Definition Name	Event Definition Description
LCDhasMessage	The node's LED or LCD has a message.
errLog	A permanent error entry is added to the error log.
fileSystems	Monitor percentage usage of file systems.
frameControllerNotResponding	The frame controller is not responding.
framePowerOff	Frame power has been turned off.
hostResponds	The node is not responding.
keyNotNormal	Key mode switch is not in Normal position.

Event Definition Name	Event Definition Description
nodeEnvProblem	Environment LED is on; h/w problem detected.
nodeNotReachable	GS not able to communicate with node.
nodePowerDown	Node power is off.
nodePowerLED	Node power is off when powerLED is not 1.
nodeSerialLinkOpen	Serial link to the node (via TTY) is open.
pageSpaceLow	Node paging space use is greater than 85%.
sdrDown	SDR daemon has died.
switchNotReachable	Switch adapter not up on IP, or node is isolated.
switchPowerLED	Switch power is off when powerLED is not 1.
switchResponds	Switch adapter not responding, or node is isolated.
tmpFull	/tmp (LV=hd3) is running out of space.
varFull	/var (LV=hd9var) is running out of space.

By default, none of these events are active. You can start event monitoring using one or more of these defined events by using the following procedure:

1. Choose the event you wish to monitor by selecting its icon in the Events Definition pane. You can choose more than one event by pressing the left-CTRL button at the same time as you select the event(s).
2. Go to the Menu bar, and select **Actions->Register**.
3. The icons for the selected events will change from all-grey to colored. The definitions for the colors assigned to icons are shown in Figure 131 on page 300.

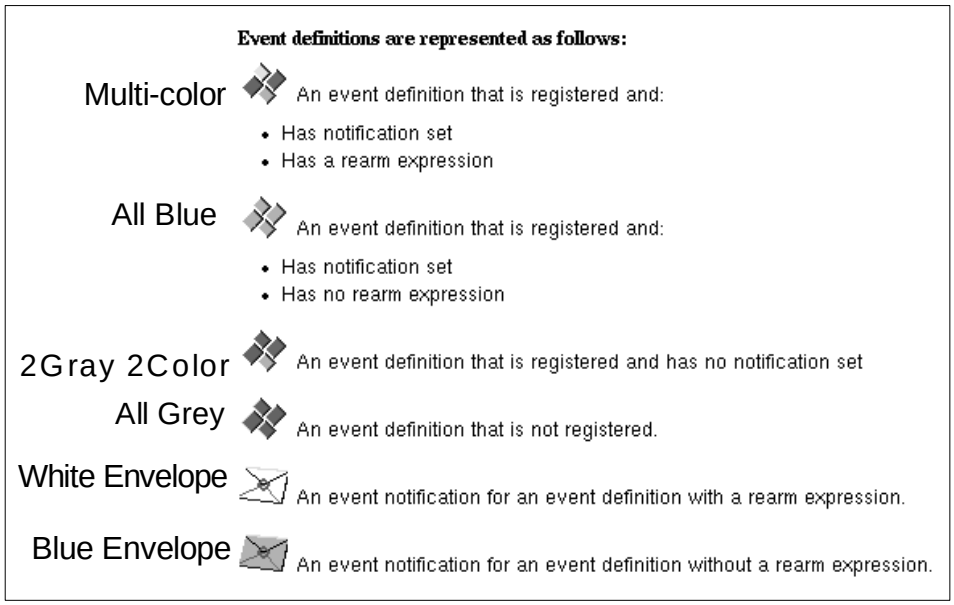


Figure 131. Icon Colors for Event Definitions

As an example, let us assume that the switchResponds event has been registered. Now if a node's switch adapter fails, or the Worm daemon dies, then the event icon will be replaced by the appropriate notification icon (in this case a white envelope, since a rearm expression has been set for this event). Once the node is back up on the switch, the icon will change back to its multi-colored status.

In addition to this visual alert, the system administrator will see an event notification log window pop up on his display. An example is shown in Figure 132 on page 301. There we can see both the original event notification when the switchResponds event was triggered, and the rearm notification once the problem was diagnosed and remedial action taken to fix it.

Ex.1010 / Page 324 of 572



Figure 132. Sample Event Notification Log

11.1.2.2 Create and Monitor Your Own Events

PSSP v3.1 introduces a simple interface for creating an Event Condition and defining an Event Definition. These procedures are extensively described in *PSSP 3.1 Announcement*, SG24-5332.

As an example, we will monitor a specific application file system for utilization greater than 80%. The steps required to do this are as follows:

1. Ensure that the Conditions Pane has been added to the Event Perspective window.
2. From the Event Perspective with the Conditions Pane highlighted, select **Actions->Create** from the menu bar.
3. In the Create Condition notebook window that is generated, you will see two Resource Variable selection panes, as shown in Figure 133.

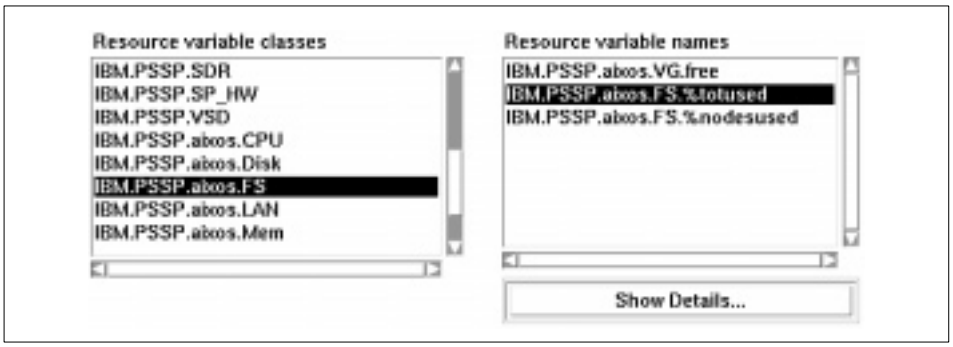


Figure 133. Event Definition - Resource Variable Selection

By scrolling through the classes pane, you find that the resource variable class associated with file systems is IBM.PSSP.aixos.FS and the variable name is IBM.PSSP.aixos.FS.%totused. Select these items, and complete the fields specifying the condition name and description, for example:

Name: apps_FS_used

Description: Monitors /apps file system on all nodes; triggers an event if utilization > 80%

4. Click the **Show Details** button to review a description of the resource variable and determine what to enter into the Event and (optional) Rearm Expression field. For this example, there are simple expressions:

   Event Expression: `X>80`

   Rearm Expression: `X<70`

5. Click **Ok**. The condition is created and appears in the Events Conditions pane.
6. Create an Event Definition based on this new condition. First, ensure that the Event Definitions pane is active. Then, from the menu bar, select **Actions->Create**. You should now see the Create Event Definition Window.
7. In the Condition Pane of this window, we can select the newly created Event Condition, apps_FS_used as shown in Figure 134.

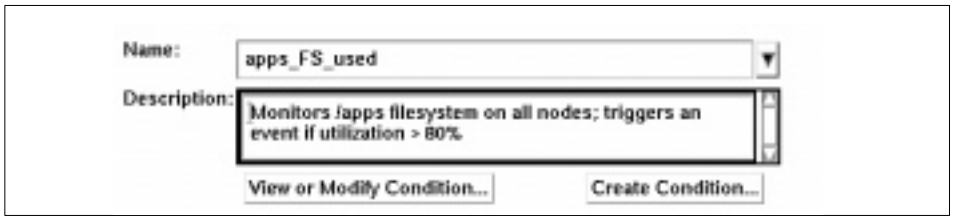


Figure 134. Event Definition - Condition Selection

8. Complete the Resource ID pane, which is shown in Figure 135 on page 303.

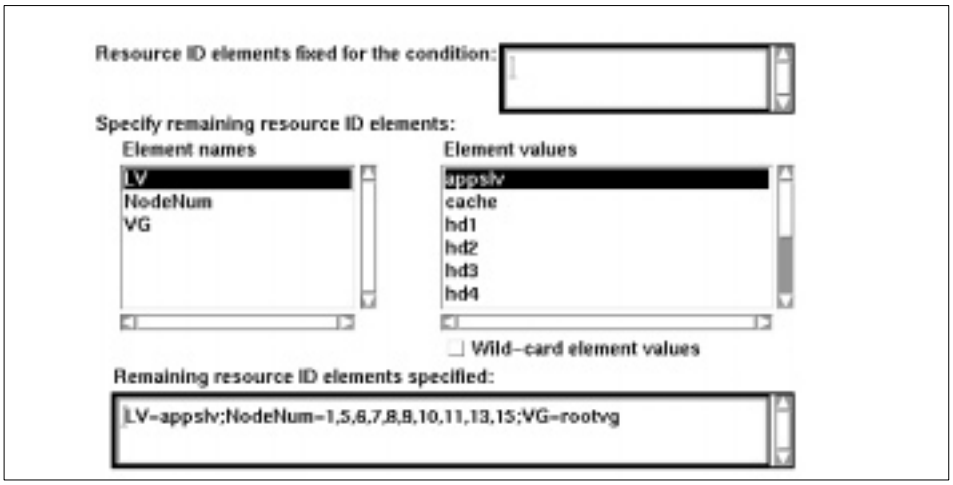


Figure 135. Event Definition - Resource Variable Identification

Here you can see that the resource elements IDs that have been selected are:

- The appslv logical volume, which resides on
- The rootvg volume group that is in
- Each node of the SP system (gaps in the node numbers indicate that there are no nodes in those slots)

9. Click on the **Create** button. The Event Definition will be saved and registered. When the file system utilization on any of these nodes becomes greater than 80% or falls below 70%, you will be notified by the Event Notification Log, as shown in Figure 136.

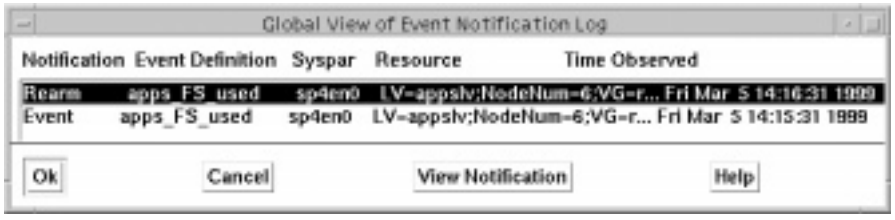


Figure 136. Event Notification Log

11.1.3 Virtual Shared Disk Perspective

IBM Virtual Shared Disk Perspective is the graphical user interface of PSSP that helps you perform shared disk management tasks. You can view, set, or change attributes and status, and you can control and monitor the operation of the shared disk management components. The IBM Virtual Shared Disk Perspective graphical user interface has the basic actions that let you perform most of your shared disk management work from within the graphical session.

Each action correlates to one or more virtual shared disk commands. You can run virtual shared disk, other PSSP, and AIX commands from within this interface as well. SMIT is also available to you for managing shared disks.

Explanation of the IBM Virtual Shared Disk Perspective graphical user interface is limited in this book to the brief introduction in this section. More information on VSDs can be found in 14.3, “Shared Disks” on page 394, and in *IBM Parallel System Support Programs for AIX Managing Shared Disks*, SA22-7349. A complete description of the capabilities of the VSD Perspective interface is given in the redbook *SP Perspectives: A New View of Your SP System*, SG24-5180.

Using this graphical interface requires your Kerberos principal to be defined in the virtual shared disk sysctl access control lists: /etc/sysctl.vsd.acl and /etc/sysctl.acl. The system administrator must run the command `sysctl svcrestart` for these changes to take effect. Once you have the correct authorization, you can start using the interface with either of two PSSP commands:

- Use the `perspectives` command to bring up the SP Perspectives Launch Pad window and then double-click on the **IBM VSD Perspective** icon to open the primary window.
- Use the `spvsd` command to open the IBM Virtual Shared Disk Perspective primary window directly.

Click **Help->Tasks** at the top right-hand corner of the primary window to see an online help system that explains how to use the IBM Virtual Shared Disk Perspective interface.

11.1.4 SP Command Line System Monitor

The SP System Monitor allows authorized operators and administrators to operate and monitor the system hardware. For information on using the command line interfaces to monitor and control the system hardware, refer to 5.3.1, “Command Line” on page 116.

Ex.1010 / Page 328 of 572

11.2 SP System Tuning & Performance Monitoring

SP system tuning is generally an exercise in optimizing networks. SP system management, interprocessor communications, shared disk access, interconnection to the enterprise, and programming depend heavily on thoughtful tuning and exploitation of networks. Once the SP networks are optimized, you can tune an individual SP node the way you would any stand-alone RS/6000 machine.

11.2.1 RS/6000 SP Performance Considerations

Tuning the SP system is a complex task. A number of sources of information are already available that provide information on specific aspects of tuning the SP for optimal performance and general information on performance and tuning in a AIX environment. See Appendix C, “Related Publications” on page 525.

SP tuning and performance documentation is also available on the World Wide Web at the following URL:

```
http://www.rs6000.ibm.com/support/sp/perf/
```

The information at this site is updated with the latest performance and tuning data.

11.2.1.1 Large Configurations

Since an RS/6000 SP may grow to 512 nodes, the rules and assumptions used in tuning small networks of RS/6000 machines change in an SP environment. SP tuning requires strong change management disciplines, which stems from the complexity of large system configurations. While the vast majority of SP installations have less than 50 nodes, valuable lessons can be learned from large configurations.

In large SP configurations, a bottlenecked node (such as a communications gateway) can affect all other nodes in the system. Bottlenecks in large systems have correspondingly amplified negative impacts. For example, if the SP administrative Ethernet is overloaded by heavy NFS traffic caused by multiple concurrent node installs, the normally light load of the RS/6000 Cluster Technology (RSCT) daemons can exacerbate the network problem. The daemons will step up their communications to cope with the apparent unavailability of resources, thereby causing performance degradation.

11.2.1.2 SP Subsystems and Daemons

PSSP introduces many subsystems and daemons to manage the SP. In the default installation, these facilities present a light load to the system. As you start to exploit their function, be careful of potential performance impacts. For example, you could configure monitoring for hundreds of resource variables with RSCT, or mistakenly monitor one resource variable that triggers a steady stream of events (such as CPU usage greater than 0 on a node, and log the message). Generally, SP subsystems are efficiently designed so they can scale, but they can also be misused.

11.2.1.3 SP Administrative Ethernet

Use care in the design and loading of the SP administrative Ethernet. Many system management functions and enablers use the SP administrative Ethernet, such as node installation and customizing, LoadLeveler, Parallel Operating Environment (POE), and RSCT. The SP administrative Ethernet may also be connected to your external networks, and open to traffic in your enterprise.

Generally, SP systems with more than 16 nodes have multiple boot/install servers on physically separate Ethernet LAN segments. This maintains adequate network performance, particularly for intensive system management tasks such as node installation.

More information on the design of the SP administrative Ethernet can be found in 3.5, “Administrative Ethernet” on page 47.

11.2.1.4 SP System Tunable Network Options

The SP installation procedures direct you to select an appropriate tuning file based on your specific system environment. Sample tuning files are provided in the directory /usr/lpp/ssp/install/config. You can copy one of the sample files (or create your own tuning file based on one of these) to /tftpboot/tuning.cust; this file will be installed on each of the nodes. It will be executed each time the node is booted to ensure that your required tuning variables are in effect. The network tuning variables that are in this file are summarized in Table 18.

Table 18. Network Tuning Variables

Network Option variable	Description
thewall	The upper bound on the amount of real memory that can be used by the communications subsystem. The units are in 1K increments.
sb_max	Upper limit on the size of the TCP and UDP buffers in allocated space per connection.

Network Option variable	Description
ipforwarding	Specifies whether the kernel should forward IP packets. A value of 1 forwards packets.
tcp_sendspace	The default size of the TCP send space value in bytes.
tcp_recvspace	The default size of the TCP receive space value in bytes.
udp_sendspace	The default size of the UDP send space value in bytes. The safe maximum is 65536 (64K).
udp_recvspace	The default size of the UDP receive space value in bytes.
rfc1323	Turns on several enhancements to the tcp_sendspace and tcp_recvspace tunables for high speed networks: •The maximum TCP window size is increased from 64 Kbytes to 4 Gigabytes. •ACKs are time-stamped for better round trip time estimates. Sequence numbers are protected against wrapping.
tcp_mssdflt	The default maximum segment size used in communicating with remote networks.

11.2.1.5 High-Speed Interconnect Network: SP Switch

The SP Switch is the interconnect fabric of the nodes. Many network tuning parameters affect all network interfaces on a node. Nodes of a switched SP system always have at least one other network - the administrative Ethernet - and usually other networks connecting into the enterprise. Optimizing these global parameters for such disparate networks as a slow Ethernet and a high-speed SP Switch is not trivial, and usually involves trade-offs.

The SP Switch introduces specific tuning parameters. You can tune the switch's device driver buffer pools, *rpool* and *spool*, by changing the rpoolsize and spoolsize parameters on the node's switch adapter. These pools are used to stage the data portions of IP packets. Their sizes interact with the node's network options settings. A detailed discussion of tuning the rpool and spool buffers is presented in an online document at:
`http://www.rs6000.ibm.com/support/sp/perf`

The send pool and receive pool are separate buffer pools, one for outgoing data (send pool) and one for incoming data (receive pool). When an IP packet is passed to the switch interface, if the size of the data is large enough, a buffer is allocated from the pool. If the amount of data fits in the IP header

Ex.1010 / Page 331 of 572

mbuf used in the mbuf pool, no send pool space is allocated for the packet. The amount of data that will fit in the header mbuf is a little less than 200 bytes, depending on what type of IP packet is being sent. The header size varies between using UDP or TCP, or having rfc1323 turned on.

To see the current send pool and receive pool buffer sizes, run the following command on your nodes:

```
[sp4n01:/]# lsattr -El css0
bus_mem_addr   0x04000000 Bus memory address        False
int_level      0xb        Bus interrupt level       False
int_priority   3          Interrupt priority        False
dma_lvl        9          DMA arbitration level     False
spoolsize      524288     Size of IP send buffer    True
rpoolsize      524288     Size of IP receive buffer True
adapter_status css_ready  Configuration status      False
[sp4n01:/]#
```

The default values for these buffer pools is set at 512 Kbytes or 524288 bytes. After reviewing your current usage of the buffer pools (by running the `vdidl3` command on the node or nodes), and following the suggestions and recommendations in the online document, you can modify the spoolsize and/or rpoolsize parameters using the `chgcss` command. For example, to change both the send and receive buffer pool size on a node to 1 Mbyte, enter:

```
chgcss -l css -a rpoolsize=1048576 -a spoolsize=1048576
```

New values for the pool sizes must be expressed in bytes, and will not take effect until the node(s) are re-booted.

The small extra latency incurred by traversing intermediate switch connections in SP systems greater than 80 nodes may also be a consideration where consistent, precision results are required. (Remember that with configurations greater than 80 nodes, you need to add an intermediate switch frame to maintain switch performance.) For example, if a parallel job is dispatched to a set of nodes that all belong to one side of the intermediate switch frame, the execution time will be slightly faster than if the job is dispatched to the same number of identically configured nodes residing on either side of the intermediate switch frame.

11.2.1.6 Large Volumes of Data

SP systems typically have large volumes of data distributed across many nodes. Access to and movement of data within the system is critical to overall system performance. NFS, VSDs, and GPFS are all mechanisms for sharing

Ex.1010 / Page 332 of 572

data in the SP. Each has particular tuning considerations and techniques, but fundamentally, an I/O request eventually comes down to a disk subsystem attached to a node. You must ensure that the I/O subsystem itself is not a bottleneck; otherwise, higher-level tuning will be fruitless.

Detailed tuning information for VSD and RVSD (the basis of GPFS) can be found in *IBM Parallel System Support Programs for AIX Managing Shared Disks,* SA22-7349.

11.2.1.7 External Networks

When you connect external networks to the SP, care must be taken to optimize the exchange of small data packets (typical of Ethernet, Token Ring, and similar slower-speed networks) and the large data packets of the SP Switch.

Adapter Transmit and Receive Queue Tuning

Most communication drivers provide a set of tunable parameters to control transmit and receive resources. These typically control the transmit queue and receive queue limits. These parameters limit the number of buffers or packets that may be queued for transmit, or limit the number of receive buffers that are available for receiving packets. These parameters can be tuned to ensure enough queueing at the adapter level to handle the peak loads generated by the system or the network.

Transmit queues

For transmit, the device drivers may provide a transmit queue limit. There may be both hardware queue and software queue limits, depending on the driver and adapter. Some drivers have only a hardware queue, some have both hardware and software queues. Some drivers internally control the hardware queue and only allow the software queue limits to be modified. Generally, the device driver will queue a transmit packet directly to the adapter hardware queue. On an SMP system, or if the system CPU is fast relative to the speed of the network, the system may produce transmit packets faster than they can be transmitted on the network. This will cause the hardware queue to fill. Once the hardware queue is full, some drivers provide a software queue and will then queue to the software queue. If the software transmit queue limit is reached, then the transmit packets are discarded. This can affect performance because the upper level protocols must then timeout and retransmit the packet.

Prior to AIX 4.2.1, the upper limits on the transmit queues were in the range of 150 to 250, depending on the specific adapter. The system default values were quite low, typically 30. With AIX 4.2.1 and later, the transmit queue limits were increased on most of the device drivers to 2048 buffers. The default

Ex.1010 / Page 333 of 572

values were also increased to 512 for most of these drivers. The default values were increased because faster CPUs and SMP systems can overrun the smaller queue limits.

For adapters that provide hardware queue limits, changing these values will cause more real memory to be consumed because of the associated control blocks and buffers associated with them. Therefore, these limits should only be raised if needed, or for larger systems where the increase in memory use is negligible. For the software transmit queue limits, increasing these does not increase memory usage. It only allows packets to be queued that were already allocated by the higher layer protocols.

Receive Queues

Some adapters allow you to configure the number of resources used for receiving packets from the network. This might include the number of receive buffers (and even their size) or may simply be a receive queue parameter (which indirectly controls the number of receive buffers). The receive resources may need to be increased to handle peak bursts on the network.

When to Increase the Receive/Transmit Queue Parameters

Normally you would consider changing the queue size if one of the following situations arises in your network:

- The CPU is much faster than the network and multiple applications may be using the same network. This would be common on a larger multi-processor system (SMP).
- Running with large values for tcp_sendspace or tcp_recvspace as set in the network options, or running applications that might use system calls to increase the TCP send and receive socket buffer space. These large values can cause the CPU to send down large numbers of packets to the adapter, which will need to be queued. A similar situation exists for udp_sendspace and udp_recvspace for UDP applications.
- Network traffic is characterized by large data bursts.
- A high traffic load of small packets may consume more resources than a high traffic load of large buffers. This is because the large buffers take more time to send on the network, so the packet rate is slower for larger packets.

SP Considerations

One of the final RS/6000 SP installation steps is setting all network adapters in SP nodes to their maximum transmit queue size. With AIX release 4.2.1 and later, the default transmit queue limit has been increased to 512.

Ex.1010 / Page 334 of 572

Attention

Be aware that the transmit queue size values recommended in *IBM PSSP for AIX Installation and Migration Guide*, GA23-7347 at **Step 59: Tune the Network Adapters** are incorrect.

Check the value of this adapter attribute on all newly installed nodes using an appropriate `dsh` command, such as:

```
# dsh -a lsattr -El ent0 | grep xmt_que_size
```

If the value for the queue size is not 512, then change it using:

```
# dsh -a chdev -P -l ent0 -a  xmt_que_size=512
```

Reboot the nodes to make the change effective. A value of 512 should be sufficient for initial operation, and can be increased if you change the tcp_sendspace or tcp_recvspace as set in the network options.

An Ethernet MTU is usually 1500 bytes. 512 MTUs represent 768,000 bytes, more than ten times the size of a single packet (MTU) on the SP Switch, which is 65,520 bytes.

11.2.2 Performance Management Tools

As part of the RS/6000 family of AIX machines, the RS/6000 SP inherits a set of mature performance management tools. In addition, IBM has enhanced the tools to scale with the SP.

11.2.2.1 AIX Facilities

AIX provides all the traditional UNIX resource monitoring tools. Each node's resources can be interrogated at the operating system level. Examples of these tools include the commands `sar`, `vmstat`, and `iostat`. If you wish to use these commands, the bos.acct fileset has to be installed on the nodes.

AIX Version 4 includes Performance Diagnostic Tool (PDT). PDT assesses the current state of a system and tracks changes in workload and performance. It attempts to identify incipient problems and suggest solutions before the problems become critical.

For the most part, PDT functions with no required user input. PDT data collection and reporting are easily enabled, and then no further administrator activity is required. Periodically, data is collected and recorded for historical analysis, and a report is produced and mailed to the adm userid. Normally, only the most significant apparent problems are recorded on the report. If

there are no significant problems, that fact is reported. PDT can be customized to direct its report to a different user or to report apparent problems of a lower severity level.

PDT optionally runs on individual nodes, assesses the system state, and tracks changes in performance and workload. It tries to identify impending problems and suggest solutions before they become critical.

The PerfPMR package was developed to ensure that reports of suspected performance problems in AIX were accompanied by enough data to permit problem diagnosis by IBM. This makes the shell scripts in PerfPMR useful to other performance analysts as well. PerfPMR is an optionally installable part of the AIX Version 4 Base Operating System.

The SP currently offers no specific reporting consolidation or meta-level commands in this area. AIX resource monitoring capabilities are well explained in *AIX Performance Monitoring and Tuning Guide*, SC23-2365.

11.2.2.2 Reporting Resource Data Information

Performance Agent, a component of Performance Toolbox/6000, is an element of the resource monitoring facility of RSCT. By itself, Performance Agent only supplies data from performance related resource variables to the System Performance Measurement Interface (SPMI); it offers little in the way of viewing or analyzing the data. IBM supplies program products specifically to exploit the performance data from Performance Agent.

Performance Toolbox (PTX/6000)

PTX/6000 comprises the Performance Manager (PM) and Performance Agent (PA), or perfagent.server. The latter is the exact same code that ships automatically with the SP. The PM component includes X-Windows and Motif-based applications that provide:

- Real-time, color graphic performance monitors for local and remote systems, such as 3dmon
- Performance analysis tools
- Performance tuning controls

PTX/6000 has a long, successful heritage of monitoring the performance of RS/6000 stand-alone machines. PTX/6000 is a separately orderable and priced IBM Licensed Program Product (LPP).

With reference to Figure 137 on page 313, the xmservd daemon of PA feeds the performance statistics into shared memory. While the aixos resource monitor within the Event Management (EM) daemon can supply AIX-level resource variable data selectively, as per EM client event registrations,

Ex.1010 / Page 336 of 572

xmservd can be further configured for specific resource monitoring needs. Performance Manager (PM) can simply act as another client of the SPMI interface (or Local Data Consumer in PTX/6000 vernacular), just as does the Resource Monitors of Event Manager. The diagram introduces another interface into shared memory: the Remote Statistics Interface (RSI). This interface allows a PM instance on another node (shown as a Remote Data Consumer, in this case the 3dmon monitoring program of PM) to access the local node's shared memory. This way, one PM installation can monitor a cluster of machines at the same time.

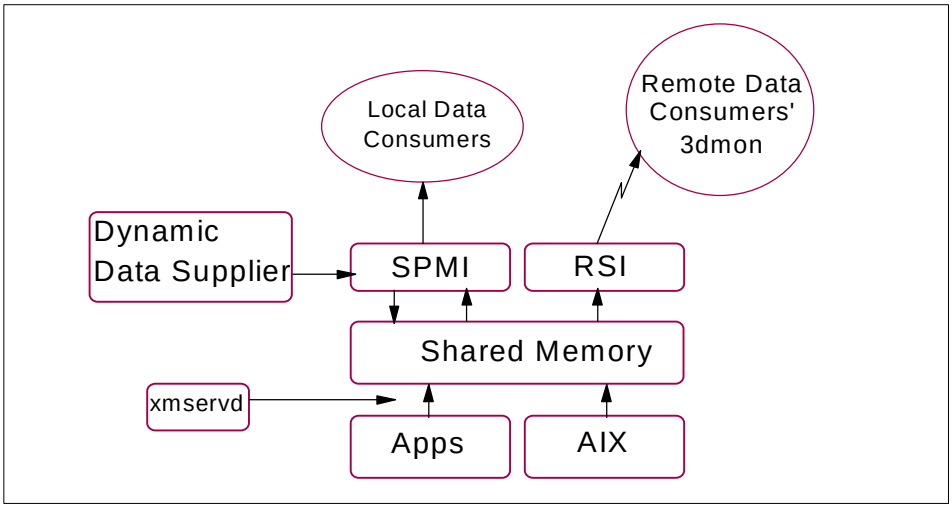


Figure 137. PTX Functional Overview

Although Performance Agent (PA) is an integral part of the SP's software implementation, PM offers no SP-specific function. Its usability does not scale: on larger SP installations (greater than 24 nodes), its graphical monitoring tools cannot illustrate on a screen all the performance statistics of all nodes. PM is not aware of specific SP hardware and software enablers.

Performance Toolbox Parallel Extensions (PTPE)

PTPE extends the function of PTX/6000 for use in an RS/6000 SP complex. PTPE is a scalable performance monitor, and when it is installed, it provides easy access to performance information. Any node or group of nodes can be monitored with a single point of control. The performance metrics and the information display criteria can be specified. Monitoring can be carried out in real time, or the performance data archived for later analysis. For detailed information on configuring and using PTPE, refer to *IBM PSSP for AIX*

Performance Monitoring Guide and Reference, SA22-7353 and the redbook *RS/6000 SP Performance Tuning*, SG24-5340.

PTPE performance statistics can be viewed:

- Averaged across the RS/6000 SP complex
- Averaged across a subset of nodes
- For individual nodes

PTPE setup, configuration and management have been integrated into Perspectives, or alternatively, can be carried out using AIX commands.

The implementation of PTPE is shown in Figure 138.

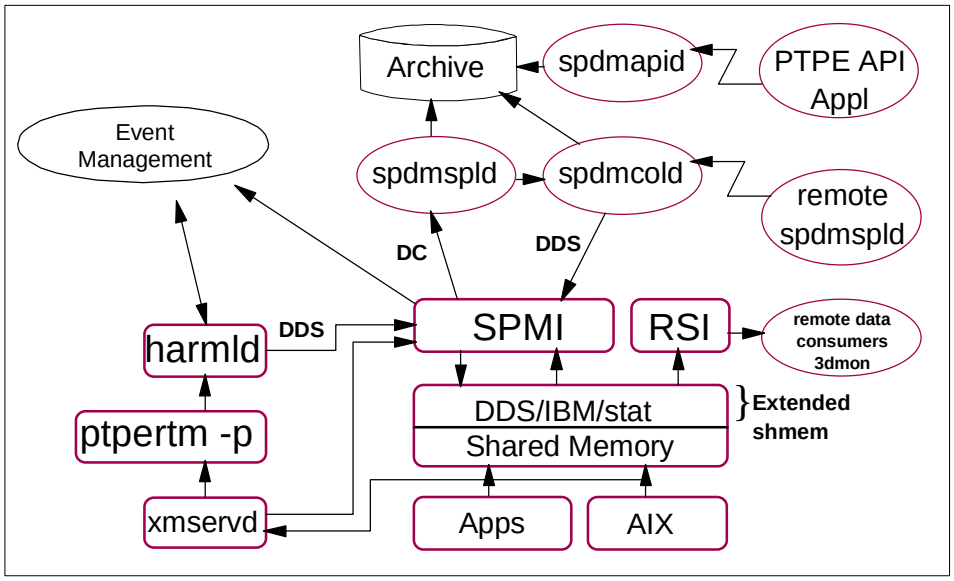


Figure 138. PTPE Architecture

In addition to the capabilities of PTX/6000, PTPE provides:

- **Collection of SP-specific data** - PTPE provides ptpertm, an additional data supplier that complements the data that xmservd collects. The SP-specific performance data is currently implemented for:
  - The SP Switch
  - LoadLeveler
  - VSD
- **SP runtime monitoring** - The system administrator should ideally have a global view of SP performance behavior. This is accomplished by grouping

nodes into specific groups. With reference to Figure 139 on page 315, similar nodes of the first tier, or Collectors, can be grouped and their performance data summarized by their respective Data Manager node in the second tier. In this way, large SP systems can be easily monitored from a single presentation application by viewing node groups instead of individual nodes. The Data Managers are administered by the Central Coordinator in the third tier. The Central Coordinator aggregates the Data Managers' summary data to provide a total performance overview of the SP. Of course, the base PTX/6000 monitoring functions can be used to focus on any particular performance aspect of an individual node.

The PTPE display monitors on the Data Manager receive data from spdmspld (SP Data Manager Sampler daemon) on the local node. To acquire data from remote Collectors, spdmspld communicates with spdmcold (SP Data Manager Collector daemon) on the remote nodes, which in turn get the data from their associated spdmspld daemon. The communications between these daemons is via a private protocol. Note that PTPE does not use the same mechanism as PTX/6000 to collect data from remote nodes.

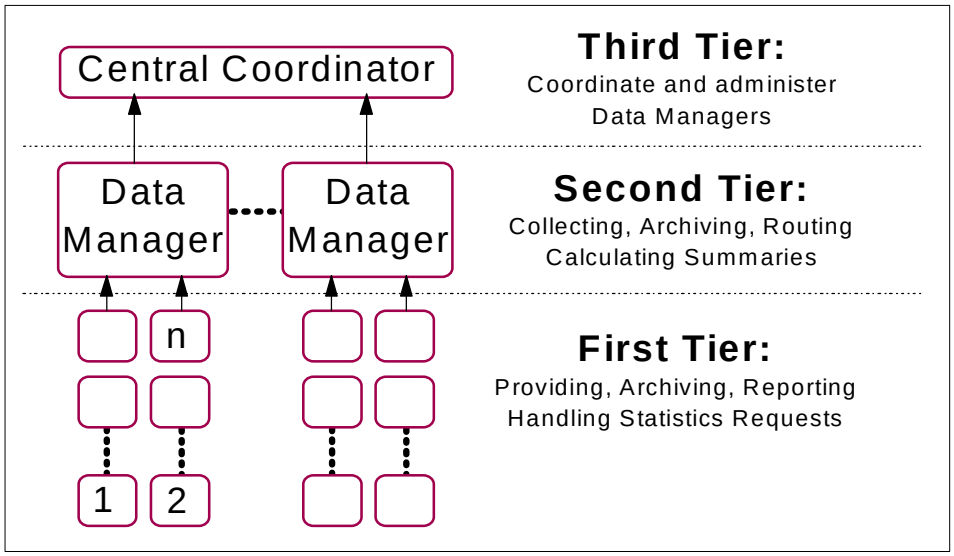


Figure 139. PTPE Monitoring Hierarchy

- **Data Analysis and Data Relationship Analysis** - PTPE provides an API to allow analysis applications to sift through all requested data. The data archive created by PTPE exists on every node, and is completely accessible through the API using the spdmapid daemon (see Figure 138

on page 314). In base PTX/6000, performance data is analyzed with the azizo utility, which is restricted to simple derivatives like maximum, minimum, and average. With the PTPE API, programs of any statistical complexity can be written to find important trends or relationships. Also, data captured for azizo use is far more limited with base PTX/6000.

PTPE Installation and Customization

To install and run the Performance Toolbox Parallel Extensions for AIX, the following IBM licensed program products must also be installed:

- AIX Version 4 Release 3.2
- RS/6000 Cluster Technology
- Performance Aide for AIX Version 4.1 (5696-899)
  This must be installed on all nodes where PTPE is to run (the nodes you intend to monitor), as well as on the control workstation.
- Performance Toolbox-Network for AIX Version 4.1 (5696-900)
  This must be installed on any nodes where you will display performance data (the nodes from which you intend to monitor).

There are three filesets in the PSSP installation media that make up PTPE:

ptpe.program The PTPE programs. This software will not run unless RSCT has been installed.

ptpe.docs Documentation material, for example man pages.

ssp.ptpegui This image should be installed on any node on which you plan to run SP Perspectives.

The software installation is straightforward, and details can be found in Chapter 3, “Installing PTPE” of *IBM PSSP for AIX Performance Monitoring Guide and Reference*, SA22-7353.

Before you can use Performance Toolbox Parallel Extensions for AIX, you must create a monitoring hierarchy as shown in Figure 139 on page 315. PTPE distributes the management of performance information among a number of data manager nodes rather than giving total responsibility to a single node. Although one central coordinator node is designated when the monitoring hierarchy is created, the data manager nodes act as intermediaries, absorbing most of the administrative overhead and greatly reducing data transfer operations. The resulting central coordinator node workload is far less than that required by a single point of control for all nodes and all data management functions.

You can create a monitoring hierarchy using either the SP Perspectives Performance Monitor or the `ptpehier` command. In our example, we will

Ex.1010 / Page 340 of 572

consider the single frame SP system shown in Figure 140, and use the `ptpehier` command.

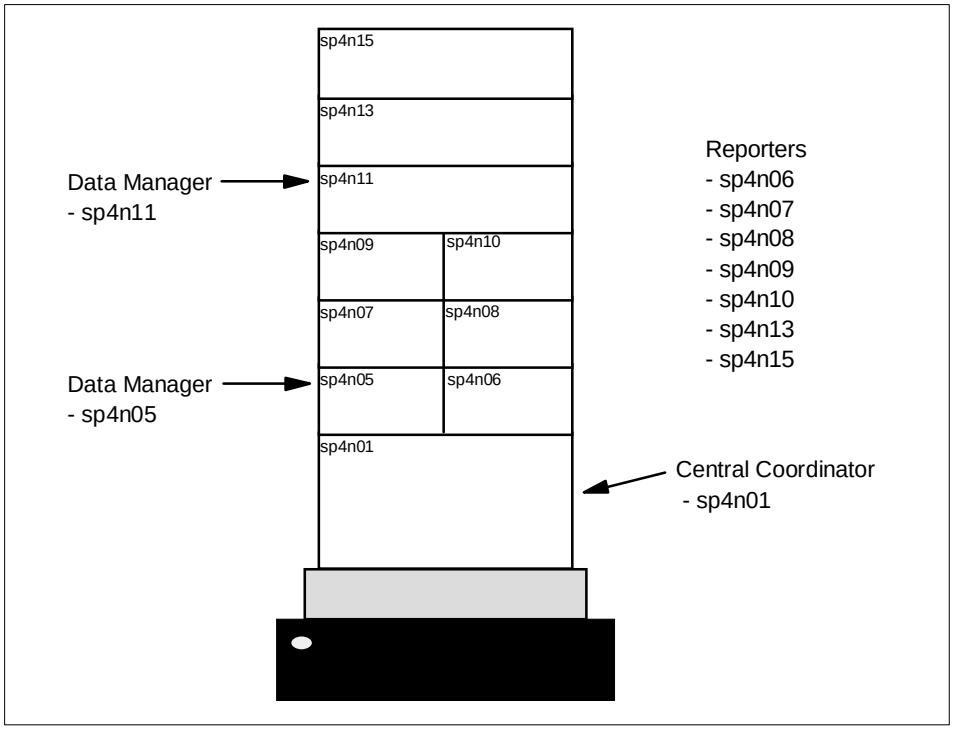


Figure 140. RS/6000 SP System with Defined Roles

Here we have decided that nodes sp4n05 and sp4n11 will be Data Managers, and sp4n01 will be the Central Coordinator; the remaining nodes will be simply Reporters. To formalize this hierarchy, you create a text file (we call it samp_hier) in the /tmp directory, with the following format:

Ex.1010 / Page 341 of 572

```
{
sp4n05.msc.itso.ibm.com
sp4n06.msc.itso.ibm.com
sp4n07.msc.itso.ibm.com
sp4n08.msc.itso.ibm.com
sp4n09.msc.itso.ibm.com
sp4n10.msc.itso.ibm.com
}
{
sp4n11.msc.itso.ibm.com
sp4n13.msc.itso.ibm.com
sp4n15.msc.itso.ibm.com
sp4n01.msc.itso.ibm.com
}
```

To initialize this hierarchy, you use the `ptpehier` command with a -c flag to specify the Central Coordinator node and -i flag to specify that the hierarchy will be read from standard input — in this case, the /tmp/samp_hier file. You can also let PTPE automatically create a hierarchy, based on your Ethernet local area subnetwork, or if you have a multi-frame SP system, then it can develop a hierarchy based on the node/frame configuration. In our example, we use the following commands to set up and confirm the hierarchy.

```
[sp4en0:/]# /usr/lpp/ptpe/bin/ptpehier -i -c sp4n01 < /tmp/samp_hier
ptpehier: Monitoring hierarchy successfully created.

[sp4en0:/]# /usr/lpp/ptpe/bin/ptpehier -p
ptpehier: The current monitoring hierarchy structure is:
        sp4n01.msc.itso.ibm.com
                sp4n11.msc.itso.ibm.com
                        sp4n11.msc.itso.ibm.com
                        sp4n13.msc.itso.ibm.com
                        sp4n15.msc.itso.ibm.com
                        sp4n01.msc.itso.ibm.com
                sp4n05.msc.itso.ibm.com
                        sp4n05.msc.itso.ibm.com
                        sp4n06.msc.itso.ibm.com
                        sp4n07.msc.itso.ibm.com
                        sp4n08.msc.itso.ibm.com
                        sp4n09.msc.itso.ibm.com
                        sp4n10.msc.itso.ibm.com
[sp4en0:/]#
```

All that remains is to initialize and start the data collection process using the `ptpectrl` command with the -i and -c flags:

Ex.1010 / Page 342 of 572

```
[sp4en0:/]# ptpectrl -i
------------------------------------------------------------
ptpectrl: Beginning setup for performance information collection.
ptpectrl: Reply from the Central Coordinator expected within 255 seconds.
ptpectrl: Setup for collecting performance information succeeded.
------------------------------------------------------------
ptpectrl: Command completed.
------------------------------------------------------------
[sp4en0:/]#

[sp4en0:/]# ptpectrl -c
------------------------------------------------------------
ptpectrl: Starting collection of performance information.
ptpectrl: Reply from the Central Coordinator expected within 250 seconds. OK.
ptpectrl: Performance information collection successfully started.
------------------------------------------------------------
ptpectrl: Command completed.
------------------------------------------------------------
[sp4en0:/]#
```

Now you can use the familiar xmperf and 3dmon commands to review the performance of the nodes in your SP complex. Since performance monitoring creates additional load on the nodes being monitored, you should shut down the monitoring process once you have collected sufficient performance data, using the `ptpectrl -s` command.

11.2.2.3 Performance Event Monitoring with RSCT Clients

Performance-related resource variables from Performance Agent are part of Event Manager's standard suite of monitored resources variables. You can always use your own programs or PMAN to interface to RSCT to monitor and take action on performance-related events. See 8.6, "Event Management (EM)" on page 200 and 11.4.2, "Problem Management (PMAN) Subsystem" on page 329 for more on this topic.

11.3 SP Accounting

The SP accounting facilities are based on the standard System V accounting system utilities of AIX. These accounting facilities are described in *AIX Version 4.3 System Management Guide: Operating System and Devices*, SC23-4126. Enabling SP system accounting support is optional. It is a separately-installed portion of the PSSP ssp.sysman fileset. SP system accounting extends the function of base AIX accounting in three ways:

- **Accounting-record consolidation** - Partial reduction of accounting data is done on each node before the data is consolidated on an accounting

master. (The accounting master will generally be the control workstation, but it can be any node.) Nodes are configured into groups called accounting classes. All the data from a given accounting class is consolidated. These accounting classes can be used to impose different charges for nodes with differing capabilities.

The administrator can use the SMIT menus and dialogs or the command line interface to specify the accounting configuration. The SP system management code automatically configures accounting on the selected nodes. The command `runacct` (a modified version of the AIX `runacct` command) is scheduled to run nightly on each node to consolidate the accounting data for that node. Data from all of the nodes is then consolidated on the accounting master node, which performs additional processing. The output of this processing is standard format data that can be used for charge-back purposes, usage monitoring or capacity planning. It also can be fed into existing accounting applications that you may have already implemented.

- **Node-exclusive-use accounting** - Standard accounting generates charges for resources used (processor, disk, and so on). If a user is given exclusive use of a set of nodes for the duration of a parallel job, standard accounting may not be an appropriate way to charge for processor usage. Instead, administrators may want to bill based on the wall-clock time that a processor is in use since it is unavailable to other users during that period (regardless of what processor cycles are actually consumed by the running job).

  The PSSP software provides an optional mechanism with which to charge for nodes that have been assigned for exclusive use. If this support is enabled, an accounting record for a marker process is created before and after the job is run. All processor accounting records associated with that user and falling within the window of the marker records are discarded. Instead, a special charge is applied based on the actual number of seconds the job runs. The fee is specified by the administrator through the standard accounting charge-fee mechanism.

- **Parallel job accounting** - The LoadLeveler job-management program allows job-based accounting by accumulating data on resource usage from all processes triggered by a specific job.

11.3.1 Configuring SP Accounting

AIX accounting requires the creation of directories, files, and crontab entries, as well as accounting configuration activity. This is appropriate for a single system, but a daunting task for an SP with tens or hundreds of nodes. The PSSP software simplifies this task greatly. After installation of the accounting

portion of the SP, default values in the SDR configure accounting across the entire system. Accounting can be enabled or disabled globally by changing the site environment data. Node-by-node accounting activation, accounting classes creation, and exclusive-use support on a node are efficiently configured using SMIT menus or SDR commands to update node attributes collectively. The accounting installation process creates all necessary directories and files on the nodes for which accounting is enabled.

The following procedure helps you to tailor your accounting system by defining accounting class identifiers and accounting-enabled attributes on a SP 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. This threshold specifies the minimum percentage of nodes for which accounting data must be present in order for processing to continue.

You can define or change these values after installation by using SMIT (the fastpath is `site_env_dialog`) or the `spsitenv` commands. An extract of the relevant SMIT screen is shown in Figure 141.

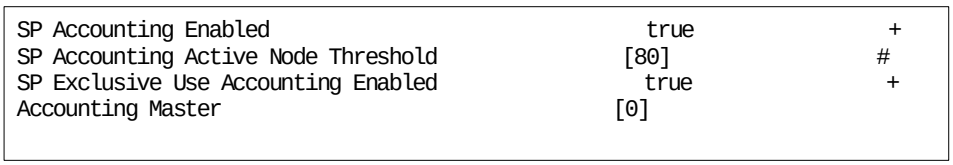


Figure 141. Site Environment- Accounting Setup

The `spsitenv` command could also be entered at a prompt on the control workstation:

```
# spsitenv acct_master=0 spacct_enable=true spacct_excluse_enable=true
```

You also need to further define SP accounting at the node level. This can be done using the SMIT fastpath `acctnode_dialog` or the `spacctnd` command. The following sequence of `spacctnd` commands will set the job charge value for nodes 1, 9 and 10 to 30.0, and these nodes will have the *default* accounting

class identifier. Nodes 5, 6, 7, 8, 11, 13 and 14 will have the *pjobs* accounting class identifier, exclusive use accounting will be enabled, and the job charge value will be set to 60

```
# spacctnd -j 30.0 1 1 1
# spacctnd -j 30.0 1 9 2
# spacctnd -c pjobs -x true -j 15.0 1 5 4
# spacctnd -c pjobs -x true -j 15.0 1 11 3
```

We can review the relevant attributes of the Node class in Figure 142 on page 322.

```
[sp4en0:/]# SDRGetObjects Node node_number acct_class_id acct_enable acct_job_charge
acct_excluse_enable

node_number  acct_class_id acct_enable  acct_job_charge acct_excluse_enable
          1 default      default      30.0         false
          5 pjobs        default      15.0         true
          6 pjobs        default      15.0         true
          7 pjobs        default      15.0         true
          8 pjobs        default      15.0         true
          9 default      default      30.0         false
         10 default      default      30.0         false
         11 pjobs        default      15.0         true
         13 pjobs        default      15.0         true
         15 pjobs        default      15.0         true
[sp4en0:/]#
```

Figure 142. SP Nodes - Accounting Setup

These accounting installation procedures result in the following activities being carried out on the accounting master node (usually the control workstation) and nodes.

1. All necessary directories and files on the nodes for which accounting is enabled and on the acct_master node are created.
2. The accounting startup command is added to the /etc/rc file on the nodes for which accounting is enabled.
3. The /var/adm/acct/nite/jobcharge file is created 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 to all nodes in the SP system, using whatever method is available in your environment.

Note: You may have to update the holidays file to reflect the correct information for the current year.

5. The crontabs file is updated to schedule the accounting processes.
6. The login and process account data files are initialized by the `nulladm` command.

```
/usr/sbin/acct/nulladm /var/adm/wtmp /var/adm/pacct
```

Tools and Considerations

Like AIX accounting, the SP merges accounting data by user ID and login name. This forces all SP nodes for which accounting is enabled to have identical user ID and login names; in other words, belong to the same user name space. Name space can be managed by NIS, SP User Management, or other mechanisms.

See *IBM Parallel System Support Programs for AIX Administration Guide*, SA22-7348, for more information on SP accounting implementation.

11.3.2 LoadLeveler Accounting

LoadLeveler is an application designed to automate workload management. In essence, it is a scheduler which also has facilities to build, submit and manage both serial and parallel jobs. The jobs can be processed by any one of a number of machines, which together are referred to as the LoadLeveler cluster. Any standalone RS/6000 can be part of a cluster, although LoadLeveler is most often run in the RS/6000 SP environment.

For more information about configuring and administering LoadLeveler on an SP system, see Chapter 17, “LoadLeveler” on page 497, and *IBM LoadLeveler for AIX: Using and Administering Version 2 Release 1*, SA22-7311.

LoadLeveler can collect a variety of accounting information, based on your local requirements. Refer to the relevant documentation for more information.

Job Resource Data - Machines

LoadLeveler can collect job resource usage information for every machine on which a job may run. A job may run on more than one machine because it is a parallel job or because the job is vacated from one machine and rescheduled to another machine.

To turn on accounting in this environment, specify

```
ACCT = A_ON A_DETAIL
```

in the LoadLeveler configuration file LoadL_config.

Job Resource Data - Serial and Parallel Jobs

Information on completed serial and parallel jobs is gathered using the UNIX *wait3* system call. Information on non-completed serial and parallel jobs is gathered in a platform-dependent manner by examining data from the UNIX process.

Accounting information on a completed serial job is determined by accumulating resources consumed by that job on the machine(s) that ran the job. Similarly, accounting information on completed parallel jobs is gathered by accumulating resources used on all of the nodes that ran the job.

To turn on accounting in this environment, specify the following keywords in the LoadLeveler configuration file LoadL_config.

```
ACCT = A_ON A_DETAIL
JOB_ACCT_Q_POLICY = num1
JOB_LIMIT_POLICY = num2
```

The value num1 is an accounting update frequency (in seconds) with a default value of 300, and num2 is also a number in seconds; the smaller of these two numbers controls how often resource consumption data on running jobs is collected.

Job Resource Data - Events

LoadLeveler can also collect job resource information based on events or times that you specify. For example, you may want to collect accounting information at the end of every work shift or at the end of every week or month.

To collect accounting information from all machines in this way, the *llctl* command is used with the *capture* keyword

```
llctl -g capture eventname
```

where eventname is any string of continuous characters that defines the event about which you are collecting accounting information. For example, to collect this type of information regularly, you can add crontab entries, such as:

```
00 08 * * * /u/loadl/bin/llctl -g capture third
00 16 * * * /u/loadl/bin/llctl -g capture first
00 00 * * * /u/loadl/bin/llctl -g capture second
```


Ex.1010 / Page 348 of 572

Job Resource Information — User Accounts

You can also keep track of resources used on an account basis by requiring all users to specify an account number in their job command files. Users can specify this account number with the *account_no* keyword. This account number must be a member of a set of account numbers specified in the LoadLeveler administration file LoadL_admin with the *account* keyword.

Examples

Once LoadLeveler accounting has been configured, you can extract job resource information on completed jobs by using the `llsummary` command. For detailed information on the syntax of this command and the various output reports that it can generate, see *LoadLeveler for AIX: Using and Administering Version 2 Release 1*, SA22-7311.

LoadLeveler stores the accounting information that it collects in a file called *history* in the /home/loadl/spool directory of the machine that initially scheduled this job. Data on parallel jobs is also stored in the *history* files of the nodes that are part of the parallel job pool.

You can produce three types of reports using the `llsummary` command. These reports are called the *short*, *long*, and *extended* versions. As their names imply, the short version of the report is a brief listing of the resources used by LoadLeveler jobs, the long version provides more comprehensive detail with summarized resource usage and the extended version of the report provides the comprehensive detail with detailed resource usage. If you do not specify a report type, you will receive the default short version.

The short report displays the number of jobs, along with the total CPU usage according to user, class, group, and account number. The extended version of the report displays all of the data collected for every job.

An extract of a short report follows.

```
$ llsummary
        Name    Jobs    Steps        Job Cpu    Starter Cpu     Leverage
       user8       8        8     0+00:00:56     0+00:00:02        28.0
     usera91       5        5     0+00:00:01     0+00:01:10         0.0
       TOTAL      13       13     0+00:00:57     0+00:01:13         0.8

       Class    Jobs    Steps        Job Cpu    Starter Cpu     Leverage
    No_Class      13       13     0+00:00:57     0+00:01:13         0.8
       TOTAL      13       13     0+00:00:57     0+00:01:13         0.8

       Group    Jobs    Steps        Job Cpu    Starter Cpu     Leverage
    No_Group      13       13     0+00:00:57     0+00:01:13         0.8
       TOTAL      13       13     0+00:00:57     0+00:01:13         0.8

     Account    Jobs    Steps        Job Cpu    Starter Cpu     Leverage
        NONE      10       10     0+00:00:28     0+00:01:12         0.4
        test       3        3     0+00:00:29     0+00:00:01        29.0
       TOTAL      13       13     0+00:00:57     0+00:01:13         0.8

$
```

The short listing includes the following fields:

Name User ID submitting jobs

Class Class specified or defaulted for the jobs

Group User's LL group

Account Account number specified for the jobs

Jobs Count of the total number of jobs submitted by this user, class, group, or account

Steps Count of the total number of job steps submitted by this user, class, group, or account

Job CPU Total CPU consumed by user's jobs

Starter CPU Total CPU time consumed by LoadLeveler starter processes on behalf of the user jobs

Leverage Ratio of job CPU to starter CPU

11.4 Problem Management

The RS/6000 SP uses many techniques to report and act on errors and problems that are encountered during system operation. In general, problem management on the RS/6000 SP exploits the existing AIX facilities and the rich services provided by the RS/6000 Cluster Technology (RSCT) (known in pre-PSSP v3.1 releases as High Availability Infrastructure or RSCT). This topic is discussed in Chapter 8, "RS/6000 Cluster Technology" on page 185.

11.4.1 AIX, BSD and PSSP-Specific Error Logging

The RS/6000 SP uses both the AIX Error Logging facilities and the BSD syslog, as well as a number of function-specific log files to record error events on each node. Commands and SMIT panels are available to perform general log management. In order to manage Syslog and the AIX Error Log, the ssp.sysman fileset must be installed.

Error logging reports debugging information into log files for subsystems that perform a service or function on behalf of an end user. The subsystem does not communicate directly with the end user and therefore needs to log events to a file. The events that are logged are primarily error events.

Error logging for the SP uses BSD syslog and AIX Error Log facilities to report events on a node basis. The System Monitor and the SP Switch use this form of error logging.

Error log entries include a DETECTING MODULE string that identifies the software component, module name, module level, and the line of code or function that detected the event that was logged. The information is formatted based on the logging facility the user is viewing. For example, the AIX Error Log facility information appears as follows:

```
DETECTING MODULE
LPP=<LPP name>,Fn=<filename>, SID=<ID_level_of_the_file>,L#=<line number>
```

An extract from an actual error log entry is shown here:

```
DETECTING MODULE
LPP=PSSP,Fn=harmld.c,SID=1.36,L#=2595,
```

The BSD syslog facility information appears as follows:

```
<timestamp, hostname, ID, PID>
LPP=<LPP name> Fn=<filename> <SID_level_of_the_file> L#=<line number>
```

An extract from the /var/adm/SPlogs/SPdaemon.log file is shown here:

```
Mar 4 20:08:58 sp4n01 Worm[10656]:
LPP=PSSP,Fn=TBSrecovery.c,SID=1.42,L#=1257,
```

11.4.1.1 Installing and Configuring the Error Log

Log management functions are built upon the sysctl facility, which uses the SP authentication services. Generating parallel AIX Error Log and BSD syslog reports and performing general log viewing require 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 this file

Ex.1010 / Page 351 of 572

must also be placed in the authentication (PSSP or AFS) database as principals.

Important

Log management mostly consists of administrative tasks normally requiring root authority, and requires a user defined in the logmgt.acl file to execute commands as the root user.

Here is an example of the /etc/logmgt.acl file:

```
[sp4en0:/]# cat /etc/logmgt.acl
#acl#
#
# This sample acl file for log management commands
# contains a commented line for a principal
#
#_PRINCIPAL root.admin@HPSSL.KGN.IBM.COM
# Principal for trimming SPdaemon.log by cleanup.logs.ws
#
_PRINCIPAL rcmd.sp4en0
_PRINCIPAL root.admin@MSC.ITSO.IBM.COM
[sp4en0:/]#
```

The file contains entries for the user root as principal root.admin and giving root the authority to execute log management commands.

The log management server functions executed by sysctl are located in /usr/lpp/ssp/sysctl/bin/logmgt.cmds. During system initialization, an include statement for this file is added to the default systctl configuration file /etc/sysctl.conf.

If you want to use an alternate sysctl configuration file, it must be updated with a statement to include the logmgt.cmds file, and the sysctld daemon must be restarted to activate the change.

Detailed information on configuring the error logs can be found in the redbook *RS/6000 SP: Problem Determination Guide*, SG24-4778, and in *IBM Parallel System Support Programs for AIX: Administration Guide*, SA22-7348.

11.4.1.2 SP Error Logs

The SP System uses the standard logs provided by both AIX and the public domain software it includes, as well as SP-specific logs. Some logs reside on the control workstation only, and some reside only on the SP nodes. Others

Ex.1010 / Page 352 of 572

reside on both. A summary of the log files and their locations is shown in Chapter 4, “Error Logging Overview” of *IBM Parallel System Support Programs for AIX: Diagnosis Guide*, GA22-7350.

You may be asked to reference some of these files and send them to your IBM Support Center representative when diagnosing RS/6000 SP problems.

11.4.2 Problem Management (PMAN) Subsystem

The PMAN subsystem provides an infrastructure for recognizing and acting on problem events. PMAN is an Event Management (EM) client application, and is packaged with PSSP in the ssp.pman fileset and requires Kerberos authentication. The EM subsystem is discussed in 8.6, “Event Management (EM)” on page 200.

PMAN receives events from EM, and can react in one or more of the following ways:

- Send mail to an operator or administrator
- Notify all logged-on users via the `wall` command or opening a window on displays with a graphical user interface
- Generate a Simple Network Management Protocol (SNMP) trap for an enterprise network manager, such as Tivoli
- Log the event to AIX and BSD error logging
- Execute a command or script

This is represented diagrammatically in Figure 143.

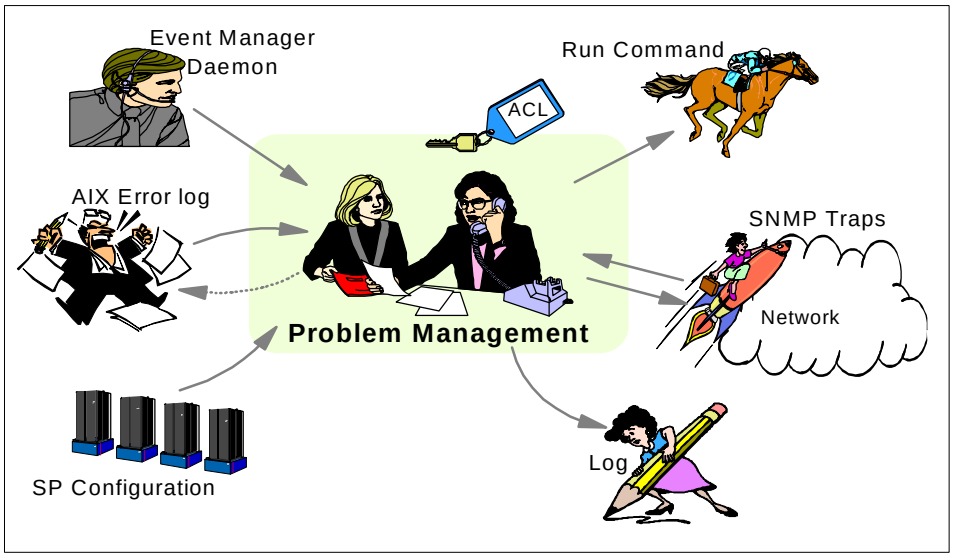


Figure 143. Problem Management Subsystem Design

Three daemons constitute the PMAN subsystem. The ways in which they inter-operate are shown in Figure 143. They are described as follows:

- **pmand** - This daemon interfaces directly to the EM daemon. The pmand daemon registers for events, receives them, and takes actions. PMAN events are stored in the SDR, and pmand retrieves them at initialization time.
- **pmanrmd** - If EM does not monitor the resource variable of interest, PMAN supplies its own resource manager daemon, pmanrmd, to access the resource variable's data. You can configure pmanrmd to periodically execute programs, scripts, or commands and place the results in one of EM's 16 user-defined resource variables. The pmanrmd daemon supplies the resulting data to the RMAPI. It also cooperates with pmand.
- **sp_configd** - This daemon creates SNMP traps from the event data in pmand, and is the interface for an enterprise network manager to access SP event, configuration, and resource variable data.

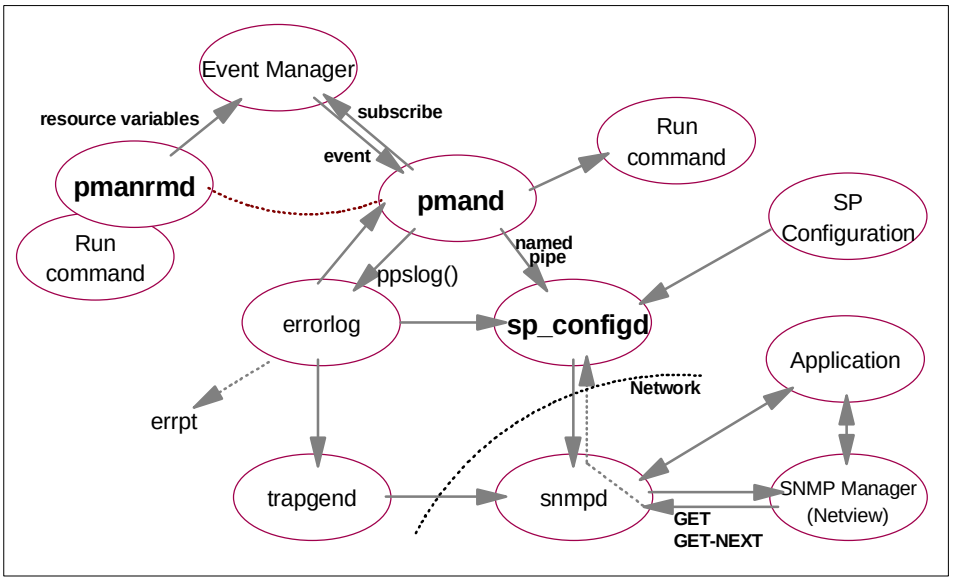


Figure 144. Problem Management Subsystem Daemons

The SP exploits AIX Error Notification Facility (AENF) to link the AIX error log and the PMAN subsystem. When a node is installed, an AENF object is created which sends all AIX Error Log entries to the pmand daemon. PMAN filters the entries based on subscriptions you define. The sp_configd daemon picks up SNMP-alertable AIX errors and passes them to snmpd for forwarding to the network manager (if this is installed in your environment).

By default, all PMAN daemons are started on all nodes and the control workstation. It is important to understand that PMAN is *not* a distributed application, but an Event Management (EM) client. The PMAN daemons on one node do not know about their counterparts on other nodes, and do not care. At initialization, each instance of pmand obtains PMAN event subscriptions from the SDR, so each node is aware of all PMAN events to be monitored. Although it is not mandatory to run the PMAN daemons everywhere, we do not recommend disabling them. PMAN must execute:

- Where you want an action to originate, given a specific event from EM
- Where you need custom resource monitoring, facilitated by the pmanrmd resource monitor

As an EM client, PMAN is an efficient infrastructure to respond to events across the system. For example, suppose your system-wide policy is to monitor for /tmp exceeding 95%. Instead of creating and maintaining the

required configuration on each node, use PMAN to subscribe to that event for all nodes. The EM distributed infrastructure monitors for you, and notifies your desired PMAN client (usually the node where the event occurs) to take appropriate action; furthermore, newly installed or re-installed nodes will automatically fall under the /tmp monitoring scheme. This is because the PMAN event is globally defined in the SDR, and PMAN is configured to run by default on all nodes.

The pmandefaults script sets the default events to be monitored, and is an excellent starting point for configuration of the PMAN subsystem. The script is documented in *IBM Parallel System Support Programs for AIX: Administration Guide*, SA22-7348. The script includes events of interest to most SP system environments, such as the following:

- The /var file system is more than 95% full.
- The /tmp file system is more than 90% 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 (Worm) daemon has terminated (nodes only).
- Problem with noncritical power loss or fan failure (nodes only).

The subscribed events will result in an event notification being mailed to the root user on the control workstation when the specified event occurs. Events are defined for all nodes in the current system partition, and all events are monitored from the control workstation. An extract from this script follows.

```
#
#  Watch /var space on each node in the partition
#
pmandef -s varFull  \
    -e 'IBM.PSSP.aixos.FS.%totused:NodeNum=*;VG=rootvg;LV=hd9var:X>95'\
    -r 'X<70'  \
    -c /usr/lpp/ssp/bin/notify_event  \
    -C "/usr/lpp/ssp/bin/notify_event -r"  \
    -n 0 -U root -m varFull
```

The syntax for this command is completely described in *IBM Parallel System Support Programs for AIX: Command and Technical Reference*, SA22-7351. The major flags you specify are:

-s — This flag specifies that this is a subscribe request and the remaining flags define the Problem Management subscription. The name of this subscription is varFull.

-e — This flag specifies the Event Management resource variable, resource identifier and expression that define the event for which actions are generated for this Problem Management subscription. You can review the Resource Variable Descriptions by using the `haemqvar` command. For example, this command can be used to see how the resource variable can be used:

```
haemqvar IBM.PSSP.aixos.FS IBM.PSSP.aixos.FS.%totused "*"
```

-r — This flag specifies the Event Management re-arm expression, which together with the resource variable, resource identifier and expression specified by the -e flag, defines the re-arm event for which actions are generated for this Problem Management subscription.

-c — This flag specifies a command to be executed when the event defined by the -e flag occurs. The command will be interpreted by the user's login program, so this command may contain additional arguments and shell metacharacters.

-C — This specifies a command to be executed when the re-arm event defined by the Nr flag occurs. The command will be interpreted by the user's login program, so the RearmCommand may contain additional arguments and shell metacharacters.

11.4.3 IBM Support Tools

There are some additional support tools available on RS/6000 SP environments.

11.4.3.1 Service Director

Service Director for RS/6000 can do automatic problem analysis and report hardware-related problems to IBM for service. Service Director is an application program that operates on all IBM RS/6000 machine types, including the RS/6000 SP. It is offered at no additional charge to customers as part of the IBM Warranty or IBM Maintenance Agreement.

Service Director is intended to be installed by trained IBM Customer Engineers/Service Support Representatives (CE/SSRs) or RS/6000 System Administrators. Basic AIX skills and the ability to use the System Management Interface Tool (SMIT) are required.

Service Director provides the following functions:

Automatic Problem Reporting — Reports problems to the IBM RETAIN (Problem Management System) via modem.

Manual Problem Reporting — Allows manual entry of problem information, then reports it to IBM RETAIN.

Automatic Problem Analysis — Product Support Applications (PSAs) will package additional information (if available) with the error information.

Definable Threshold Levels for Call Placement — Allows you to increase the threshold a specific error must reach before generating a service call. You can also tell Service Director to ignore specific errors or resources altogether.

Problem Reporting — Provides the ability for one RS/6000 to be the reporting server to IBM for all RS/6000s accessible from one of its TCP/IP networks. Provides a notify option that can send e-mail messages, informing users of Service Director activities.

Service Focal Point — Allows for a single focal point to view errors reported from any machine on the network.

Security — Service Director accesses only system data from diagnostics, system error logs, and Vital Product Data. The TTY configuration disables login capability on the port. The modem configuration disallows auto-answer. At *no* time does Service Director ever access customer data.

A customer-supplied modem and analog line is required. The automatic call function may not be offered in every country. Check with your IBM marketing representative for specific implementation conditions that apply in your country.

11.4.3.2 Gathering AIX Service Information

In many cases, IBM support personnel need to collect a substantial amount of detailed information to help isolate problems. To facilitate this process, AIX gives you the ability to collect service information for diagnosis by IBM Software Support. Included with the base AIX operating system is the snap command which assists you in compiling system configuration information quickly and easily. The snap command can be used to collect the following information:

- Configuration files
- Error and other log files
- Command line outputs

Once this information is compiled, you can view it and compress it for downloading to diskette or tape or for remote transmission. You may be asked by support specialists to execute the snap command to help them accurately identify your system problem.

Disk Space Requirements

Approximately 8 MB of temporary disk space is required when executing all of the snap options on an average system. If only one or two options are chosen, the disk space required will be substantially less, depending on the option. The program automatically checks for free space in the /tmp/ibmsupt directory or the directory specified with the -d flag. If there is not enough space, you will have to expand the file system. You can suppress this check for free space by using the -N option.

Output Directory

The default directory for the output from the snap command is /tmp/ibmsupt. If you want to name an optional directory, use the -d option with the path of the desired output directory. Each execution of the snap command appends to previously created files.

Options

The main options of the snap command are:

-g — Gathers the output of the `lslpp -L` command. Support specialists use the output to re-create your operating system environment if other problem determination techniques fail. The output is stored in /tmp/ibmsupt/general/lslpp.L.

Also, the -g flag gathers general system information and outputs it to /tmp/ibmsupt/general/general.snap.

-D — Gathers dump and /unix (assumes dump device to be /dev/hd7).

-a — Gathers information for all of the groups.

-c — Creates a compressed tar image of all files in the /tmp/ibmsupt directory tree (or other output directory).

Note: Other information that is not gathered by the snap command can be copied to the snap directory tree before executing the tar/compress option. For example, you may be asked by the support specialist to provide a test case that demonstrates the problem. The test case should be copied to the /tmp/ibmsupt directory. When the -c option of the snap command is executed, the test case will be included.

-o — Creates a tar file and downloads it to removable media.

-v — Displays the output of the commands executed by the snap command.

Before executing the snap -c or snap -o commands, any additional information required by the Support Center should be copied to the /tmp/ibmsupt/testcase directory (or an alternate directory).

The snap -c and snap -o commands are *mutually exclusive*. Do not execute both during the same problem determination session.

- The snap -c command should be used to transmit information electronically.
- The snap -o command should be used to transmit information on a removable output device.

11.4.3.3 Gathering SP-Specific Service Information

Specialized “snap-like” utilities are provided with PSSP software to collect log files and diagnostic information for specific SP hardware and/or software components. The generated compressed tar files can then be sent to your IBM Support Center representative for problem diagnosis and resolution.

css.snap

The css.snap script collects log files created by switch support code such as device drivers, the Worm, diagnostic outputs and so on, into a single package.

The completed output file is found in the /var/adm/SPlogs/css directory and will have the naming convention of css.snap.<date-time>.tar.Z.

The css.snap script is called automatically from the fault service daemon when certain serious errors are detected. However, it can also be issued from the command line when a switch or adapter related problem is indicated with:

```
# /usr/lpp/ssp/css/css.snap
```

Important

Be aware that css.snap uses a number of undocumented utilities to collect the information required. Some of these, such as the *read_regs* and *tbXdump* routines, can have a disruptive effect when used on a running system.

After running css.snap to collect diagnostic information, it is advisable to run /usr/lpp/ssp/css/rc.switch in order to reset or reload the switch adapter and eliminate the residual effects of these utilities.

Ex.1010 / Page 360 of 572

vsd.snap

The vsd.snap script collects the data necessary to report SP VSD & RVSD related problems. There are a number of flags you can use to limit the considerable output that is generated by the default -w ALL option. The vsd.snap syntax is:

```
vsd.snap [-w VSD | -w RVSD | -w CFG | -w ALL] [-d output_directory]
```

where -w is used to select output information, and -d is used to specify the output directory; the default is /tmp/vsd.snapOut. The output file generated (a compressed tar file) has the filename format:

```
vsd.snap.node_number.xxxxxxxx.out.tar.Z
```

where xxxxxxxx is a timestamp. You must ensure that there is sufficient free space available in the target file system for the output file to be created, otherwise the command exits with an appropriate message.

Chapter 12. User Management

One of the challenges for the SP system administrator is managing the users in the system. Should a user be allowed to access to one node, some nodes or all nodes? The SP can be viewed as one logical unit, therefore users need to be defined across all nodes, with the same login name, the same login password, the same group characteristics, the same home directory and so on. This is only achieved by sharing the user database across the SP system. An SP system administrator therefore has the responsibility of maintaining consistent copies of files such as /etc/passwd, /etc/group and /etc/security/passwd across the nodes.

There are three methods available to maintain a consistent user database:

1. Manage each user individually over each node in an SP system. This is a time consuming effort because the system administrator has to be aware of any changes (such as a password change) to the user database on every node. Once a change is implemented on one node, the system administrator has to then update every node in the system.
2. Use the Network Information System (NIS), available with AIX. NIS is a three-tiered system of clients, servers and domain in which any changes to a set of defined common files (such as /etc/passwd and /etc/group) are automatically propagated throughout the system.
3. Use the SP File Collections facility provided with the PSSP software. File Collections defines and maintains sets of common files which are updated throughout the system at regular intervals.

It is recommended that SP system administrators choose method 2 or 3 to maintain a user database. Both options provide the system administrator the ability to manage the user database from a single point of control, for example, the control workstation. In an SP system with a large number of users and/or a large number of nodes, options 2 and 3 may be the only feasible choices. Option 1 requires a large manual effort, while options 2 and 3 make user management much more automated.

Two types of users can reside within an SP system. AIX users are those created through AIX on an individual node and reside only on that node. SP users are created through SP user management (SPUM) and can have access to every node. It is possible to have both types of users on a given node. This makes it difficult, if not impossible, to use file collections and NIS to manage the user database because the user database on each node is different from the other nodes. File collections and NIS are designed to manage one consistent copy of the user database across the system.

© Copyright IBM Corp. 1999

SP access control (SPAC) is provided with the PSSP software to give you the capability to use NIS or file collections to maintain a user database and restrict user login on particular nodes.

The automounter is a subsystem that mounts file systems on demand when they are first referenced and subsequently unmounts them after a period of inactivity. Its use enables users to maintain one home directory across the SP nodes.

This chapter begins with a look into the files within the AIX operating system which make up the user database. We then examine SPUM, SPAC and NIS. We conclude with a look at file collections and the automounter.

12.1 Managing User Databases

The purpose of user management is to maintain a user database across all nodes in an SP system. This enables the users to use the SP as one logical machine, despite its makeup of multiple, individual RS/6000s. User management tasks include:

- Adding user accounts
- Deleting user accounts
- Changing user account information, including the login password
- Listing user account information
- Controlling user logins

There are distinct sets of commands for managing AIX and SP users. To manage AIX users, use the AIX commands `mkuser`, `rmuser`, `chuser` and `lsuser`. To manage SP users, the PSSP software provides the SP user management (SPUM) commands. The two sets of command perform similar functions; the difference is the type of users they manage. SPUM is discussed in 12.1.2, “SP User Management (SPUM)” on page 341.

12.1.1 AIX: files (local)

To AIX, a user is defined by the existence of an entry in the /etc/passwd file. A user’s password is kept in encrypted format in the /etc/security/passwd file. Users may be grouped together; group information is kept in the /etc/group and /etc/security/group files. Together, these four files form the foundation of an AIX user database.

Ex.1010 / Page 364 of 572

Optional files to further tailor user accounts include /etc/security/user, /etc/security/limits, /etc/security/environ, /etc/security/login.cfg, /etc/passwd.nm.idx, /etc/passwd.id.idx and /etc/security/passwd.idx.

The *.idx files are password index files used to improve login performance.

The other files in /etc/security may or may not be used depending upon your system's requirements. A more in-depth look into user login control using the AIX files can be found in 7.2.4, "Login Control in AIX" on page 147.

If any of these optional files are used, they need to be included in the user database to be replicated across the system.

12.1.2 SP User Management (SPUM)

The PSSP software provides SP user management (SPUM) commands to enable an SP administrator to manage SP users. SPUM commands add and delete SP users, change account information, and set defaults for SP users' home directories. There are four SPUM commands: `spmkuser`, `sprmuser`, `spchuser` and `splsuser`. You do not have to use SPUM if you are familiar with the AIX commands and can manipulate them to execute the same functions as the SPUM commands.

You can enable SPUM both during and after the installation process. It is also possible to change the default SPUM settings after enablement. This is done through SMIT panels or the use of the `spsitenv` command. For the SMIT panel, run `smit enter_data` and select **Enter Site Environment Information**. Figure 145 on page 342 shows this SMIT panel.

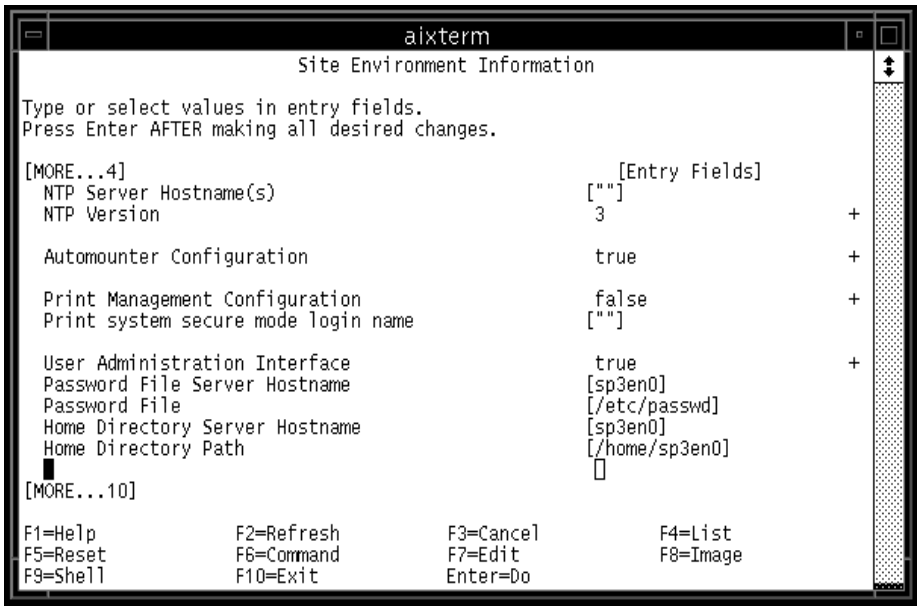


Figure 145. SMIT Panel for Controlling SPUM Settings

There are five fields for configuring SPUM:

- **User Administrative Interface** indicates whether you want to use SPUM or not.
- **Password File Server Hostname** indicates the hostname of the server which has the master password file. The default is set to the control workstation. You can set up another machine to serve the master password file, however, this machine cannot be one of the SP nodes.
- **Password File** is the path for the master password file. The default is /etc/passwd, the file that AIX uses. If you are using either file collections or NIS, you can choose to use another file.
- **Home Directory Server Hostname** is the name of the machine where the users' home directories reside. The default is the control workstation, although it can be set to any machine both inside and outside of the SP system. However, if the control workstation is not used, you need to ensure that security authentication has been set up. For the node, follow the authentication setup steps detailed in "Security Features of the SP System" in *IBM Parallel System Support Programs for AIX: Administration Guide*, SA22-7348. For a machine outside the SP system, you need to consider whether it is within the authentication realm of the SP or not. If

this machine is in the same realm, update the .klogin file of the machine to include the administrator's principal name. If not, you need to include a .rhosts file on the machine to permit the root user from other machines to perform the necessary actions against the home directories, for example, create a new directory when a new user is created.

- **Home Directory Path** is the default path of users' home directories. The default is /home/<control workstation name> and is stored in the SDR.

To add an SP user, you can use SMIT or the command `spmkuser`. To use SMIT, run `smit spmkuser`. Figure 146 shows this SMIT panel.

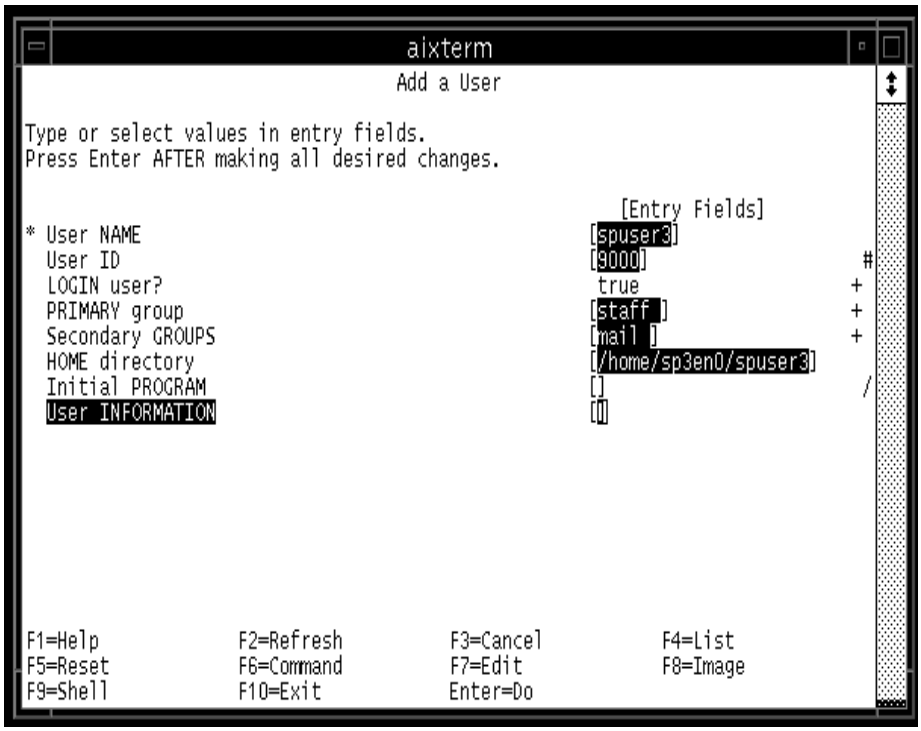


Figure 146. SMIT Panel for Adding SP Users

The only mandatory field is the name of the user. If all other fields are left blank, the system generates its own user ID, places the user in the staff group, does not put it into any secondary groups and creates a home directory based on the defaults specified when SPUM is turned on. If you specify a home directory path that is different than the default, this entry is used.

When a new user is added, the system generates a random password for the user and stores it in /usr/lpp/ssp/config/admin/newpass.log. This needs to be communicated to the user. The password in this file is not erased, even after the user has made changes to the password. It is therefore recommended that the contents of this file be deleted on a regular basis.

To delete users, you can use SMIT or `sprmuser`. To use SMIT, run `smit sprmuser`. Figure 147 shows this SMIT panel.

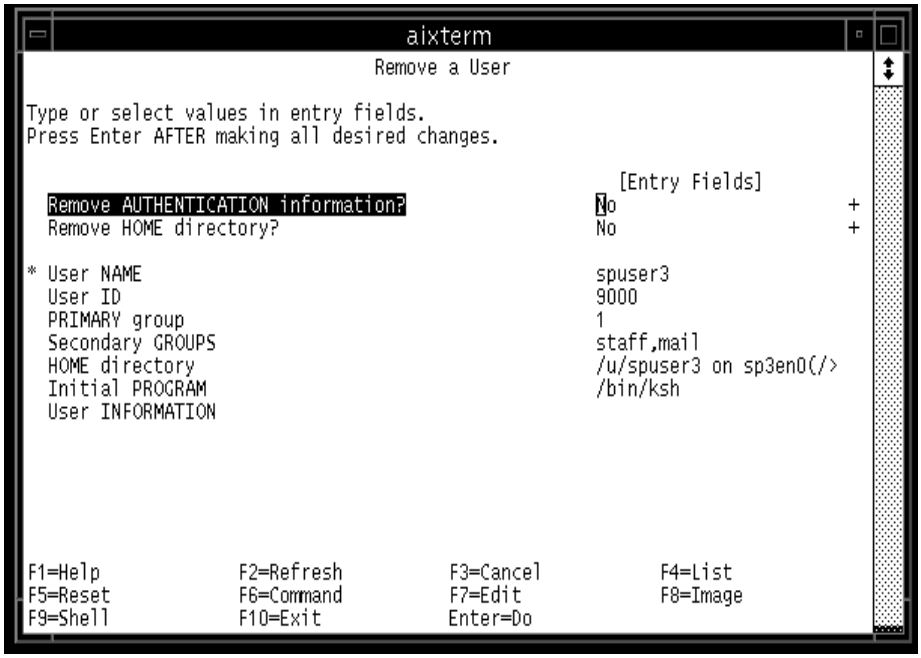


Figure 147. SMIT Panel for Deleting SP Users

The important options are the first two: **Remove AUTHENTICATION information?** and **Remove HOME directory?**. These are, by default, set to no to preserve data should the user need to be created again.

To change a user's information, you can again use SMIT or the command spchuser. Run `smit spchuser` to access the SMIT panel. Figure 148 on page 345 shows this SMIT panel.

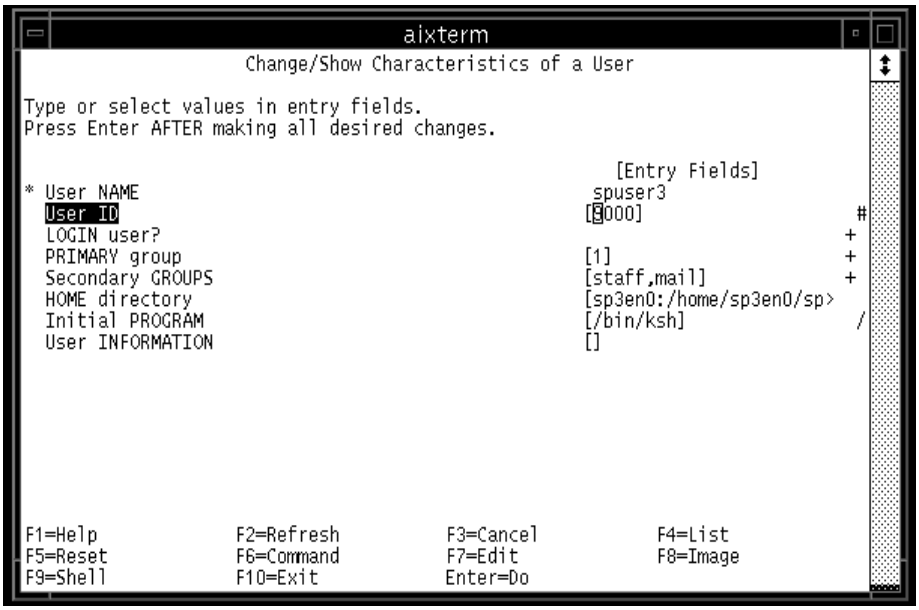


Figure 148. SMIT Panel for Changing SP User Information

Changes that you make here to an SP user can override the default settings specified when SPUM is initially enabled.

To list an SP user account and its information, you can run `splsuser` or use SMIT by running `smit spchuser`. The same panel is used for listing and changing user information.

When SPUM is enabled, user password changes are handled in one of two ways, depending on whether NIS is used or not. If NIS is not in use, PSSP assumes that file collection is in use and restricts password changes to the machine that has the master copy of the /etc/passwd file. SPUM does this by linking the AIX password commands to SP password commands on the nodes. The commands that are modified are:

- /bin/chfn, which is linked to /usr/lpp/ssp/config/sp_chfn
- /bin/chsh, which is linked to /usr/lpp/ssp/config/sp_chsh
- /bin/passwd, which is linked to /usr/lpp/ssp/config/sp_passwd

The original AIX files are copied to new files which have the same name, but end in .aix. The SP password files tell users to log in to the appropriate machine to change their passwords.

User password changes when NIS is in use are discussed in 12.2, “Network Information System (NIS)” on page 348.

Attention

If both file collections and NIS are not used, but SPUM is enabled, user password changes are still restricted to the machine with the master /etc/passwd file. However, in this instance, after a change has been made, the system administrator needs to propagate the files /etc/passwd and /etc/security/passwd across the SP system.

12.1.3 SP Access Control (SPAC)

You may want to run either file collections or NIS to maintain a user database, but have a requirement to restrict SP user access on nodes. Or, if your SP system is used for a parallel application, there may be a need to restrict user login on the processing nodes to improve performance. In either case, you can use SP access control (SPAC). At the heart of SPAC is the command `spacs_cntrl`, which is run by the SP system administrator or by a job submission program on nodes where login control is required.

SPAC makes use of login and rlogin attributes in the /etc/security/user file to control a user’s ability to log into particular nodes. It sets the attributes to either true or false depending on whether you want users to login (true) or not (false). The login attribute is used to control user local (serial line) logins while rlogin controls remote (network) logins. If you are using file collections, be sure to remove /etc/security/user from any file collections because it is unique among nodes. If you are using NIS, no action is needed because NIS by default does not handle this file.

Figure 149 on page 347 is a user’s default setting as specified in /etc/security/user. Notice that the login and rlogin attributes are set to **true**.

Ex.1010 / Page 370 of 572



Figure 149. Sample of an Entry in /etc/security/user

SPAC controls the login and rlogin attributes by using `spacs_cntrl` with four keywords: block, unblock, allow and deny.

The block and unblock keywords are used by a system administrator to control login capabilities on a node. When a user is in block state on a node, both login and rlogin are set to equal false and all logins are disallowed. This is done by running `spacs_cntrl block <user>`. Similarly, if an user is in unblock state, login and rlogin are set to equal true and all logins are allowed. This is done by running `spacs_cntrl unblock <user>`.

Users may be controlled by `spacs_cntrl` individually or in a group. If you want to submit a group of users, run the script /usr/lpp/ssp/samples/block_usr_sample to build a file /tmp/usr.input which contain user IDs that can be blocked. For further details, refer to Chapter 6, "Managing User Accounts" in *IBM Parallel System Support Programs for AIX: Administration Guide*, SA22-7348.

The allow and deny keywords are used by a job submission program to control a user's state while executing a parallel job. The keywords are used on a transaction basis to update a user's state. A user can be in one of the following four states:

- Allowed login
- Denied login
- Allowed login after a deny request
- Denied login after an allow request

If a user's state is requested to change more than once using the allow and deny keywords, the file spacs_data is created to keep track of outstanding requests. When a change request is submitted, a check is made against the spacs_data file. If the request is the same as what is in the file, the request is not stored; instead, a counter is incremented. If the next request is opposite to what is in spacs_data, the counter is decremented, the user removed from spacs_data, and the /etc/security/user file is updated to reflect the change.

When a job submission program is using the allow and deny keywords to control user login on nodes, be careful that the system administrator does not run `spacs_cntrl block` or `spacs_cntrl unblock` on the same node. The block or unblock state automatically causes `spacs_cntrl` to clear the contents in the spacs_data file.

Further information for SPAC is in *IBM Parallel System Support Programs for AIX: Administration Guide*, SA22-7348.

12.2 Network Information System (NIS)

The main purpose of NIS is to centralize administration of files like /etc/passwd within a network environment.

NIS separates a network into three components: domain, server(s) and clients.

A NIS domain defines the boundary within which file administration is carried out. In a large network, it is possible to define several NIS domains to break the machines up into smaller groups. This way, files meant to be shared among for example, five machines, stay within a domain that includes the five machines, not all the machines on the network.

A NIS server is a machine that provides the system files to be read by other machines on the network. There are two types of servers: Master and Slave.

Ex.1010 / Page 372 of 572

Both types keep a copy of the files to be shared over the network. A master server is the machine where a file may be updated. A slave server only maintains a copy of the files to be served. A slave server has three purposes:

- To balance the load if the master server is busy.
- To back up the master server.
- To enable NIS requests if there are different networks in the NIS domain. NIS client requests are not handled through routers; such requests go to a local slave server. It is the NIS updates between a master and a slave server that go through a router.

A NIS client is a machine which has to access the files served by the NIS servers.

There are four basic daemons that NIS uses: ypserv, ypbind, yppasswd and ypupdated. NIS was initially called yellow pages, hence the prefix yp is used for the daemons. The daemons work in the following way:

- All machines within the NIS domain run the ypbind daemon. This daemon directs the machine's request for a file to the NIS servers. On clients and slave servers, the ypbind daemon points the machines to the master server. On the master server, its ypbind points back to itself.
- The ypserv daemon runs on both the master and the slave servers. It is this daemon that responds to the request for file information by the clients.
- The yppasswd and ypupdated daemons run only on the master server. The yppasswd daemon makes it possible for users to change their login passwords anywhere on the network. When NIS is configured, the /bin/passwd command is linked to the /usr/bin/yppasswd command on the nodes. The yppasswd command sends any password changes over the network to the yppasswd daemon on the master server. The master server changes the appropriate files and propagates this change to the slave servers using the ypupdated daemon.

Important

NIS serves files in the form of maps. There is a map for each of the file that it serves. Information from the file is stored in the map, and it is the map that is used to respond to client requests.

By default, the following files are served by NIS:

- /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

Attention

By serving the /etc/hosts file, NIS has an added capability for handling name resolution in a network. Refer to *Managing NIS and NFS* by O'Reilly and Associates for detailed information.

To configure NIS, there are four steps, all of which can be done via SMIT. For all four steps first run `smit nfs` and select **Network Information Service (NIS)** to access the NIS panels, then:

Step 1 Choose **Change NIS Domain Name of this Host** to define the NIS Domain. Figure 150 on page 351 shows this SMIT panel. In this example, SPDomain has been chosen as the NIS domain name.

Ex.1010 / Page 374 of 572

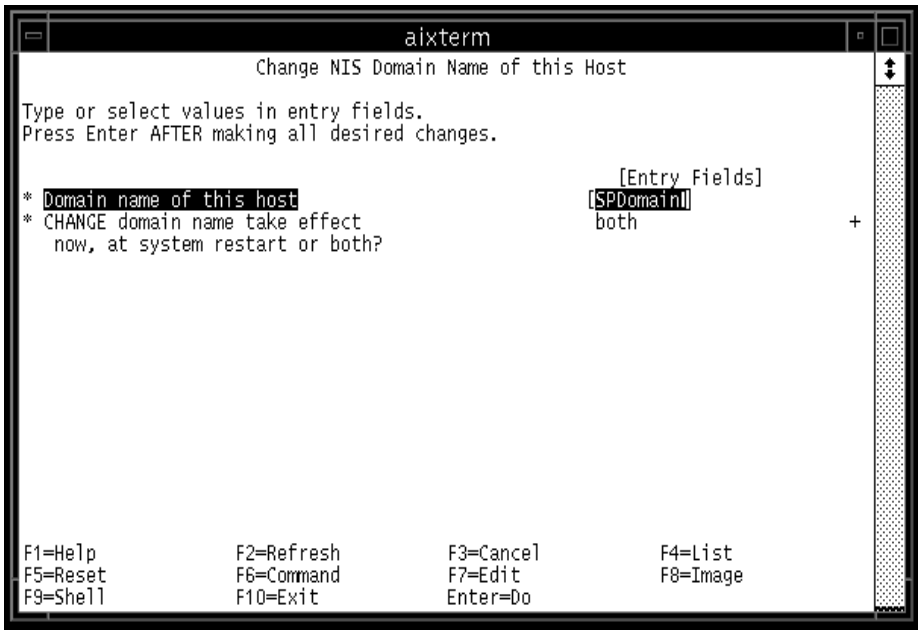


Figure 150. SMIT Panel for Setting a NIS Domain Name

Step 2 On the machine that is to be the NIS master (for example, the control workstation), select **Configure/Modify NIS** and then **Configure this Host as a NIS Master Server**. Figure 151 on page 352 shows the SMIT panel. Fill in the fields as required. Be sure to start the yppasswd and ypupdated daemons. When the SMIT panel is executed, all four daemons (ypbind, ypserv, yppasswd and ypupdated) are started on the Master server. This SMIT panel also updates the NIS entries in the local /etc/rc.nfs file.

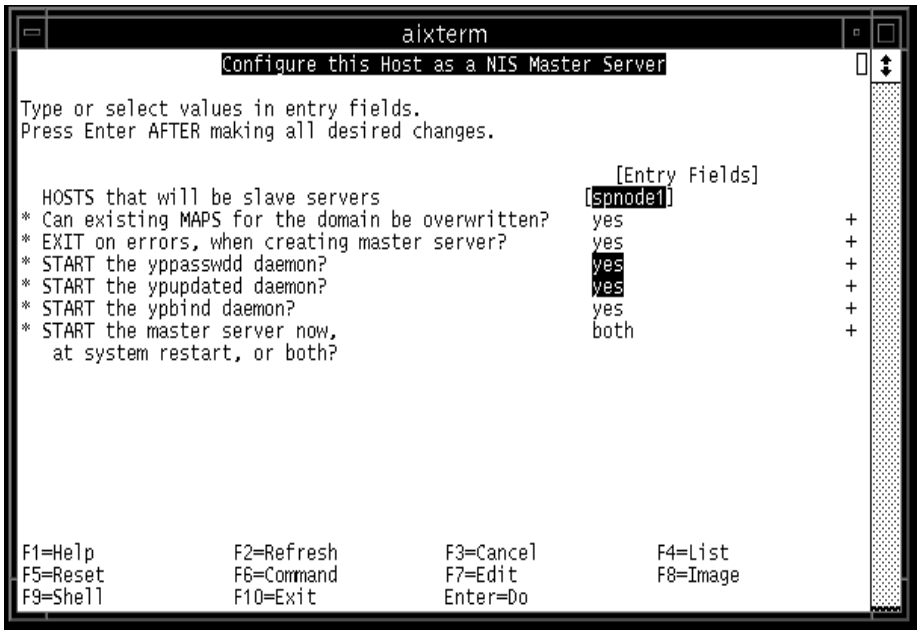


Figure 151. SMIT Panel for Configuring a Master Server

Step 3 On the machines set aside to be slave servers, go to the NIS SMIT panels and select **Configure this Host as a NIS Slave Server**. Figure 152 on page 353 shows the SMIT panel for configuring a slave server. This step starts the ypserv and ypbind daemons on the slave servers and updates the NIS entries in the local /etc/rc.nfs file(s).

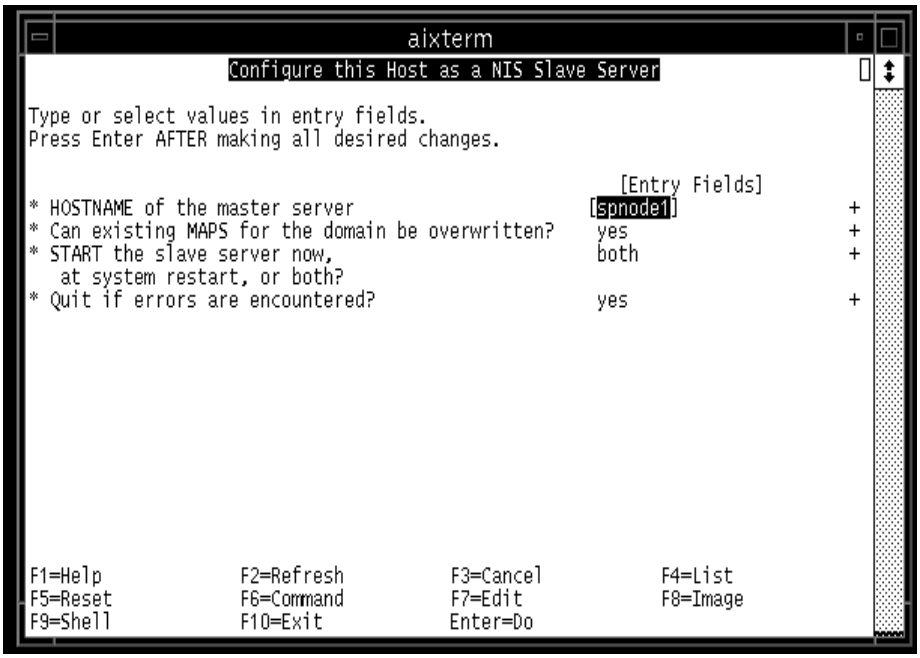


Figure 152. SMIT Panel for Configuring a Slave Server

Step 4 On each node that is to be a NIS client, go to the NIS SMIT panels and select **Configure this Host as a NIS Client**. This step starts the ypbind daemon and updates the NIS entries in the local /etc/rc.nfs file(s). Figure 153 on page 354 shows this SMIT panel.

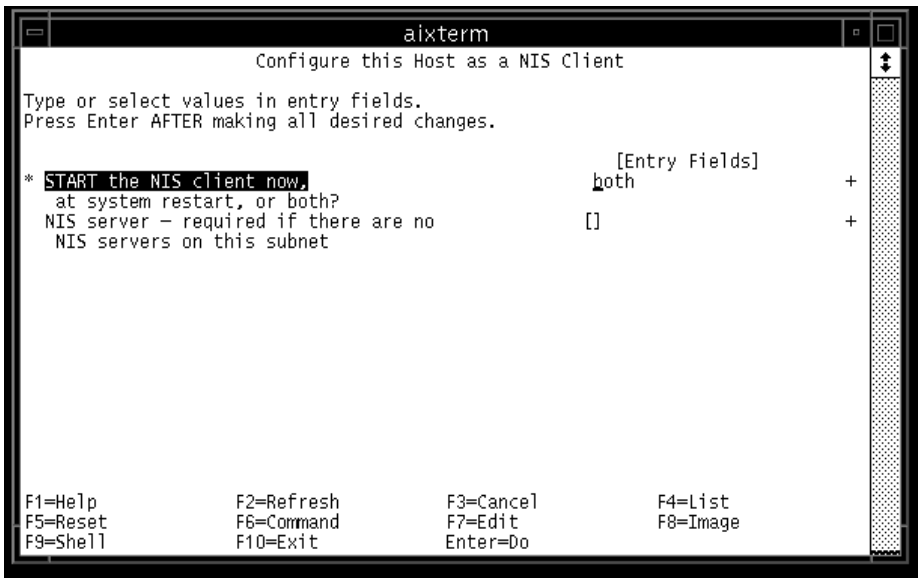


Figure 153. SMIT Panel for Configuring a NIS Client

Once configured, when there are changes to any of the files served by NIS, their corresponding maps on the master are rebuilt and either pushed to the slave servers or pulled by the slave servers from the master server. These tasks are done via the SMIT panel or the command `make`. To access the SMIT panel, select **Manage NIS Maps** within the NIS panel. Figure 154 on page 355 shows this SMIT panel.

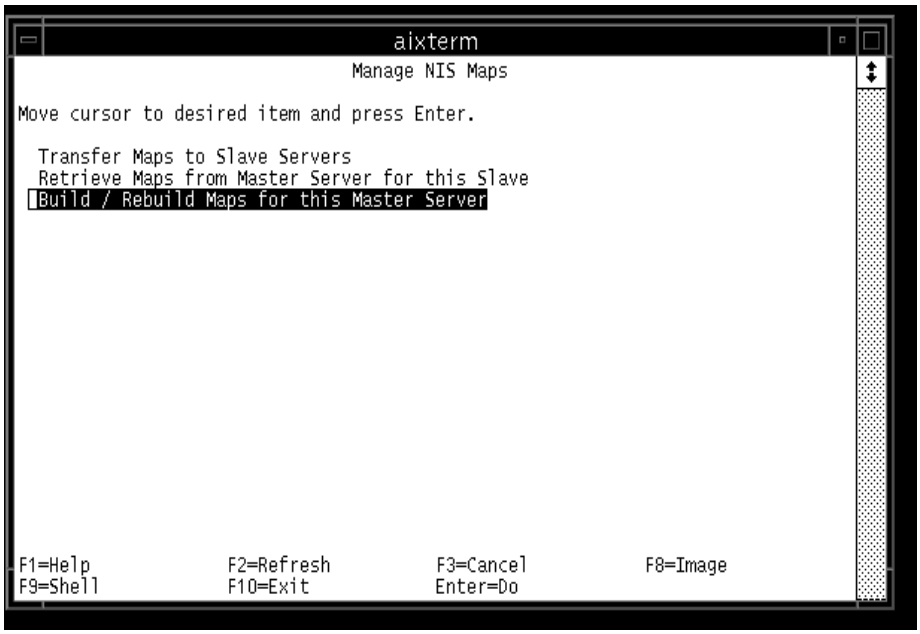


Figure 154. SMIT Panel for Managing NIS Maps

Select **Build/Rebuild Maps for this Master Server** and then either have the system rebuild all the maps with the option **all**, or specify the maps that you want to rebuild. After that, return to the SMIT panel shown in Figure 154 and select either **Transfer Maps to Slave Servers** (from the master server) or **Retrieve Maps from Master Server for this Slave** (from a slave server).

12.3 File Collections

The File Collections facility is included with PSSP to group files and directories on multiple nodes into sets known as file collections. This simplifies their maintenance and control across an SP system. File Collections is installed by default on an SP system.

To turn off File Collections, run `smit enter_data`, select **Site Environment Information**, and choose **false** for the field **File Collection Management**. Figure 155 on page 356 shows this SMIT panel.

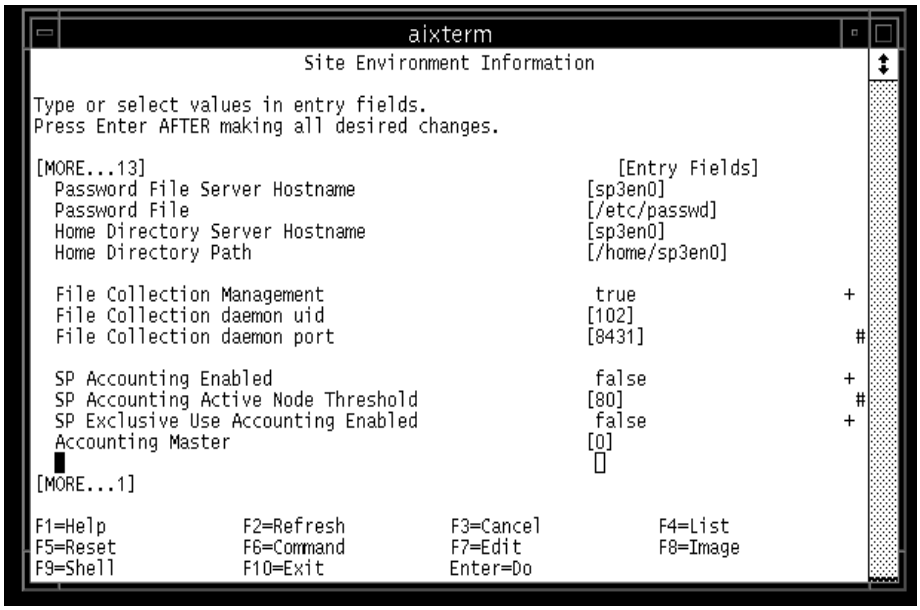


Figure 155. SMIT Panel for Setting File Collections

File collections are managed by the Perl program called supper, which in turn is based on the Software Update Protocol (SUP) and can be run as a command. A file collection has to be defined to supper so supper can recognize and maintain it. supper interacts with the file collection daemon supman to manage the file collections. supman is also installed as a unique userid for file collection operations and requires read access permission to all files that are to be managed as part of a file collection.

A file collection can be one of two types: primary or secondary. A primary file collection can contain a group of files or a secondary file collection. When a primary file collection is installed, the files in this collection are written to a node for usage. What if you want to use a node to serve a file collection to other nodes? This is made possible by using a secondary file collection. When a secondary file collection is installed on a node, its files do not get executed on the node, rather they are stored ready to be served out to other nodes.

File collections have two possible states: Resident or Available. A resident file collection is a group of files that are installed in their true locations and can be served to other systems. An available file collection is one that is not installed in its true location but is able to be served to other machines.

An installed secondary file collection can only be in the available state.

File collections uses a hierarchical structure to distribute files to nodes. The control workstation is defined by default to be the Master server, which is the machine where the "master" copy of a file is kept. It is only at this location that a file may be changed. Files are then distributed from the master server to all the defined boot/install servers to be distributed out to the nodes. If no boot/install servers are defined, file collections are served by the control workstation since the control workstation itself is a boot/install server.

You may change this hierarchy and use a boot/install server as a master server for one or some of the file collections. This way, you can maintain different copies of files on different groups of nodes. To implement this, you run `supper offline` on a boot/install server against a file collection. This prevents that file collection from being updated by the control workstation. Changes specific to the group of nodes served by the boot/install server can now be made on the boot/install server.

Attention

Password Changes

Recall that if NIS is not running, the password control files are changed so that all password updates are done on the master server. This may be a problem because you may not want all users to have access to the control workstation.

A workaround is to use the script cw_restrict_login, located in /usr/lpp/ssp/config/admin, to allow uses to log into the control workstation only for the purposes of changing their passwords. The details for running this script can be found in Chapter 6, "Managing User Accounts" in *IBM Parallel System Support Program for AIX: Administration Guide*, SA22-7348.

12.3.1 Predefined File Collections

Four predefined file collections are shipped with PSSP:

- sup.admin

  This file collection contains the files that define other file collections and the programs that load and manage file collections.

- user.admin

This file collection contains the AIX user account files: /etc/passwd, /etc/group, /etc/group, /etc/security/passwd, /etc/security/group and the login performance files /etc/passwd.nm.idx, /etc/passwd/id.idx, /etc/security/passwd.idx.

- power_system

  This file collection contains files that are system dependent. By default, it only contains the node.root collection.

- node.root

  This file collection contains files which are specific to the nodes. An example is a .profile different from the one on the control workstation that you want all users to use on the nodes.

The first three file collections are primary and resident on all nodes and boot/install servers. They are available to be served by the control workstation and boot/install servers. The node.root collection is a secondary file collection stored within the power_system file collection, available to be served by the control workstation and boot/install servers and resident on boot/install servers and nodes.

Attention

File Collections and NIS

It is possible to run both NIS and file collections, because NIS only handles the system administration files while file collections can be configured to handle other files. In this case, simply configure the user.admin file collection to not handle the user administration files of /etc/passwd,

12.3.2 File Collections File Structure

The file collection files are stored under /var/sysman. In particular, the master files which define and control file collections are stored in /var/sysman/sup. Each file collection has its own directory within /var/sysman/sup to stores its unique characteristics. The actual files that make up the file collection may be stored elsewhere on the machine.

Figure 156 on page 359 summarizes the directory structure.

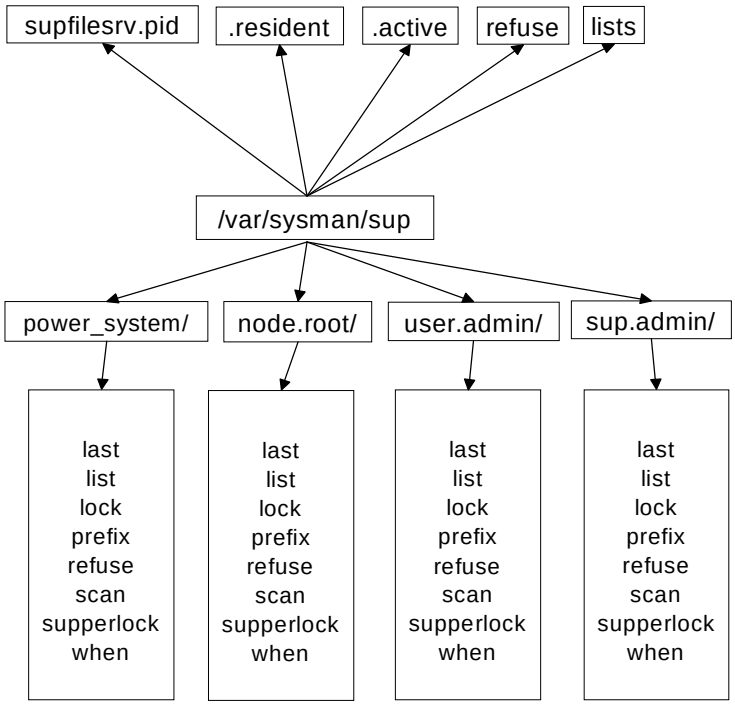


Figure 156. Summary of the File Collection Directory Structure

The files under /var/sysman/sup do the following:

- supfilesrv.pid holds the PID of the supfilesrv daemon, which serves the file collections on the system.
- .resident stores the names of all the file collections that can be served.
- .active describes the active volume group.
- refuse is a file that is useful on client systems which specifies the files to be excluded from file collection updates.
- lists is a file containing links to each of the individual file collection subdirectories.
- power_system, node.root, user.admin and sup.admin are all subdirectories set up to store the information regarding each file collection.

Within each file collection subdirectory, there is a set of “master files” which define the file collection:

last — is maintained by the system to hold a list of files and directories that have been updated.

list — holds the details of files that belong to the file collection. It is not necessary to define each individual file, you can use rules or wild cards. It is also not necessary to specify the full path of the files, just the path relative to the directory specified in the prefix file is sufficient.

lock — is an empty file that SUP uses to lock a collection and prevent multiple updates on a single file collection at one time.

prefix — specifies the starting point that supper uses when scanning the files that belong to a file collection.

refuse — is a system-generated file that contains a list of the files to be excluded from a supper update. The system creates this file based on the entries in the /var/sysman/sup/refuse file.

scan — is a system-created file that contains the list of files which make up the file collection, together with their permissions and time stamps. This file is populated when supper runs the scan process.

supper lock — is similar to the lock file and is used by supper to lock the file collection during updates.

when — is a system-maintained file that has the time of the last file collection update. It protects against accidental re-writes of files with older versions.

12.3.3 How File Collections Work

Here is how supper uses the file collection files. It can run one of three processes against a file collection: scan, install or update.

When supper runs the scan, it begins at the directory specified by the prefix file and traverses the directory tree to find all the files that meet the criteria in the list file. This process produces the scan file. The scan file is optional for the other supper processes. However, its presence can improve their performance since they no longer have to execute the equivalent of a scan.

The install process takes a file collection and copies the files onto the target machines for use, while the update process takes any changes in the files of a file collection and propagates them throughout the system. Both processes read from the scan file, if present, to identify the files to install or update. If a scan file is not present, supper performs a search to identify the files to install or update. An install or update also checks the /var/sysman/sup/refuse file to see if there are files to be excluded from the process.

Ex.1010 / Page 384 of 572

During a `supper update`, the lock and supper lock files are used to block other processes from changing the files in the file collection. In addition, the subcommand checks the when file to ensure that it is not updating with an older version of the files.

Attention

The `supper update` process is a “pull” process. It is run from the clients to extract updated files from the master. For example, if a user changes their password on the control workstation, it is a node’s responsibility to "pull" this change from the control workstation. Unlike NIS, the control workstation does not "push" a change.

When file collections is first installed, the supper update process is automatically included in the crontab file on the nodes in the SP. The frequency for an update is once per hour. If you want to change this, simply modify the crontab entry.

For the purpose of system administration, the process to run most frequently is the update process, to ensure that the files in a file collection are kept up-to-date. When you are updating files, make sure that you are writing to the copy of the file on the master server. Once the file is updated, run `supper scan <file collection name>` on the master server in case there have been changes to the number of files making up the file collection. Then, run `supper update <file collection name>` on the nodes to make sure that the nodes receive the change.

It is possible to run supper with a series of subcommands to extract information regarding a file collection. These are detailed in both *IBM Parallel System Support Programs for AIX: Administration Guide*, SA22-7348 and *IBM Parallel System Support Programs for AIX: Command and Technical Reference*, SA22-7351.

12.3.4 Refusing Files in a File Collection

It is possible to exclude certain files from particular nodes during an update by `supper`. To do so, add the names of the specific files to the refuse file on the node’s /var/sysman/sup directory. When `supper update` is run on the nodes, it checks /var/sysman/sup/refuse to see what files it needs to exclude from the update being executed.

As the files listed are passed over for update, the system writes the names of these files into /var/sysman/<file collection name>/refuse.

It may seem confusing to have two refuse files. However, this gives you the ability to confirm that the system has excluded the files you specified. Remember that you are to write only to /var/sysman/sup/refuse. The file /var/sysman/<file collection name>/refuse is written by the system and available for you to read to confirm that those are the files you requested to be excluded from the update.

12.3.5 Adding or Deleting Files in a File Collection

What if you want to add or delete a file within a file collection? These two situations require special considerations because you have to update the proper files in the file collection's directory.

The first step is to make sure the file is added or deleted on its server. You need to keep in the mind the exact location of the file relative to the directory that is specified in the prefix file. If you are adding a file, ensure that the file is readable by everyone else on the machine.

The second step is to run `supper scan` to update the scan file. This step may be optional since the scan file is not a requirement in a file collection. However, it is recommended that you do run this command because it increases the performance of the next command on large systems.

Run `supper update` on nodes to make this change effective.

12.3.6 Building a File Collection

To illustrate how to build a file collection, we are going to use an example where we create a file collection called samplefc. It includes four files in the /sample directory: file1, file2, file3 and file4.

There are seven steps to building a file collection:

1. Identify the files that you want to collect. In the example, the files are file1, file2, file3 and file4.
2. Create a File Collection directory on the master server and make sure that it is owned by bin and in the group bin. We therefore run:

   ```
   cd /var/sysman/sup
   mkdir samplefc
   chown bin samplefc
   chgroup bin samplefc
   ```

3. Create a list file in this directory to state the rules for including or excluding files in the file collection. The easiest way is to copy an existing list file

Ex.1010 / Page 386 of 572

from another file collection and modify it to suit your needs. We copy and then edit the list file from another file collection to include only the line

```
always ./file*
```

This line in the list file instructs `supper` to always upgrade the files beginning with the word file.

4. Add a link to this file collection directory in the file /var/sysman/sup/lists.

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

5. Update the /var/sysman/file.collections file to include information regarding this file collection. We add the following lines:

   ```
   # samplefc - a file collection to manage the files file1 file2 file3 and file4
   primary samplefc - /sample - / sample O power yes
   ```

   The first line is a comment. The fields in the second tells supper that this is a primary file collection, named samplefc, with no file system associated with it, which has files accessed through the /sample directory, with no specified file system size, having the scan process start at /sample, that all files in the directory should be evaluated, that this file collection runs on machines with the power architecture (RS/6000s) and that this file collection can be run on machines with different architecture.

   There are further details in Chapter 5, “Managing File Collections” in *IBM Parallel System Support Programs for AIX: Administration Guide*, SA22-7348.

6. Update the /var/sysman/sup/.resident file to include your new file collection. We add in the line:

   ```
   samplefc 0
   ```

   where 0 indicates that this file collection is served by the control workstation.

7. Build the scan file in the file collection’s own directory by running:

   ```
   supper scan samplefc
   ```

12.3.7 Installing a File Collection

There are two instances where it is necessary to install a file collection:

1. After you have created your own file collection
2. To set up boot/install servers to help distribute file collections to nodes

In both cases, it is a three-step process:

1. Run `supper update sup.admin` on the nodes and boot/install servers where you want to install the file collection. The sup.admin file collection contains

the files that identify and control all other file collections. You want the most up-to-date version.

2. Run `supper install <file collection name>` on each node or boot/install server that needs the file collection.
3. Add the `supper update <file collection name>` command to the crontab file to ensure that it is pulling down updates on a regular basis.

12.3.8 Removing a File Collection

A file collection cannot be globally removed; it can only be removed from the place where it is installed.

It is a two-step process:

1. Run `supper scan <file collection>` to create an updated scan file.
2. Run `supper remove <file collection>` to remove the file collection from the node.

If you want to remove the file collection from every node where it is installed, you have to run these steps on each node.

Attention

Do not remove any of the default file collections that are included when PSSP is installed. These are required by PSSP for its operation.

12.4 Managing the Automounter

The automounter is provided for an SP system administrator to manage the mounting of file systems across an SP system. The convenience of using the automounter is best illustrated with an example.

Consider an SP system with four nodes. You, as the system administrator, want to let all users have the capability to access all four nodes. You have chosen to use file collections to handle the management of the user database. What about the users' home directories? As with the user database, it is preferable to allow users to maintain one home directory across all four nodes.

To do this, one option is to store the home directories in an NFS file system on the control workstation and export it to the four nodes. Using NFS on its own, however, presents a problem. First, you have to decide whether each user's home directory is mounted as an individual file system, or whether the

Ex.1010 / Page 388 of 572

entire /HOME directory is to be mounted on all nodes. Second, NFS requires the addition of the home directory entries into the file /etc/filesystems and you then have the added duty of maintaining this file whenever changes are made. Finally, NFS-exported directories are not automatically unmounted when they are no longer used. If you have chosen to mount each user's directory as a separate file system you have to find a way to unmount this directory every time they log out of a node to minimize local system hangs due to NFS server outages.

The automounter can take care of all of this for you. When the mounting of a file system is under automounter control, the automounter transparently mounts the required file system. When there is no activity to that file system for a prescribed period of time, the file system is unmounted.

The automounter uses map files to determine which file systems it needs to control. There is typically one map file per file system that you want to manage. It is also possible to use NIS maps instead of automounter maps to specify which file systems the automounter is to handle. The usage of NIS maps is discussed in greater detail in "Managing the Automounter" in *IBM Parallel System Support Programs for AIX: Administration Guide*, SA22-7348. The rest of this chapter is devoted to the use of automounter maps to handle the mounting of file systems.

There are three versions of the automounter available for use within an SP system. At PSSP 2.2 and below, the Berkeley Software Distribution (BSD) version of the automounter, called AMD, is used. From PSSP 2.3 onwards, the automounter included with the AIX operating system is used.

There are two versions of the automounter available with AIX. At AIX 4.3.0 and below, it is simply known as the "automounter". At AIX 4.3.1 and above, it is called the "AutoFS automounter". The biggest difference between the two automounters is that in AutoFS, there is a kernel implementation that separates the `automount` command from the automountd daemon. This makes it possible to change map information without having to stop and re-start the automount daemon process. For the purposes of our discussion, we are going to refer to the two automounters as automounter and AutoFS.

12.4.1 The BSD Automounter

At PSSP 2.2 and below, AMD is used to handle the mounting of user home directories and other file systems. AMD is different from the automounter and AutoFS in its configuration and capabilities. For example, while it is possible for all three to select multiple network interfaces to use for mounting a file system, only AMD permits you to prioritize the interfaces to always try to use

Ex.1010 / Page 389 of 572

the switch interface to maximize performance. The automounter and AutoFS do not have this capability. This and other differences are detailed in *Inside the RS/6000 SP*, SG24-5145.

If you want to learn about the configuration of AMD, refer to *SP System Management: Easy, Lean and Mean*, GG24-2563.

12.4.2 The AIX Automounter

The automounter and the AutoFS automounter are kernel extensions which have an automountd daemon running in the background to service requests to mount file systems.

When a request is recognized, the automountd daemon uses NFS system calls to mount a directory onto the requesting machine. It then monitors the activities carried out against this file system. If it senses no activity against the file system after a specified amount of time (by default set to 5 minutes), automountd unmounts the file system from the requesting machine.

Both the automounter and AutoFS use map files residing on the server and clients to control file system mounts. There is a master map file, /etc/auto.master which contains entries for each file system to be controlled. Note that AutoFS by default looks for a map file named /etc/auto_master first. If it does not find an /etc/auto_master, then it looks for the /etc/auto.master. The use of the /etc/auto_master file is a requirement for AutoFS, and the /etc/auto.master is included in case users are migrating from the automounter and want to retain the old /etc/auto.master. Figure 157 on page 367 is an example of a /etc/auto.master file.

Ex.1010 / Page 390 of 572

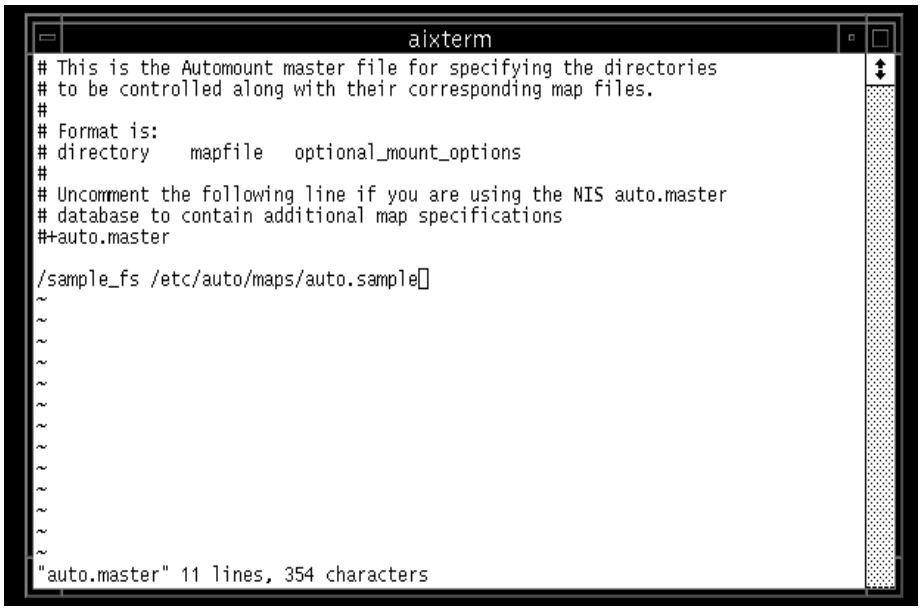


Figure 157. Sample /etc/auto.master File

This example tells the automountd that /sample_fs is to be served over the network according to the /etc/auto/maps/auto.sample map file. By default, the directory /etc/auto/maps is set up to store all the maps that automountd references.

The /etc/auto/maps/auto.sample file contains such entries:

```
dir_1 sp3en0:/share:&
dir_2 sp3en0:/tmp/:&
dir_3 sp3en0:/tmp/:&
```

The entries tell automountd that when /sample_fs/dir_1 is accessed on a client machine, mount /share from the server sp3en0. Similarly, an access to /sample_fs/dir_2 mounts /tmp/dir_2 from sp3en0 and /sample_fs/dir_3 mounts /tmp/dir_3 from sp3en0.

There is a difference in the mount points used by automounter and AutoFS on a client machine. In the preceding example, AutoFS simply mounts a file system from the server at the mount point specified. Therefore, /share is mounted at /sample_fs/dir_1, /tmp/dir_2 is mounted at /sample_fs/dir_2 and /tmp/dir_3 is mounted at /sample_fs/dir_3.

Ex.1010 / Page 391 of 572

However, the automounter mounts the directories at /tmp_mnt/<file system name>/<first sub-directory accessed>, then creates a symbolic link from the local directory to the mounted file system.

In the example, /share is mounted at /tmp_mnt/sample_fs/share and then a symbolic link is created from /sample_fs/dir_1 pointing to this directory.

The mount point /tmp/*, however, depends on which of dir_2 and dir_3 is first accessed. For example, if dir_2 is accessed first, then the mount point for /tmp/* is /tmp_mnt/sample_fs/dir_2/*.

Then /tmp/dir_2 is mounted as /tmp_mnt/sample_fs/dir_2/dir_2, with /sample_fs/dir_2 symbolically linked to this directory.

Then /tmp/dir_3 is mounted as /tmp_mnt/sample_fs/dir_2/dir_3, with /sample_fs/dir_3 symbolically linked to this directory.

If dir_3 is mounted first, then the mount point is /tmp_mnt/sample_fs/dir_3/*.

This can create problems for C-shell users because the C-shell `pwd` built-in command returns the actual path of a file directory. Since there is no guarantee that a file directory is always the same, C-shell commands cannot be based on the return value from `pwd`.

What if the client is the server? That is, we are trying to access a directory on the same machine on which it is residing. For example, we are trying to access /sample_fs/dir_1 on the machine sp3en0. In this case, a local mount of /share to the mount point of /sampe_fs/dir_1 is carried out.

In an SP, the automounter and AutoFS write error messages to an error log file, /var/adm/SPlogs/auto/auto.log. In addition, you can use the daemon facility of syslog subsystem to record errors. By default, all daemon.notice and greater messages are written by syslogd to /var/adm/SPlogs/SPdaemon.log.

For more information on the AIX and AutoFS automounter, refer to *AIX 4.3 System Management Guide: Communications and Networks*, SC23-4127 and "Managing the Automounter" in *IBM Parallel Systems Support Programs for AIX: Administration Guide*, SA22-7348.

12.4.3 Examples (SP Setup, Migration, Coexistence)

This section is based on the use of AutoFS. The setup is the same whether you are using AutoFS or the automounter because they can both reference

the same map files. Differences are noted when they exist, otherwise, you can apply the same steps to both the automounter and AutoFS.

12.4.3.1 SP Setup

In an SP, AutoFS can be used to manage the mounting of both user home directories and other file systems across the system. The entry in the SDR which determines whether AutoFS is used or not is held in amd_config. You can change the value in this entry either by the command `spsitenv` or the **Automounter Configuration** field in the Site Environment Information SMIT panel. Figure 158 shows this SMIT panel.

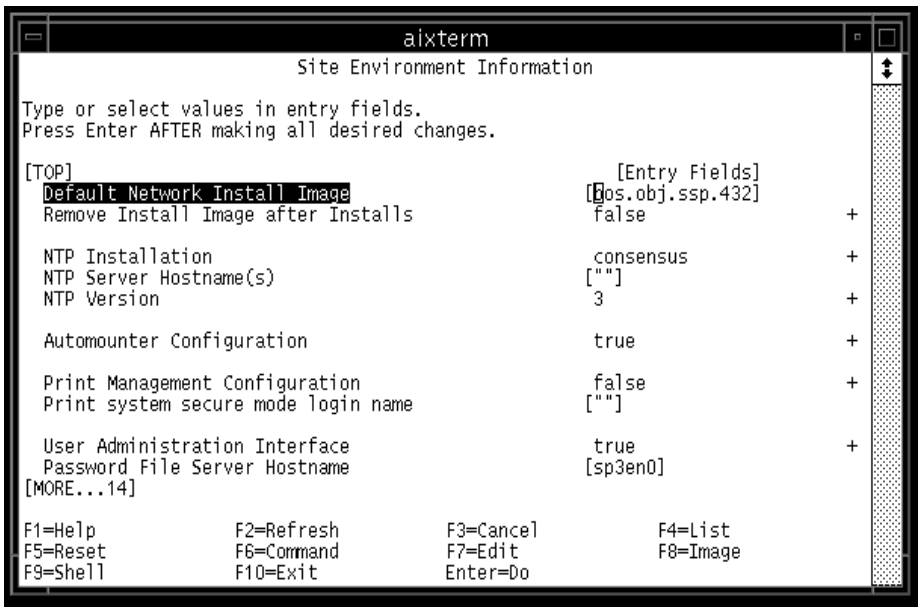


Figure 158. SMIT Panel for Turning On or Off the Automounter

When **Automounter Configuration** or amd_config is set to true, the system creates the necessary directories to store the configuration files for AutoFS and starts the automountd daemon. You will notice that the automountd daemon belongs to the autofs group. By default, AutoFS is set up to handle /u in an SP system, therefore, the /etc/auto/maps/auto.u file is installed and there is an entry for /u in the /etc/auto.master file.

If SPUM has also been turned on, the SP can then add, remove or change entries in the auto.u map file when SP users are added, removed or changed.

Figure 159 shows the default /etc/auto.master file when AutoFS Configuration (amd_config) is set to true.

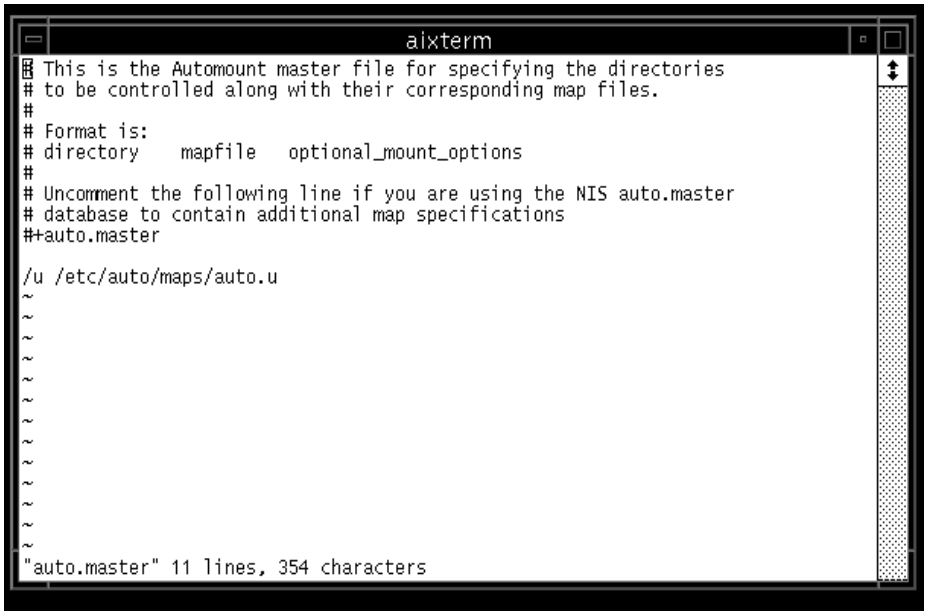


Figure 159. Default /etc/auto.master File

Figure 160 on page 371 is an example of a /etc/auto/maps/auto.u file after two SP users have been created on the system.

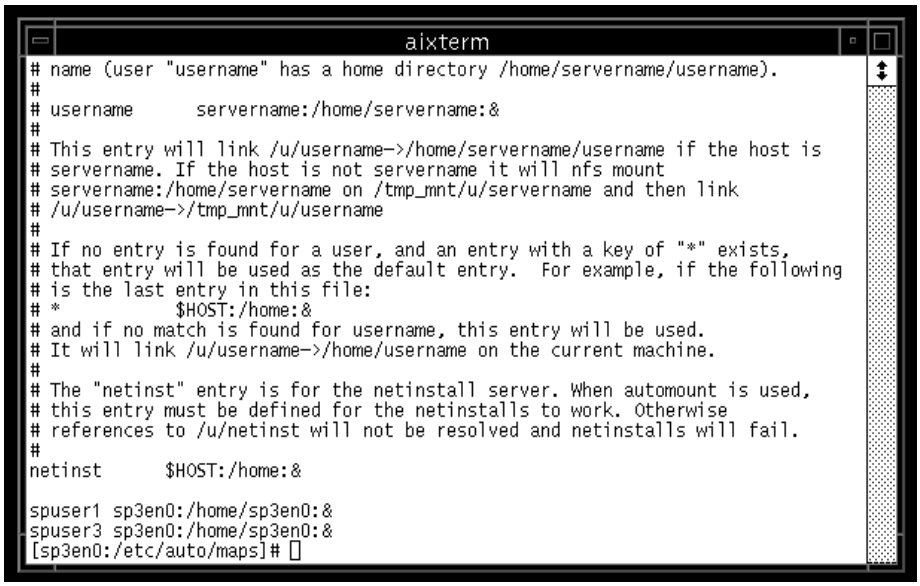


Figure 160. Sample /etc/auto/map/auto.u File

The control workstation (sp3en0) creates user directories under /home/sp3en0. For example, when spuser1 logs into a node, the user login facility searches for /u/spuser1, triggering the automountd daemon to mount /home/sp3en0/spuser1 over /u/spuser1.

Attention

If AutoFS is not enabled during the install process, and is subsequently enabled, then any user that has been created while AutoFS is disabled is not automatically added into the auto.u file. You have to manually add this into the file or use the `mkadment` command once the automounter is enabled.

12.4.3.2 Migration

If you are migrating from PSSP 2.2 and below to PSSP 2.3 and above, you have to convert the AMD map files into AutoFS map files. The two types of map files are not compatible. Since AutoFS and the automounter can share the same map files, there is no need for conversion above PSSP 2.3.

Assume that you are using AMD and have not customized any of the system created AMD configurations and map file at PSSP 2.2. During the migration, the system configuration process will detect that amd_config is set to true,

Ex.1010 / Page 395 of 572

create a new /etc/auto directory structure, and add the default AutoFS configuration files. If SPUM is also enabled, then the system will convert your existing AMD map file into the automounter map file (/etc/amd/amd-maps/amd.u to /etc/auto/maps/auto.u).

The command which actually converts the map file is `mkautomap`. It, however, can only be used to convert the amd.u file.

If you have modified the amd.u file, it may not be properly converted by `mkautomap`. You have to check the auto.u file the command builds. If the file does not properly convert, you have to manually build the appropriate entries in the auto.u file.

If you have created your own AMD map files, you have to rebuild the equivalent automounter map files.

If you have customized AMD in any way, you have to consider whether AutoFS (automounter) is going to allow the same customization. AutoFS customization is described in "Managing the Automounter" in *IBM Parallel Systems Support Program for AIX: Administration Guide*, SA22-7348.

12.4.3.3 Coexistence

It is possible to run both AMD and AutoFS within the same SP system. The SP maintains both the AMD and auto sub-directories under /etc.

In this instance, the control workstation runs both versions of the automounter and the nodes run either one of the automounters. That is, if a node is running PSSP 2.2, it is going to continue to run AMD. If a node is running PSSP 2.3 and above, it is going to run either automounter or AutoFS, depending on the AIX level.

If you are at PSSP 2.3 and above, and you have some nodes that are AIX 4.3.0 and below and some nodes that are AIX 4.3.1 and above, you need to run the compat_automountd. This automountd can handle both automounter and AutoFS requests and is specifically included in for this purpose.

If you have to run a mixed environment of PSSP 2.2 and above, with AIX a mixture of AIX levels (both below 4.3.0 and above 4.3.1), you will run both compat_automountd in place of automountd, and AMD.

If SPUM is enabled, any SP user added, deleted or have information changed causes updates made to both the AMD and automounter map files.

When the user database gets sent to the nodes for updates (for example, by using File Collections), the nodes may receive and update both copies of the

Ex.1010 / Page 396 of 572

maps. However, since only one of the automounters is running, only one of the maps is used for mounting the home directories.

Chapter 13. Backup and Recovery

Having gone through the installation process listed in Chapter 10, “Installation and Configuration” on page 257, we now look at how we can protect what we have painstakingly installed. Failures due to hard disks, operating system corruption, irrecoverable data errors and so forth can cause much distress to an organization whose business depends on the availability of the system. No administrator will disagree on the importance of having good backups. Not only do they prevent data loss, they also reduce recovery time in the event of system failures.

There are various aspects of backup that need to be considered, namely, the operating system, user data and configurations.

Users familiar with AIX tend to agree that backing up of the AIX operating system is made easy with the implementation of the Logical Volume Manager (LVM). The SP system runs on AIX, which is why the backup procedure on the SP system is not much different from that of a non-SP system.

No doubt when we deal with the SP system, we are dealing with a cluster of RS/6000 systems. The strategies involved in backup and recovery can prove to be of utmost importance to an administrator dealing with hundreds of nodes. Poor management in this area can drastically increase backup time and result in unnecessary waste of archival space. Always ask yourself when you need to back up and what to back up. A quick guide as to when to back up the operating system would be when major changes are made. This includes installation of new software, upgrades, patches, changes to configurations, hardware changes especially to system planar, hard disks belonging to rootvg, power supplies and so forth.

This chapter briefly describes the backup and restore procedure for the RS/6000 SP system. There are products on the market that assist the backup of data. Examples include ADSM and Sysback from IBM, and Networker from Legato.

13.1 Backup/Recovery of rootvg

On an RS/6000 system, we have to ensure that proper backup is done for the volume group rootvg. On a failed system, the very first thing we need to recover is the rootvg volume from a reliable backup.

First we give a brief description of `mksysb` before we look into how we can back up the CWS using this command.

© Copyright IBM Corp. 1999

The mksysb command creates a bootable image by first putting a boot sector at the beginning of the media. This type of bootable image is usually done on tape media. If the mksysb image were created in a file, the file would not contain the boot image and would therefore not be bootable on its own.

Bootable images contain four parts, as shown in Figure 161.

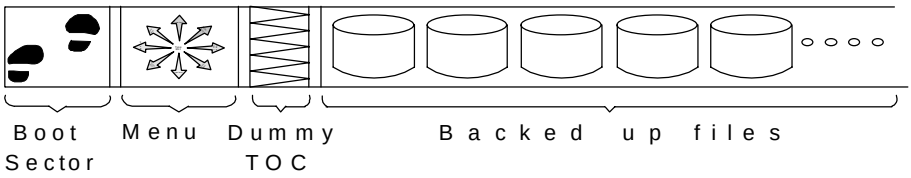


Figure 161. Contents of an mksysb Backup Image

The first part is the boot sector from which the node does a bootstrap boot, building the basic kernel in memory.

The second part is a menu that is displayed on the console of the node that is booting. This menu can be overridden with commands at the time the bootable image is created, or at the time of restoration by a certain file with a specific internal format.

The third part is a dummy table of contents, which is not used.

The last part contains the actual data files. This is the largest part. One of the first files that is in this data section is a file that contains the size, location and details about the various logical volumes and file systems that were mounted on the root volume group at the time of the mksysb. This file is looked at by the booting process to reconstruct the logical volume manager part of the system to what it was before so that the rest of the files on the tape can be restored into the same file systems that they were in.

A few points you have to consider about the mksysb command:

- It only backs up mounted file systems in the root volume group.
- If there is a database in a raw logical volume on the root volume group, it will not get backed up by this command.
- If there are other volume groups on a node, mksysb will not pick them up, and will not back them up, but the savevg command is designed specifically for non-root volume groups.

On a traditional RS/6000, the mksysb image is used for backup of systems, not cloning of systems - the mksysb image created on one type of machine will generally not be restorable on another type of machine. Note, however, that this is not true in the RS/6000 SP system: a mksysb image made on one node *will* restore on any other node.

13.1.1 Backup of the Control Workstation

The heart of the SP is the Control Workstation (CWS). It is through the CWS that node installation is made possible. This is because nodes, although they are basically RS/6000 machines put together in a frame, do not have consoles attached. The CWS provides these nodes with virtual consoles among other controls. Understanding the importance of the CWS, we now need to look at how we can take backups of this system. Although we talked about the CWS with so much hype, it is nothing more than a normal RS/6000 machine with additional software installed to function as a control master for the SP. We use the same `mksysb` and `savevg` commands to back it up. Care has to be taken, though, to back up the /spdata as well as the /tftpboot directories, as they could be created as file systems on a different volume group.

To back up the CWS, you can either use the command line or the SMIT screen. You have to be a root user to perform mksysb backups.

On the command line, use the following:

```
/usr/bin/mksysb -i -X /dev/rmt0
```

On the SMIT screen, use the following:

```
# smitty mksysb
```

See Figure 162 on page 378:

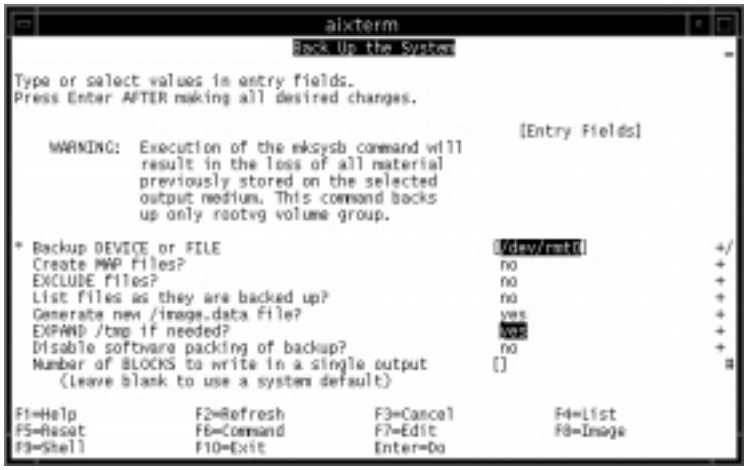


Figure 162. SMIT mksysb

The restore procedure of the CWS is dependent on the type of system you have as a CWS. Refer to your CWS system's installation guide about how to reinstall your system.

13.1.2 Backup/Recovery of Nodes

We now look at how we back up nodes. If we have gone through the installation of a new node, we understand that the image of the backup comes from a file and not from a bootable tape. This is made possible by the use of Network Install Manager (NIM). The PSSP package comes with a standard minimum mksysb image. The way this minimum image was made is similar to the method we will use to back up our nodes.

A mksysb to a file is very much the same as doing it onto tape media. Here, however, instead of specifying a tape drive, we specify a filename. The file can either reside on an NFS-mounted file system from the CWS or a local file system. For an NFS-mounted file system, the moment the mksysb is done, the file can be backed up to tape on the CWS. For a mksysb that is created on a local file system, you can either `ftp` or `rcp` it over to the CWS or any host that has a locally attached tape drive for back up.

Following is a simple example to illustrate the creation of a mksysb image of a node to a file system on the CWS. It is always best to ensure that the system is not in use before doing a backup.

On the CWS, export the file system for read and write access to the node:

```
[sp5en0:/] # /usr/sbin/mknfsexp -d '/spdata/sys1/install/images' -t 'rw'
-r 'sp5n09' '-B'
```

On the node, mount the NFS-exported file system from the CWS:

```
[sp5n09:/] # /usr/sbin/mount sp5en0:/spdata/sys1/install/images /mnt
[sp5n09:/] # df
Filesystem    512-blocks      Free %Used    Iused %Iused Mounted on
/dev/hd4           16384      8224   50%      964    24% /
/dev/hd2          598016    141472   77%     9295    13% /usr
/dev/hd9var        65536     59920    9%      355     5% /var
/dev/hd3           65536     63304    4%       33     1% /tmp
/dev/hd1            8192      7840    5%       18     2% /home
sp5en0:/spdata/sys1/install/images   3907584   2687680   32%    8009
2% /mnt
```

Next, kick off the `mksysb` on the node.

```
[sp5n09:/] # /usr/bin/mksysb -i -X /mnt/mksysb.image.node9
```

The recovery of nodes is similar to the steps listed in the installation section and are not further elaborated on here. Refer to 10.3, “Frame, Node And Switch Installation” on page 267 for details on how to reinstall a node.

13.2 Backup/Recovery of System Database Repository (SDR)

We have been looking at backing up the system at an overall level. The CWS functions as a control for the whole SP complex. The System Database Repository (SDR) stores the configuration information about the whole system. Any changes made to the SDR need to be backed up. It is always advisable that, before any changes to the system configuration are made (like adding new nodes, frames, or changing hostnames, IP addresses), the SDR is backed up. If any errors were made, the SDR can easily be recovered without having to go through the whole mksysb restore procedure, which is time-consuming.

The command to back up the SDR is `SDRArchive`. It takes a snapshot of the SDR contents and saves it in a tar file located in the /spdata/sys1/sdr/archives directory on the CWS. The format of the backup is backup.yyddd.hhmm, where:

- yy denotes the last two digits of the year (for example, 99 for the year 1999).
- dd denotes the Julian date (039 is the 39th day of the year).
- hh denotes the hour in 24-hour format.

- mm denotes the minute.

It is quite normal for administrators to do many SDR archives and end up not knowing which archived file is usable. The `SDRArchive` command provides an append_string where you can add meaningful words to tell you more about the archived file. For example, if migrating PSSP to a higher version, you would want to archive your system's SDR with the following format:

```
# SDRArchive Before_Migrate
```

This will give you the following SDR archive file:
/spdata/sys1/sdr/archives/backup.99039.1001.Before_Migrate

If anything goes wrong with the SDR, we can restore the previous SDR information with the `SDRRestore` command:

```
# SDRRestore backup.99039.1001.Before_Migrate
```

The SDR should be backed up before making changes to it. Also, a few copies of SDR done on separate dates should be kept in case of corruption to any one of them. Although SDRArchive/SDRRestore is a handy tool to back up and restore the SDR, it should not be a replacement for a full system backup. There are misconceptions that the SDR is the heart of the SP and restoring it is all that is needed to restore a fully functional CWS. This is definitely an erroneous (and dangerous) idea. There are many components that make up the SP; the SDR is just one part of it.

13.3 Backup/Recovery of Kerberos Database

This section touches on the backup and restore of the Kerberos V4 database. For information on Distributed Computing Environment (Kerberos Version 5), refer to its related manuals.

On the Kerberos Authentication Server (KAS), which by default is the CWS, the Kerberos database is stored in the /var/kerberos/database directory. Just like the SDR mentioned earlier, backup of this database should be done prior to making changes to it. However, unlike the SDR where there are commands to facilitate the backup and restore, backing up Kerberos is done manually. The files that you would want to back up are as follows:

On KAS:

- /var/kerberos/database/*
- /etc/krb-srvtab

Ex.1010 / Page 404 of 572

- /etc/krb.conf
- /etc/krb.realms
- /.k
- $HOME/.klogin
- $KRBTKFILE or /tmp/tkt<uid>

On the nodes:

- /etc/krb-srvtab
- /etc/krb.conf
- /etc/krb.realms
- $HOME/.klogin
- $KRBTKFILE or /tmp/tkt<uid>

For information relating to each file, refer to 7.4, “Managing Kerberos on the SP” on page 156.

To restore the Kerberos database, you will have to restore all the files you backed up.

There are instances where even restoring these files cannot help to recover the Kerberos database. In cases like those, you will need to rebuild the Kerberos database.

13.4 Backup/Recovery of User Specified Volume Group

Apart from the operating system, which by now we have an idea as to how to backup and restore, we next look at the data portion of the system. As a general rule, we normally advise administrators to separate their system files from their data files. The rootvg size should be kept to a minimum. Separate volume groups should be created to accommodate customers’ data. This helps in the sense that if anything goes wrong with the rootvg, only this volume group needs to be restored; the data residing on the other volume groups is kept intact.

Commands provided by AIX for backing up files are `tar`, `cpio`, `backup`, `rdump`. These can be used for journaled file systems (JFS). For databases, like Oracle, SAP, Sybase and so on, you will need to refer to the relevant database backup procedure; using conventional AIX backup commands may render your backups useless.

While mksysb is used for backing up the rootvg, savevg is used for backing other volume groups. Like mksysb, only mounted journaled file systems are backed up with the savevg command.

The restoration of a savevg backup is accomplished with the restvg command. For information on usage of these commands, refer to AIX command documentation.

13.5 Proper Documentation

As with any setup, proper documentation is very important. We may have a very good backup procedure in place and good knowledge about the setup, but what is there to prevent us from forgetting about what we have done? What happens if the administrator leaves the company?

Any setup should be accompanied by documentation containing the procedure to setup, the structure of the setup, the problems faced during the setup, and so on. This documentation should be kept updated when there are changes to the system. The more you put into the documentation, the better position you are in to efficiently bring back a failed system.

Ex.1010 / Page 406 of 572

Part 4. Application Enablers

© Copyright IBM Corp. 1999

Chapter 14. Data Storage

SP nodes use AIX's Logical Volume Manager (LVM) as a building block for data storage. In turn, LVM uses a three-layered logical approach of volume group (VG), logical volume (LV), and file system to structure the data on the hard disks. For further details on LVM and how it works, refer to *AIX Storage Management*, GG24-4484.

Each SP node is capable of operating as a unique machine. This is because each node has its own operating system and stores its own data. Since each node uses LVM to store its data, however, the stored data is only accessible to the users and processes on the node. If users and processes on other nodes have to access this data, then file or disk sharing software has to be installed above LVM.

There are two strategies for sharing files. The first is to locate all shared files in file systems on one node, then use products like Network File Systems (NFS), Distributed File System (DFS) and Andrew File System (AFS). The second is to create a distributed environment, where data stored on any node is accessible by other nodes, using IBM's Virtual Shared Disks (VSD), Hashed Shared Disks (HSD), Recoverable Virtual Shared Disks (RVSD) or General Parallel File System (GPFS).

This chapter begins with a look at data stored locally on nodes, then examines the tools previously mentioned for sharing data among nodes, starting with NFS, DFS and AFS, then moving on to VSD, HSD and RVSD, and concluding with GPFS.

14.1 Local Disks

Each SP node stores data on disks physically attached to it. This data includes the AIX operating system, client portions of the PSSP software, application binaries and application data. This has both advantages and disadvantages.

Storing data on nodes permits an SP system to be used for applications like server consolidation. Each node has its own AIX operating system and can operate independently from other nodes to serve different applications. All nodes are centrally managed through one machine, the control workstation. Node locality is also scalable — as nodes are added, disks can be added linearly.

© Copyright IBM Corp. 1999

However, administrating these nodes can be a tedious task. There are multiple copies of the operating system and application software to install, backup and keep current. Each node can have unique configurations and tuned parameters that need to be maintained. If there are a large number of nodes, such tasks can be very time consuming.

Recall that an SP node uses LVM to handle the storage of data onto hard disks. LVM has many features which make it suitable for such a task. However, it does have one drawback: stored data is accessible to only the users and processes on the node on which the data is stored. In an SP system, this is a major inconvenience.

Consider one example: backups. The control workstation needs to make a backup copy of the data on one node, and then store it on a hard disk back on the control workstation itself. Data on the node is stored using LVM and can only be accessed by the local users and processes. If the control workstation wants to access this data, it uses a two-step process:

1. Use an application like telnet to log into the node and execute a backup.

2. Then use an application like ftp to download the backup image for storage on the control workstation.

If it is possible for the node and the control workstation to share data with each other, this process can be reduced to one step. The node “sees” the area where the control workstation stores backup images. All the control workstation has to do is initiate the backup on the node, and the node can store its backup image into this area.

Here is a second example. Consider a parallel application that runs on two nodes. If a user on one node wants to change a piece of data on the other node, LVM does not permit it. The user has to go through steps similar to the previous example to either obtain the data and process it locally, then move it back to the other node, or go to the other node and process it there. Either way, it is an inconvenience.

We can therefore conclude that LVM is insufficient as the sole means for handling data storage in an SP system.

In summary, the design that allows the storage of data on SP nodes has both advantages and disadvantages. Our concerns in this chapter center on the use of LVM within this design. We now look at the file sharing software which overcomes LVM’s limitation.

Ex.1010 / Page 410 of 572

14.2 Global File Systems

This section gives an overview of the most common *global* file systems. A global file system is a file system which resides locally on one machine (the file server), and is made globally accessible to many clients over the network. All file systems described in this section use UDP/IP as the network protocol for client/server communication (NFS Version 3 may also use TCP).

One important motivation to use global file systems is to give users the impression of a single system image by providing their home directories on all the machines they can access. Another is to share common application software which then needs to be installed and maintained in only one place. Global file systems can also be used to provide a large scratch file system to many machines, which normally utilizes available disk capacity better than distributing the same disks to the client machines and using them for local scratch space. However, the latter normally provides better performance, so a trade-off has to be made between speed and resource utilization.

Apart from the network bandwidth, an inherent performance limitation of global file systems is the fact that one file system resides completely on one machine. Different file systems may be served by different servers, but access to a single file, for example, will always be limited by the I/O capabilities of a single machine and its disk subsystems. This might be an issue for parallel applications where many processes/clients access the same data. To overcome this limitation, a *parallel* file system has to be used. IBM's parallel file system for the SP is described in 14.4, “General Parallel File System (GPFS)” on page 409.

14.2.1 Network File System (NFS)

Sun Microsystem's Network File System (NFS) is a widely used global file system, which is available as part of the base AIX operating system. It is described in detail in Chapter 10, "Network File System" of *AIX Version 4.3 System Management Guide: Communications and Networks*, SC23-4127.

In NFS, file systems residing on the NFS server are made available through an *export* operation, either automatically when the NFS startup scripts process the entries in the /etc/exports file, or explicitly by invoking the `exportfs` command. They can be mounted by the NFS clients in three different ways. A *predefined* mount is specified by stanzas in the /etc/filesystems file, an *explicit* mount can be performed by manually invoking the `mount` command, and *automatic* mounts are controlled by the `automount` command, which mounts and unmounts file systems based on their access frequency. This relationship is sketched in Figure 163 on page 388.

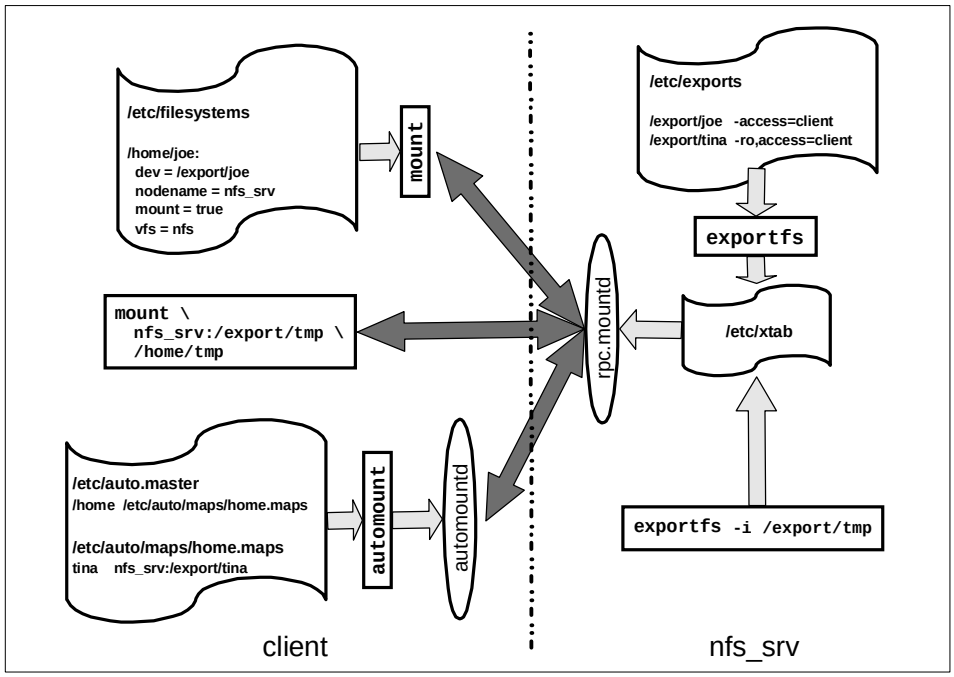


Figure 163. Conceptual Overview of NFS Mounting Process

The PSSP software uses NFS for network installation of the SP nodes. The control workstation and boot/install servers act as NFS servers to make resources for network installation available to the nodes, which perform explicit mounts during installation. The SP accounting system also uses explicit NFS mounts to consolidate accounting information.

NFS is often used operationally to provide global file system services to users and applications. Among the reasons for using NFS are that it is part of base AIX, it is well known in the UNIX community, it is very flexible, and it is relatively easy to configure and administer in small to medium-sized environments. However, NFS also has a number of shortcomings. We summarize them here to provide a basis to compare NFS to other global file systems.

Performance: NFS Version 3 contains several improvements over NFS Version 2. The most important change probably is that NFS Version 3 no longer limits the buffer size to 8 kB, improving its performance over high bandwidth networks. Other optimizations include the handling of file attributes and directory lookups, and increased write throughput by

collecting multiple write requests and writing the collective data to the server in larger requests.

Security: Access control to NFS files and directories is by UNIX mode bits, that means by UID. Any root user on a machine which can mount an NFS file system can create accounts with arbitrary UIDs and so can access all NFS-mounted files. File systems may be exported read-only if none of the authorized users needs to change their contents (like directories containing application binaries), but home directories will always be exported with write permissions as users must be able to change their files. An option for secure NFS exists, but is not widely used. Proprietary access control lists (ACLs) should not be used since not all NFS clients understand them.

Management: A file system served by an NFS server cannot be moved to another server without disrupting service. Even then, clients mount it from a specific IP name/address and will not find the new NFS server. On all clients, references to that NFS server have to be updated. To keep some flexibility, alias names for the NFS server should be used in the client configuration. These aliases can then be switched to another NFS server machine should this be necessary.

Namespace: With NFS, the client decides at which local mount point a remote filesystem is mounted. This means that there are no global, universal names for NFS files or directories since each client can mount them to different mount points.

Consistency: Concurrent access to data in NFS is problematic. NFS does not provide POSIX single site semantics, and modifications made by one NFS client will not be propagated quickly to all other clients. NFS does support byte range advisory locking, but not many applications honor such locks.

Given these shortcomings, we do not recommend the use of NFS in large production environments that require fast, secure, and easy to manage global file systems. On the other hand, NFS administration is fairly easy, and small environments with low security requirements will probably choose NFS as their global file system.

14.2.2 The DFS and AFS File Systems

There are mainly two global file systems which can be used as an alternative to NFS. The Distributed File System (DFS) is part of the Distributed Computing Environment (DCE) from the Open Software Foundation (OSF), now the Open Group. The Andrew File System (AFS) from Transarc is the base technology from which DFS was developed, so DFS and AFS are in many aspects very similar. Neither DFS nor AFS are part of base AIX; they are available as separate products. Availability of DFS and AFS for platforms other than AIX differs, but not significantly.

For reasons that are discussed later, we recommend using DFS rather than AFS except when an SP is to be integrated into an existing AFS cell. We therefore limit the following high-level description to DFS. Most of these general features also apply for AFS, which has a very similar functionality. After a general description of DFS, we point out some of the differences between DFS and AFS that justify our preference of DFS.

14.2.2.1 What is the Distributed File System

The DFS is a distributed application that manages file system data. It is an application of the Distributed Computing Environment (DCE) in the sense that it uses almost all of the DCE services to provide a secure, highly available, scalable, and manageable distributed file system.

DFS data is organized in three levels:

- Files and directories. These are the same data structures known from local file systems like the AIX Journaled File System (JFS). DFS provides a global namespace to access DFS files.
- Filesets. A DFS *fileset* is a group of files and directories which are administered as a unit, for example, all the directories that belong to a particular project. User home directories may be stored in separate filesets for each user, or may be combined into one fileset for a whole (AIX) group. Note that a fileset cannot be larger than an aggregate.
- Aggregates. An *aggregate* is the unit of disk storage. It is also the level at which DFS data is exported. There can be one or more filesets in a DFS aggregate. Aggregates cannot be larger than the logical volume in which they are contained.

The client component of DFS is the *cache manager*. It uses a local disk cache or memory cache to provide fast access to frequently used file and directory data. To locate the server that holds a particular fileset, DFS uses the *fileset location database (FLDB) server*. The FLDB server transparently accesses

Ex.1010 / Page 414 of 572

information about a fileset's location in the FLDB, which is updated if a fileset is created or moved to another location.

The primary server component is the *file exporter*. The file exporter receives data requests as DCE Remote Procedure Calls (RPCs) from the cache manager, and processes them by accessing the local file systems in which the data is stored. DFS includes its own Local File System (LFS), but can also export other UNIX file systems (although with reduced functionality). It includes a *token manager* to synchronize concurrent access. If a DFS client wants to perform an operation on a DFS file or directory, it has to acquire a token from the server. The server revokes existing tokens from other clients to avoid conflicting operations. In this way, DFS is able to provide POSIX single-site semantics.

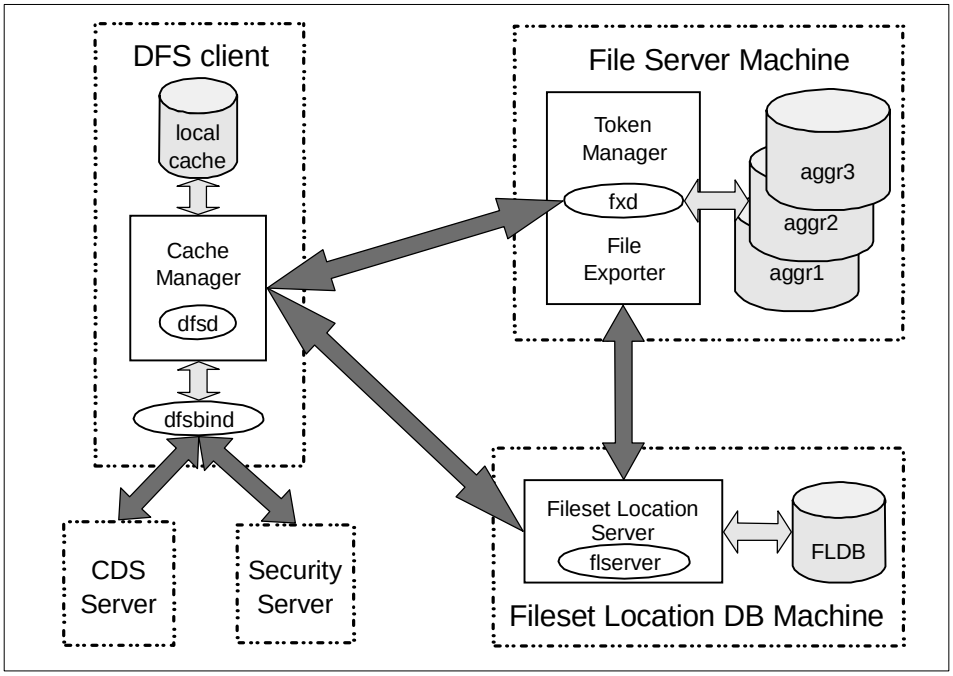


Figure 164. Basic DFS Components

Figure 164 shows these DFS components. This is an incomplete picture; in reality there are many more DFS components (such as the replication server) and various management services like the fileset server and the update server. More detailed information about DFS can be found in *Distributed Computing Environment for AIX, Version 2.2: Quick Beginnings,* SC23-4188.

and *IBM DCE for AIX: DFS Administration Guide and Reference*, provided with the product in soft copy only.

The following list summarizes some key features of DCE/DFS, and can be used to compare DFS with the discussion in 14.2.1, “Network File System (NFS)” on page 387.

Performance: DFS achieves high performance through client caching. The client to server ratio is better than with NFS, although exact numbers depend on the actual applications. Like NFS, DFS is limited by the performance of a single server in the write case. However, replication can help scale read-only access.

Security: DFS is integrated with the DCE Security Service, which is based on Kerberos Version 5. All internal communication uses the authenticated DCE RPC, and all users and services which want to use DFS services have to be authenticated by logging in to the DCE cell (except when access rights are explicitly granted for unauthenticated users). Access control is by DCE principal; root users on DFS client machines cannot impersonate these DCE principals. In addition, DCE Access Control Lists can be used to provide fine-grained control; they are recognized even in a heterogeneous environment.

Management: Since fileset location is completely transparent to the client, DFS filesets can be easily moved between DFS servers. Using DCE’s LFS as the physical file system, this can even be done without disrupting operation. This is an invaluable management feature for rapidly growing or otherwise changing environments. Because there is no local information on fileset locations on the client, administering a large number of machines is much easier than maintaining configuration information on all of these clients.

Namespace: DFS provides a global, worldwide namespace. The file system in a given DCE cell can be accessed by the absolute path /.../cell_name/fs/, which can be abbreviated as /: (slash colon) within that cell. Access to foreign cells always requires the full cell name of that cell. The global name space ensures that a file will be accessible by the same name on every DFS client. The DFS client has no control over mount points: filesets are mounted into the DFS namespace by the servers. Of course, a client may

use symbolic links to provide alternative paths to a DFS file, but the DFS path to the data will always be available.

Consistency: Through the use of a token manager, DFS is able to implement complete POSIX single site read/write semantics. If a DFS file is changed, all clients will see the modified data on their next access to that file. Like NFS, DFS supports byte range advisory locking.

Operation: To improve availability, DFS filesets can be replicated, that is, read-only copies can be made available by several DFS servers. The DFS server processes are monitored and maintained by the DCE *basic overseer server* (BOS), which automatically restarts them as needed.

In summary, many of the problems related to NFS either do not exist in DFS, or have a much weaker impact. DFS is therefore more suitable for use in a large production environment. On the other hand, DCE administration is not easy and requires a lot of training. The necessary DCE and DFS licenses also add extra cost.

14.2.2.2 Differences Between DFS and AFS

Apart from the availability (and licensing costs) of the products on specific platforms, there are two main differences between DFS and AFS: the integration with other services, and the mechanism to synchronize concurrent file access. The following list summarizes these differences:

Authentication AFS uses Kerberos Version 4, in an implementation that predates the final MIT Kerberos 4 specifications. DCE/DFS uses Kerberos Version 5. For both, the availability of other operating system services (like telnet or X display managers) that are integrated with the respective Kerberos authentication system depends on the particular platform.

Authorization DFS and AFS ACLs differ, and are more limited in AFS. For example, AFS can only set ACLs on the directory level, not on file level. AFS also cannot grant rights to a user from a foreign AFS cell, whereas DFS supports ACLs for foreign users.

Directory Service DCE has the Cell Directory Service (CDS), through which a client can find the server(s) for a particular service. The DFS client uses the CDS to find the Fileset Location Database. There is no fileset location information on the client. AFS has no directory service. It relies on a local

configuration file (/usr/vice/etc/CellServDB) to find the Volume Location Database (VLDB), the Kerberos servers, and other services.

RPC Both DFS and AFS use Remote Procedure Calls (RPCs) to communicate over the network. AFS uses Rx from Carnegie Mellon University. DFS uses the DCE RPC, which is completely integrated into DCE, including security. AFS cannot use dynamic port allocation, DFS does so by using the RPC *endpoint map*.

Time Service DFS uses the DCE Distributed Time Service, discussed in Chapter 6, “Time Service” on page 131. AFS clients use their cache manager and NTP to synchronize with the AFS servers.

Synchronization Both DFS and AFS use a token manager to coordinate concurrent access to the file system. However, AFS revokes tokens from other clients when *closing* a file, whereas DFS revokes the token when *opening* the file. This means that DFS semantics are completely conforming with local file system semantics, whereas AFS semantics are not. Nevertheless, AFS synchronization is better than in NFS, which does not use tokens at all.

It is obvious that DFS is well integrated with the other DCE core services, whereas AFS requires more configuration and administration work. DFS also provides file system semantics that are superior to those of AFS. So unless an existing AFS cell is expanded, we recommend that you use DFS rather than AFS to provide global file services.

14.3 Shared Disks

A serial application can exceed the I/O capacity of a single node, even after the application’s data has been spread across many disks and I/O adapters of the node. Parallel applications need access to data that is spread across many nodes to improve I/O performance.

If we can provide a node with the capability to access data residing on any other node in the SP system, we satisfy the access need of parallel applications and also offer a possibility for improved performance of serial applications.

IBM makes this possible with its family of offerings based on the Virtual Shared Disks (VSD) technology.

Ex.1010 / Page 418 of 572

14.3.1 Virtual Shared Disks (VSD)

VSD allows data stored in raw Logical Volumes (LVs) on one node to be globally accessible over an IP network. A VSD may be viewed as an LV that other nodes may access as though it has been locally configured. There may therefore be multiple VSDs defined and configured on any given node.

This technology is designed with applications like Oracle's parallel database in mind. Oracle's database architecture is highly centralized. Any processing node must be able to "see" the entire database. Therefore, in Oracle's parallel database, all nodes must have access to all disks, regardless of where these disks are physically attached.

A node which has VSDs defined and configured is called *a server node*. A node which accesses other nodes' VSDs is called *a client node*. A node may be both a client node and a server node.

VSD is designed to run over any IP network. We do, however, strongly recommend that the switch network be used for superior performance.

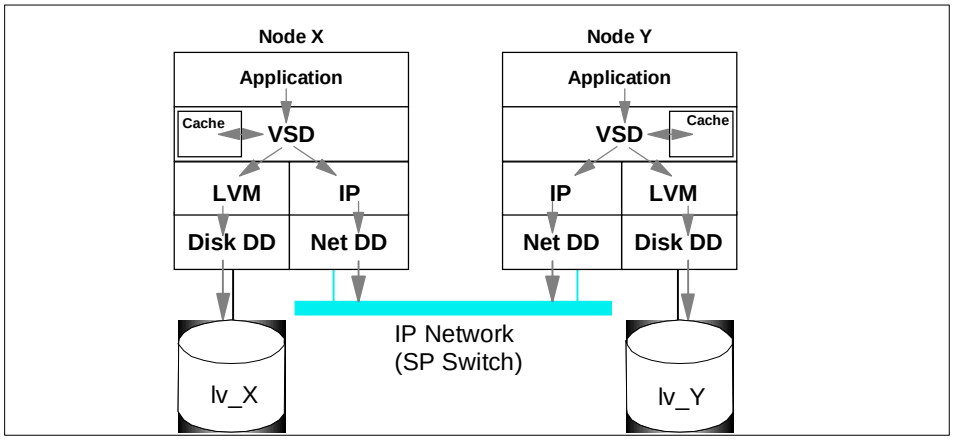


Figure 165. VSD Architecture

As shown in Figure 165, VSD provides a device driver loaded as a kernel extension which sits below the application layer, and above the LVM and IP device drivers. When the application requests a piece of data, this request is sent to the VSD device driver, which then looks for the data in the appropriate place.

For example, the application on Node X is looking for a piece of data and passes this request to the VSD device driver. Node X's VSD driver can then look for the data in one of three places:

1. VSD cache

   The VSD cache stores all I/O requests in case that piece of data is required later. This area is shared by all VSDs defined and configured on one node. The data here is stored in 4 KB blocks, a size optimized for Oracle's parallel database program. Note that this is an optional feature. If your I/O requests involve operations much larger than 4 KB, using this cache may prove to be counterproductive because of the additional costs of fragmenting it into the 4 KB blocks.

2. lv_X

   The VSD device driver passes the request to Node X's LVM and disk device drivers to fetch the data.

3. Node Y

   The VSD device driver passes the request through the IP and network device drivers to Node Y. Node Y's IP and network device drivers then pass the request to Node Y's VSD device driver, which finds the data, either in Node Y's VSD cache (if applicable) or in lv_Y. The data is fetched, and then passed back to Node X in a similar fashion, making use of the IP and network device drivers.

VSD device drivers use their own stripped-down IP protocol for performance reasons. In addition, VSD uses unique buffers, buddy buffers and pbufs to handle the network transmissions. Detailed information on these buffers, including how to tune them, can be found in *IBM Parallel System Support Programs For AIX: Managing Shared Disks*, SA22-7349.

Because VSDs are LVs, they do not have a file locking mechanism to preserve data integrity. This task falls upon the application that is using the VSDs.

VSDs can be defined, configured and managed by the command line, or by SMIT panels, or by the graphical user interface Perspectives. Their definitions are stored in the SDR on the control workstation.

The VSD software is an optional feature of PSSP is installed on the nodes and the control workstation. The VSD software consists of the filesets shown in Table 19 at PSSP 3.1:

Table 19. VSD Fileset Names and Descriptions

Fileset	Description
vsd.cmi	VSD SMIT panels
vsd.vsdd	VSD device driver

Ex.1010 / Page 420 of 572

Fileset	Description
vsd.sysctl	VSD sysctl commands

Additional information on VSD filesets can be found in *PSSP 3.1 Announcement*, SG24-5332.

To build VSDs in a SP system, perform the five general steps as follows:

1. Install the filesets on the nodes you intend to run VSD.
2. Designate the VSD nodes.

   Every node that is going to be involved with VSDs, whether as a client or as a server, has to be designated as a VSD node in the SDR on the control workstation. In Perspectives, this is done by choosing the action **Designate as an IBM VSD Node**.

   - For SMIT panels, run `smit vsd_data` and select the **VSD Node Information** option.
   - From the command line, use `/usr/lpp/csd/bin/vsdnode`.

   It is more advantageous to use the Perspectives interface or SMIT panels because both automatically bring up a list of tuning parameters that you can set. If the command line option is used, the settings have to be included when `vsdnode` is run.

   Figure 166 on page 398 is an example of the Perspective panel used for designating VSD nodes (in this case, nodes 1 and 15).

Ex.1010 / Page 421 of 572

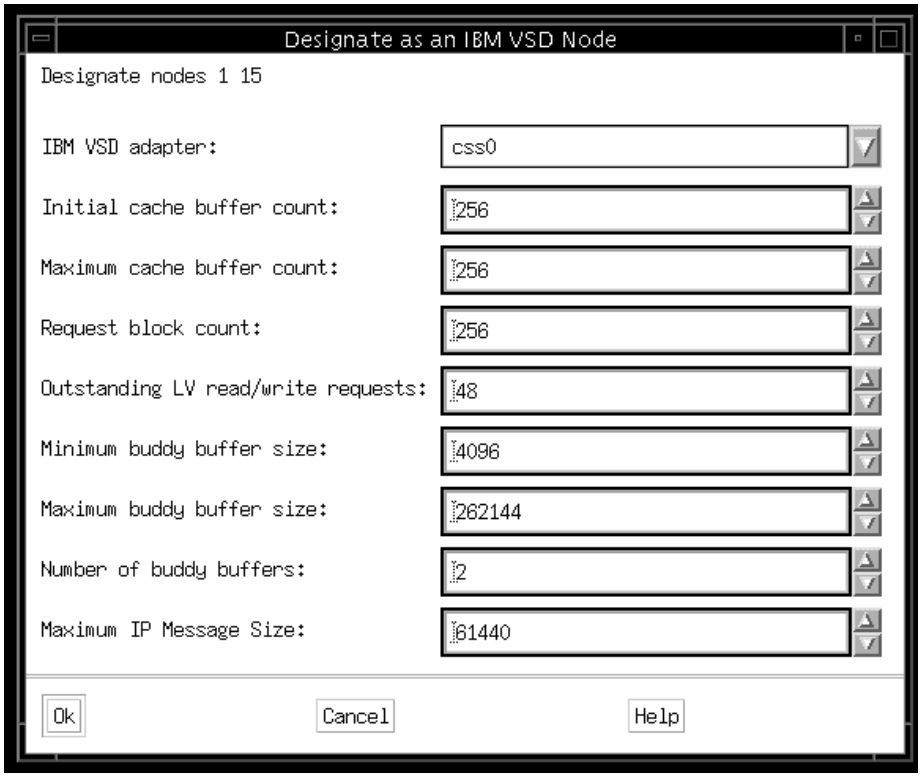


Figure 166. Perspectives Panel for Designating VSD Nodes

3. Create or define the VSDs for each node that is to be a VSD server.

   There are two options to set up VSDs on the server nodes: create them or define them.

   Creating VSDs means that there are no VGs and LVs established on the server node to be used for VSDs. In this case, one can again go through the Perspectives interface or the command line. In Perspectives, it is necessary to first add a pane that shows the VSDs on the server node (even if none has yet been created). Once the pane is added, use the action **Create** to create both the global VG and the LVs to be used as VSDs. For SMIT, run `smit vsd_data` and then select the **Create a Virtual Shared Disk** option. From the command line, run `createvsd`. Once again, it we recommend that you use the Perspectives interface or the SMIT panels because they automatically bring up all the settings and options available.

Figure 167 is an example of the Perspectives panel used for creating VSDs.

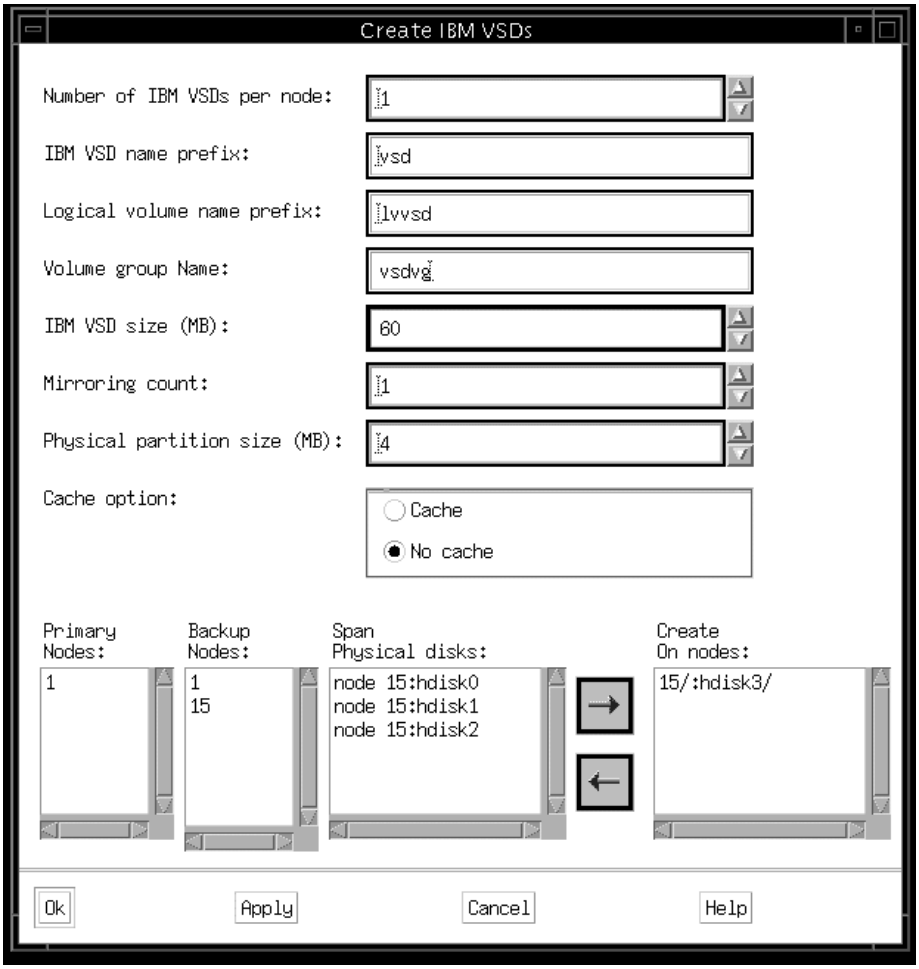


Figure 167. Perspectives Panel for Creating VSDs

Defining VSDs means that VGs and LVs have already been established locally on the node and you want the SP to use them for VSDs, that is, you want to define them in the SDR on the control workstation. In Perspectives, from the VSD pane, use the action **Define** to define a LV as a VSD within a global VG. For SMIT panels, run `smit vsd_data` and then select the **Define a Virtual Shared Disk** option. The command line equivalent is `defvsd`.

Figure 168 is an example of the Perspectives panel for defining VSDs.

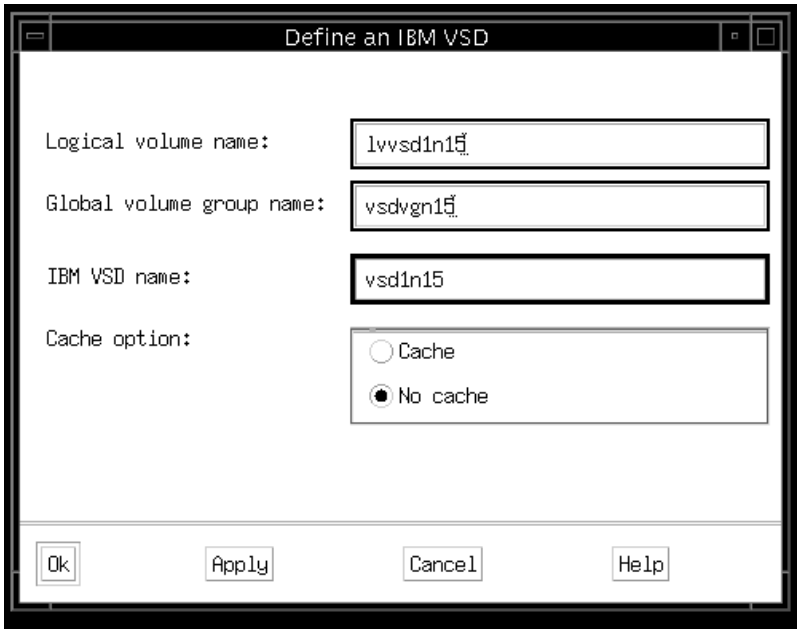


Figure 168. Perspectives Panel for Defining VSDs

4. Configure the VSDs.

   This step takes the VSDs that have been created and defined on a server and makes them available for use on all the nodes that are going to read from and write to them.

   In Perspectives, there are two options to carry out this task. You can select the VSDs you want to configure, then the action **Configure**. Alternatively, you can select the nodes on which you want to configure the VSD, then use the action **Configure IBM VSDs**. For SMIT, run `smit vsd_mgmt` and select the **Configure a VSD** option. The command line equivalent is `cfgvsd`.

   Figure 169 on page 401 is an example of the Perspectives panel for configuring VSDs.

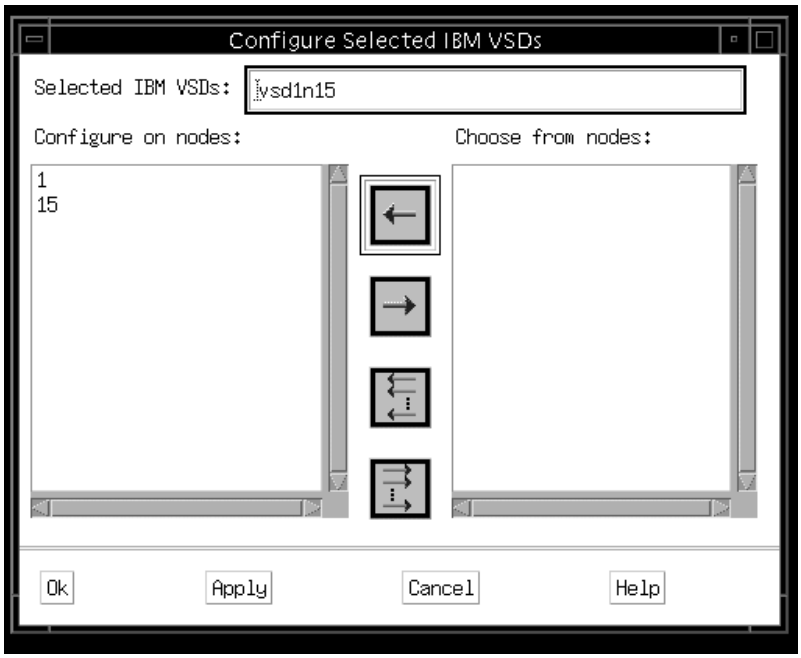


Figure 169. Perspectives Panel for Configuring VSDs

5. Activate the VSDs.

   This is the step that puts the defined and configured VSDs into the active, or available state. In order for a VSD to be used, it must be in the active (available) state on both the server and client nodes.

   In Perspectives, there are two choices. Select the VSD nodes, then run the action **Change IBM VSDs State**, or select the VSDs and run the action **Change State**. For SMIT, run `smit vsd_mgmt` and select the **Start a VSD** option. For the command line, run `startvsd`.

   Figure 170 on page 402 is an example of the Perspectives panel for activating VSDs.

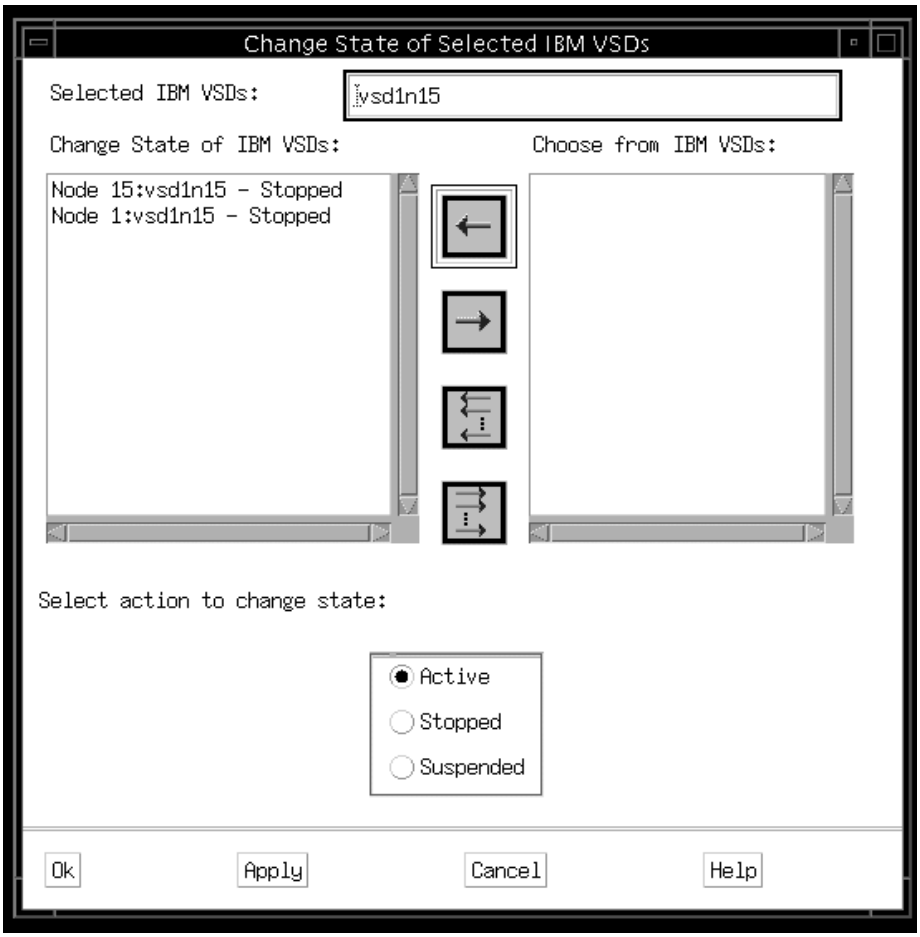


Figure 170. Perspectives Panel for Activating VSDs

Once activated, VSDs have resource variables, which may be monitored to provide operational statistics. Monitoring is set up by administrators in the PSSP Event Perspectives, and it is described in detail in *IBM Parallel System Support Programs for AIX: Managing Shared Disks*, SA22-7349.

There are five different states in which a VSD can be found, depending on the circumstances in the SP system at that time. These circumstances include changes to the configuration (either on a server node or the entire SP system) and problems in the system, application or network. By moving the VSDs into different states, the system is better able to keep track of I/O

operations, for example, and record how far to roll back to properly recover from failures. These states are summarized in Figure 171:

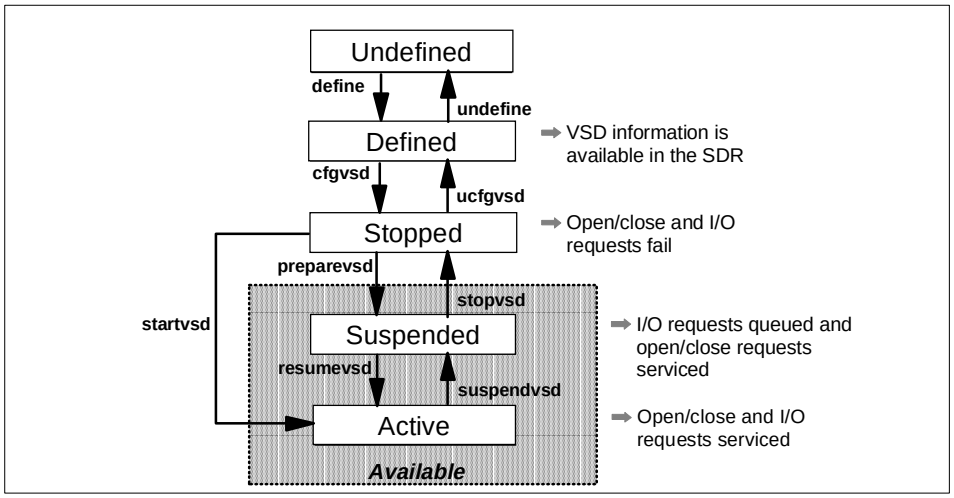


Figure 171. VSD States and Commands

For more information on the commands `vsdnode`, `createvsd`, `defvsd`, `cfgvsd`, `preparevsd`, `stopvsd`, `resumevsd`, `suspendvsd`, and `startvsd`, refer to *IBM Parallel System Support Programs for AIX: Command and Technical Reference*, SA22-7351.

In the past, any changes made to VSDs required the system administrator to first stop the VSDs, unconfigure them from all VSD nodes, make the changes, re-configure the VSDs, and re-start them. This is at best a tedious task. In PSSP 3.1, improvements have been made so that the following functionalities can be dynamically carried out; that is, there is no need to stop and re-start the VSDs:

1. The addition and subtraction of VSD nodes
2. The addition and subtraction of VSDs running on a node
3. Turning on or turning off the cache option
4. Increasing the size of individual VSDs

In addition, a separate filesystem is now used to store all VSD configuration files and logs, pending the availability of disk space. This filesystem is called /var/adm/csd. This feature alleviates the space utilization problem in the /var file system. If there is insufficient disk space to have a separate filesystem, a directory with the same name is created.

Although VSD provides distributed data access, it does not provide a locking mechanism to preserve data integrity. This task falls upon the application itself.

VSDs can be run over any IP network. However, the SP Switch network is the only communication device available on the RS/6000 SP capable of providing the necessary bandwidth and scalability for VSD to operate at good performance. The SP Switch permits:

- High I/O bandwidth for optimal VSD performance
- Scalable growth for any applications using VSDs

14.3.2 Hashed Shared Disks (HSD)

VSDs running over the SP switch can enable efficient access to raw LVs within an SP system. However, there may be instances where the performance bottleneck lies with the I/O capacity within an SP node itself. To help alleviate this problem, IBM provides Hashed Shared Disk (HSD). HSD is a striped version of VSD. The HSD architecture is shown in Figure 172.

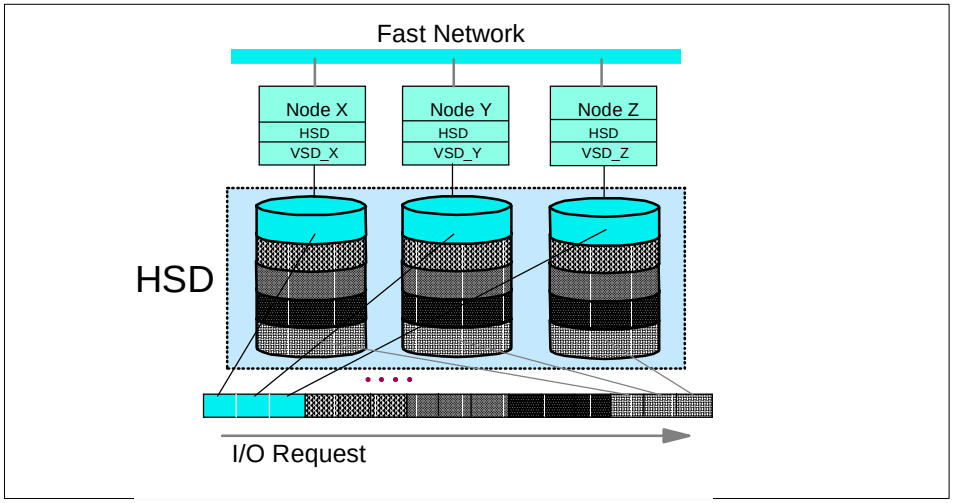


Figure 172. HSD Architecture

HSD adds a device driver above the VSD layer on individual nodes. When an I/O request is made by the application, the HSD device driver breaks it down into smaller blocks, depending upon the strip size specified and the number of VSDs making up the HSD. The strip size defines the amount of data which is read from or written out to one VSD per instruction. The I/O request is then distributed among all the VSDs which make up the HSD for handling.

HSDs are created or defined in a manner similar to the steps outlined in 14.3.1, “Virtual Shared Disks (VSD)” on page 395. Instead of creating, defining, or configuring VSDs, select the same options for HSDs. In this case, the commands are `createhsd`, `defhsd`, and `cfghsd`. The only significant difference is that an HSD activates automatically upon configuration; that is, there is no need to run step 4 (`startvsd`).

HSD helps to spread the data over several VSD and VSD nodes, thereby reducing the chance of I/O performance bottlenecks on individual nodes. It is, however, difficult to manage because any changes, either at the HSD level or at the VSD level, require the deletion and re-creation of the HSD. Other restrictions with HSD are documented in *IBM Parallel System Support Programs For AIX: Managing Shared Disks*, SA22-7349.

14.3.3 Recoverable Virtual Shared Disks (RVSD)

Let us refer again to Figure 165 on page 395. In this figure, there are two VSDs and two VSD nodes. What if Node Y has to be shut down for maintenance? Users and processes on Node X then have no way of accessing the data in lv_Y. This problem may be alleviated by twin-tailing the disks holding lv_X and lv_Y, so that the two VSD nodes are connected to the two VSDs, and using the software Recoverable Virtual Shared Disk (RVSD) to provide a failover mechanism between the two nodes.

RVSD designates nodes which serve VSDs as *primary nodes*. The backup nodes are called *secondary nodes*.

In the example in Figure 165, with RVSD installed and configured, if Node Y has to be shut down for maintenance, then lv_Y is going to failover and be controlled by Node X. While Node Y is down, Node X becomes a server to both VSDs lv_X and lv_Y. When Node Y is back up and operational, RVSD returns back to it the control of lv_Y. This failover is transparent to the users and processes in Node X.

There are different versions of the RVSD software for different versions of PSSP. Each version of RVSD can interoperate with each of the supported levels of PSSP as shown in Table 20.

Table 20. RVSD Levels Supported by PSSP Levels

	PSSP 2.2	PSSP 2.3	PSSP 2.4	PSSP 3.1
RVSD 1.2	Y	Y	Y	Y
RVSD 2.1	N	Y	Y	Y
RVSD 2.1.1	N	N	Y	Y

Ex.1010 / Page 429 of 572

	PSSP 2.2	PSSP 2.3	PSSP 2.4	PSSP 3.1
RVSD 3.1	N	N	N	Y

RVSD is set to operate with the functionality of the lowest PSSP level installed in the SP system, regardless of whether this node is running RVSD or not. To overcome this problem, PSSP 3.1 introduces the command `rvsdrestrict`, which allows system administrators to set the functionality level of RVSD. A detailed description of `rvsdrestrict` in *PSSP 3.1 Announcement*, SG24-5332.

RVSD contains two subsystems: rvsd and hc. Subsystem rvsd controls the recovery for RVSD, while the hc subsystem supports the development of recoverable applications.

The rvsd subsystem records the details associated with each of the VSD servers: the network adapter status, the number of nodes active, the number of nodes required for quorum, and so on.

Quorum in RVSD is defined as the minimum number of active nodes required to continue serving the VSDs. The default setting is half the number of VSD nodes plus one (a majority). If too many nodes become inactive (for example, shutdown), the quorum is lost and the VSDs are stopped. It is possible to change this setting via the `ha.vsd` command.

The rvsd subsystem handles three types of failures: node, disk cable and adapters, and communication adapters. Node failures are described later in this section. Disk cable and adapter failures are also handled as node failures, but only for those VGs which are affected. For example, if there are two global VGs served by one primary node, and one disk within one of the VG fails, only that VG is going to be failed over to the secondary node. The primary node will continue to serve the other VG. When the disk has been replaced, it is necessary to manually run the `vsdchgserver` command to change the control back to the primary node. Note that rvsd only handles communication adapter failures for the Ethernet and SP switch networks. It is possible to run RVSD using other types of network adapters, but there is no failover mechanism in those instances. Communication adapter failures are handled in the same manner as node failures due to RVSD's dependence upon an IP network to function.

The hc subsystem, also called the Connection Manager, shadows the rvsd subsystem, recording the same changes in state and management of VSDs that rvsd records. There is, however, one difference: hc records these changes *after* rvsd has processed them. This ensures that RVSD recovery

events have a chance to begin and complete before hc client application events are carried out. In turn, this serialization helps to preserve data integrity.

Another way to preserve data integrity during application recovery is the concept of fencing, available only with RVSD.

If a recoverable database application is running in a system and one of the running nodes fails, any data locked by the failed application instance is returned to a consistent state. A failed node, however, may still issue I/O requests to the VSDs. It is important that such requests do not get executed, because data may get corrupted. An application's recovery script may use the command `fencevsd` to prevent I/O operations from failed nodes to VSDs in the system. The syntax for this command is `fencedvsd -v [name of vsd] -n [node number or node list]`. For example, to fence node 13 from a vsd named vsd1n15, run `fencevsd -v vsd1n15 -n 13`.

When the failed node is active again, the application's recovery script can issue `unfencevsd` to permit it to issue VSD I/Os. The syntax is similar to that for `fencevsd`.

The command `lsfencevsd` may be run to display a map of all fenced nodes and the VSDs from which they are fenced.

Further information about `ha.vsd`, `fencevsd`, `unfencevsd`, `lsfencevsd` and the rvsd and hc subsystems can be found in *IBM Parallel System Support Programs for AIX: Managing Shared Disks*, SA22-7349.

For problem determination purposes, RVSD keeps a set of logs under /var/adm/csd, which may be gathered with the `vsd.snap` command. Two particular files to focus upon for debugging are the vsd.log and vsd.debuglog files. The vsd.log keeps track of the events that are taking place during a failover, while vsd.debuglog provides additional details like group membership.

Using rvsd, RVSD handles a node failover as a two-event process: vsd.down1 and vsd.down2.

Consider a scenario when a primary node goes down. The first event logged is vsd.down1. At vsd.down1, the primary node runs `suspendvsd` and `stopvsd` to bring its VSDs into the stopped state. On all non-primary nodes (both secondary servers and clients), `suspendvsd` is run to bring the VSDs into the suspend state. This ensures no further I/O activities to the VSDs, and thereby maintains data integrity. Refer to Figure 171 on page 403 for a pictorial review of VSD states.

At vsd.down2, varyoffvg is run on the primary node against the VG to which the VSDs belong. After this, the secondary node runs exportvg, importvg and varyonvg against the same VG to break the primary node's reservation, followed by preparevd and resumevsd to bring the VSDs back to the active state. The VSDs are now served by the secondary node. Client nodes then run resumevsd to bring the VSDs to the active state so they can access them. Finally, rvsd decreases its count of up nodes by 1.

Figure 173 summarizes the RVSD node failover events.

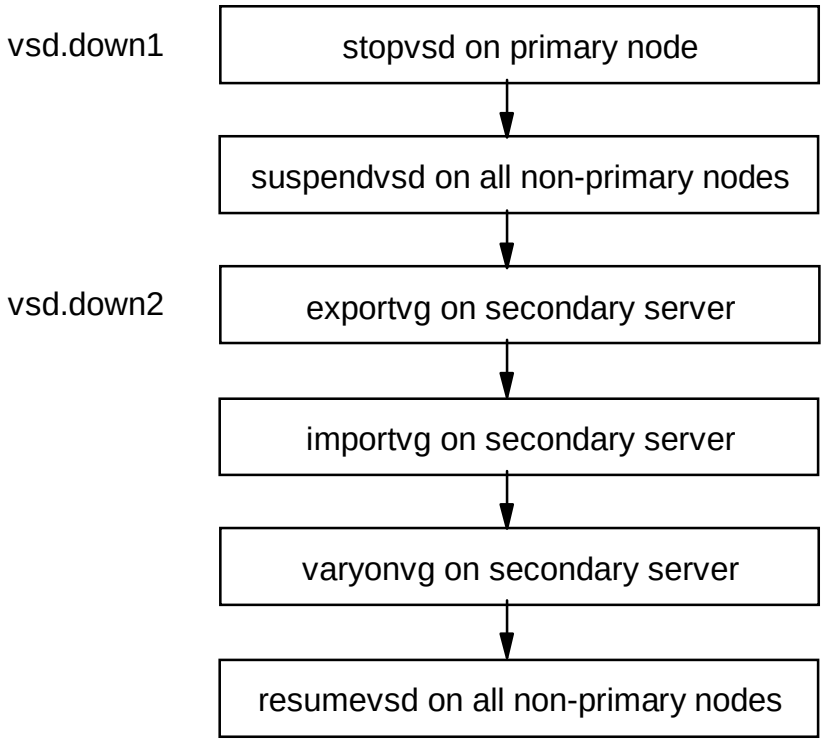


Figure 173. Summary of RVSD Failover Events

When the primary node is back up and running, there is again a two-step process to return control of its VSDs.

Event vsd.up1 has the clients running suspendvsd and the secondary node running both suspendvsd and stopvsd against the VSDs to stop all I/O activities.

Ex.1010 / Page 432 of 572

Event vsd.up2 then has the secondary node running `varyoffvg` against the VG to which the VSDs belong, followed by the primary node running `exportvg`, `importvg` and `varyonvg` against the same VG to reestablish reservation. Then `preparevsd` and `resumevsd` on the primary node and `resumevsd` on all other nodes are run to return the VSDs to the active state for normal operations. Finally, rvsd increases its count of up nodes by 1.

Figure 174 summarizes the RVSD node recovery events.

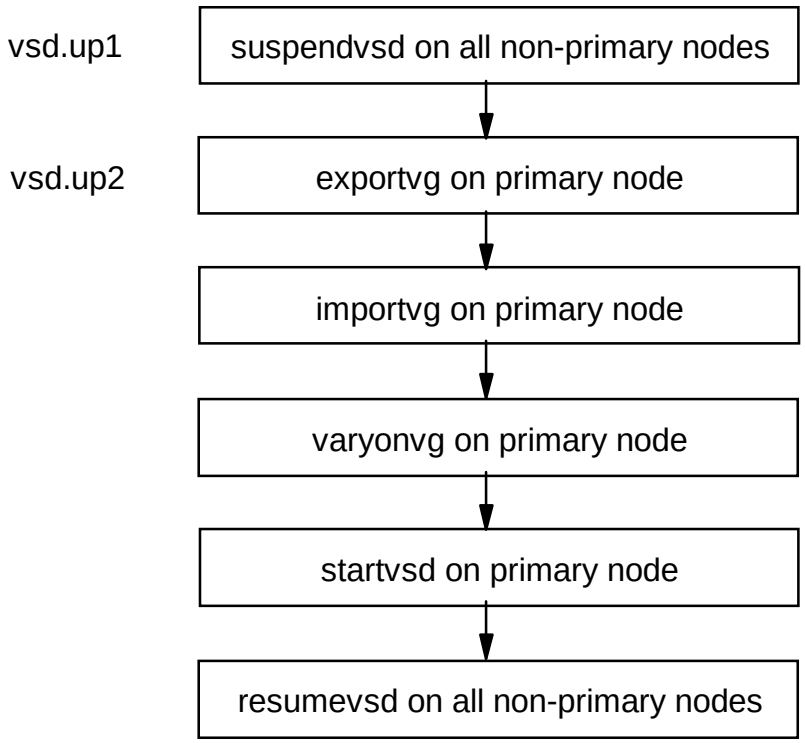


Figure 174. Summary of RVSD Recovery Events

14.4 General Parallel File System (GPFS)

Another method for sharing data globally within an SP system is the IBM-developed General Parallel File System (GPFS). GPFS provides a file system which is accessible by multiple users and by processes running on multiple nodes.

GPFS exploits the VSD technology to built a file system over multiple hard disks on multiple nodes. This file system is then mountable for use by a number of designated nodes. Using a token management system, GPFS enables concurrent read and write access to files. Therefore, both serial and parallel applications can use GPFS.

Although VSDs can be implemented over any IP network, GPFS is only supported over the SP Switch. This is because GPFS is designed for high performance, and the SP Switch offers the highest bandwidth and speed possible within an SP system. Part of GPFS's high performance design involves the striping of data across multiple disks on multiple nodes. This provides multiple servers for one file system, spreading and balancing I/O requests.

Because GPFS spreads a file system across many nodes and disks, it is possible to create very large file systems to hold very large files.

Additional features of GPFS include the capability to add or delete disks, even while the file system is mounted; a dynamic restripe of the file system to improve data block allocations; the use of a log on each GPFS node to track transactions. This log is then replicated on another GPFS node to enhance recovery from errors.

GPFS configurations are stored in the SDR on the control workstation. When a GPFS node boots, it checks the SDR to see if there have been changes. If so, these are applied during the boot. This permits changes to be made to GPFS even if not all the GPFS nodes are up and running.

Figure 175 on page 411 shows the software architecture of GPFS.

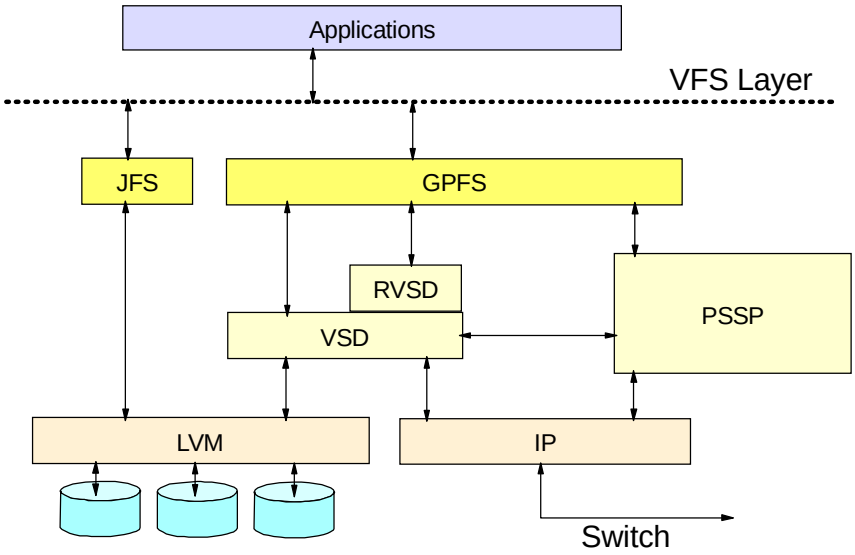


Figure 175. GPFS Software Architecture

The GPFS device driver is a kernel extension which sits between the application and r/vsd layers. It uses standard AIX Virtual File System (VFS) system calls, making it possible for most AIX applications to run over GPFS without modifications. The application views a GPFS file system similar to an AIX Journal File System (JFS). In operations, when GPFS fetches data, it uses the r/vsd device driver to go to the VSD node which has the data. This design makes it possible to separate nodes involved with GPFS into GPFS nodes and VSD server nodes. GPFS nodes are those which mount and use a GPFS while disk server nodes are those which have the underlying VSDs configured on them. The GPFS device driver does not need to know where the data is stored; this is left to the r/vsd device driver. Using the r/vsd device driver makes it possible to separate GPFS nodes into two types: the GPFS nodes, which are those that mount and use GPFS; and the VSD server nodes, which have the underlying hard disks configured as VSDs. The advantage with this is that the VSD servers, unless an application requires it, are not further burdened with running both VSD and GPFS software.

Underneath these layers, GPFS uses the traditional UNIX file system structure of i-nodes, indirect blocks and data blocks to store files. A detailed discussion of these components and their impact on a GPFS file system can be found in *IBM General Parallel File System for AIX: Installation and Administration Guide*, SA22-7278.

Nodes that run GPFS are grouped together to form a GPFS domain, which is managed by the mmfsd daemon, described in 14.4.1, “The mmfsd Daemon” on page 412.

A restriction with current version of GPFS is that it does not handle memory-mapped files. It is expected to be resolved in a future version.

The GPFS filesets (LPPs) consists of 6 different filesets, all beginning with mmfs. This prefix refers to GPFS’s development alongside IBM’s Multi-Media LAN Server product. These filesets are:

- mmfs.base.usr.3.1.0.0
- mmfs.gpfs.usr.1.2.0.0
- mmfs.util.usr.3.1.0.0
- mmfs.msg.en_US.usr.3.1.0.0
- mmfs.man.en_US.shr.3.1.0.0
- mmfs.gpfsdocs.shr.3.1.0.0

RVSD software is a prerequisite to GPFS. Although GPFS can operate without the high availability of disks provided by RVSD, there are commands within RVSD that GPFS needs, such as `fencevsd` and `unfencevsd`. These commands are described in 14.3.3, “Recoverable Virtual Shared Disks (RVSD)” on page 405.

The procedure to install the GPFS filesets and the prerequisites are described in *IBM General Parallel File System for AIX: Installation and Administration Guide*, SA22-7278.

14.4.1 The mmfsd Daemon

At the heart of the GPFS software lies the GPFS daemon (mmfsd). This daemon runs on every GPFS node to coordinate, synchronize and manage GPFS-related duties. Such duties include: data and metadata management (disk space allocation, data access, disk I/O operations and so on), security and quota management. To carry out all of these duties, an mmfsd daemon can take on four different roles or personalities, as follows:

Configuration Manager

In this role, the mmfsd daemon is responsible for configuration tasks within a GPFS domain, such as selecting the Stripe Group Manager for each file system and determining whether quorum exists.

The concept of quorum in GPFS is similar to that found in LVM. At least half the number of nodes, plus one, in the GPFS domain must be up and running. If quorum is lost, the Configuration Manager unmounts the file system, suspending further operations to maintain data consistency and integrity.

There is one GPFS Configuration Manager per system partition. It is the first node to join the group Mmfs Group in Group Services (for more information on Group Services, refer to 8.5, “Group Services (GS)” on page 193). If this node goes down, Group Services selects the next oldest node in the Mmfs Group to be the Configuration Manager.

Stripe Group Manager

A stripe group consists of the set of hard disks which makes up a GPFS file system. There is one Stripe Group Manager per GPFS file system to perform the following services:

- Process changes to the state or description of the file system
  1. Add/delete/replace disks
  2. Change disk availability
  3. Repair file system
  4. Restripe file system
- Control disk region allocation

In addition, the Stripe Group Manager handles the token requests for file access, passing each request to the Token Manager Server for processing.

The Configuration Manager is responsible for selecting the Stripe Group Manager. It avoids overloading any one node in a system partition by selecting a different node for each GPFS file system to act as the Stripe Group Manager.

This selection may be influenced by the GPFS administrator. Create the file /var/mmfs/etc/cluster.preference and list in it the switch hostnames of the nodes, one per line, that you want to be a Stripe Group Manager. The Configuration Manager looks for this file, and if found, selects a node from this list. There is no order of priority given to the nodes in cluster.preference; the only requirement is that the node listed in the file be up and available when the selection is made.

If the node that is the Stripe Group Manager goes down, another node is selected to take over. The selection comes from another node in the cluster.preference file, if it is used, or simply another node in the system

partition. In either instance, priority is given to those nodes that are not serving as a Stripe Group Manager.

Metadata Manager

A Metadata Manager maintains the integrity of the metadata for open files within a GPFS file system. There is one Metadata Manager for each open file. The Metadata Manager is the only node that can update the metadata pertaining to the open file, even if the data in the file is updated by another node.

The Metadata Manager is selected to be the first node that opened the file. This node stays as the Metadata Manager for the file until one of following events occur:

- The file is closed everywhere.
- The node fails.
- The node resigns from the GPFS domain.

When the Metadata Manager goes down, the next node to use the metadata service takes over as the Metadata Manager.

Token Manager Server

The use of tokens helps GPFS manage and coordinate file access among the GPFS nodes. In turn, this preserves data integrity. The concept of tokens in GPFS is similar to that of file locks in LVM. There is one Token Manager Server per GPFS file system. It resides in the same node as the Stripe Group Manager, and processes all requests for access to files within the GPFS.

There is a token manager component which runs on every GPFS node. When that node wants to access a file, its token manager requests a token from the Token Manager Server. The Token Manager Server determines if any locking conflicts exist among any previously granted tokens and the current request.

If no conflicts exist, a token is granted.

If conflicts exist, the Token Manager Server sends a list called a *copy set* back to the requesting node's token manager. It is then the token manager's responsibility to process the copy set and negotiate with the token manager(s) on the node(s) in the copy set to obtain a token and access the file.

This type of negotiation among nodes helps to reduce the overhead on the Token Manager Server.

Ex.1010 / Page 438 of 572

For availability purposes, there are two copies of every token: one at the Token Manager Server and one at the token manager.

If a node goes down, it can recover by having its token manager request a copy of the token from the Token Manager Server.

If the Token Manager Server node goes down, the new Token Manager Server (which is also the node that is the new Stripe Group Manager) then obtains the client copy of all tokens from all the token managers in the GPFS domain.

14.4.2 GPFS Setup

Chapter 2 of *IBM General Parallel File System for AIX: Installation and Administration Guide*, SA22-7278 is devoted to a series of steps for planning a GPFS implementation. It is recommended that this section be read and understood prior to installing and using GPFS.

GPFS tasks cannot be done on the control workstation; they must be performed on one of the GPFS nodes.

There are three areas of consideration when GPFS is being set up: the nodes using GPFS, the VSDs to be used, and the file systems to be created. These areas are discussed in detail in the remainder of this section, with reference to a sample file system setup consisting of four nodes. Nodes 12, 13 and 14 are GPFS nodes, while node 15 is the VSD server node.

Important

Do not attempt to start the mmfsd daemon prior to configuring GPFS. Starting the mmfsd daemon without configuring GPFS causes dummy kernel extensions to be loaded and you will be unable to create a file system. If this occurs, configure GPFS and then reboot the node(s).

Carry out the following procedures to configure GPFS, then start the mmfsd daemon to continue creating the file system.

Nodes

The first step in setting up GPFS is to define which nodes are GPFS nodes. The second step is to specify the parameters for each node.

There are three areas where nodes can be specified for GPFS operations: node count, node list, and nodes preferences.

The *Node Count* is an estimate of the maximum number of nodes that will mount the file system, and it is entered into the system only when the GPFS file system is created. It is recommended that you overestimate this number. This number is used in the creation of GPFS data structures that are essential for achieving the maximum degree of parallelism in file system operations. Although a larger estimate consumes a bit more memory, insufficient allocation of GPFS data structures can limit a node's ability to process certain parallel requests efficiently, such as the allotment of disk space to a file. If it is not possible to estimate the number of nodes, apply the default value of 32. A larger number may be specified if more nodes are expected to be added.

However, it is important to avoid wildly overestimating, since this can affect buffer operations. This value cannot be changed later. If you need to change this value, the file system must be destroyed and recreated using the new value.

A *node list* is a file which specifies to GPFS the actual nodes to be included in the GPFS domain. This file may have any filename. However, when GPFS configures the nodes, it copies the file to each GPFS node as /etc/cluster.nodes. The GPFS nodes are listed one per line in this file, and the switch interface is specified because this is the interface over which GPFS runs.

Figure 176 on page 417 is an example of a node list file. The filename in this example is /var/mmfs/etc/nodes.list.

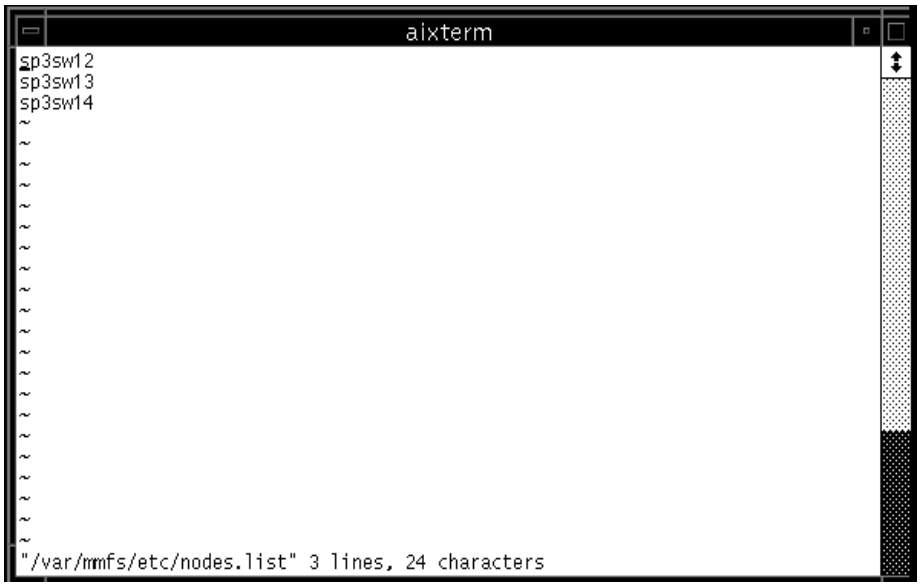


Figure 176. Sample Node List File

Once the nodes which form the GPFS domain are chosen, there is the option to choose which of these nodes are to be considered for the personality of stripe group manager (see stripe group manager in 14.4.1, “The mmfsd Daemon” on page 412). There are only three nodes in the GPFS domain in this example, so this step is unnecessary. However, if there are a large number of nodes in the GPFS domain, it may be desirable to restrict the role of stripe group manager to a small number of nodes. This way, if something happens and a new stripe group manager has to be chosen, GPFS can do so from a smaller set of the nodes (the default is every GPFS node) particularly if some node configurations are better suited to the role. To carry this out, follow the format for creating a node list to create the file /var/mmfs/etc/cluster.preferences (this filename must be followed).

To configure GPFS, you can use SMIT panels or the `mmconfig` command. The `mmconfig` command is further described in *IBM General Parallel File System by AIX: Installation and Administration Guide*, SA22-7278. The SMIT panel may be accessed by typing `smit gpfs` and then selecting the **Create Startup Configuration** option. Figure 177 on page 418 shows the SMIT panel used to configure GPFS (this is being run on node 12 in our example). This step needs to be run only on one node in a GPFS domain.



Figure 177. SMIT Panel for Configuring GPFS

It is possible to configure GPFS to automatically start on all nodes whenever they come up. Simply specify yes to the autoload option in the SMIT panel or the -A flag in the `mmconfig` command. This eliminates the need to manually start GPFS when nodes are rebooted.

The pagepool and mallocsize options specify the size of the cache on each node dedicated for GPFS operations. mallocsize sets an area dedicated for holding GPFS control structures data while pagepool is the actual size of the cache on each node. In this instance, pagepool is specified to the default size of 4M while mallocsize is specified to be the default of 2M, where M stands for megabytes and must be included in the field. The maximum values per node are 512 MB for pagepool and 128 MB for mallocsize.

The priority field refers to the scheduling priority for the mmfsd daemon. The concept of priority is beyond the scope of this book: refer to AIX documentation for more information.

Notice the file /usr/lpp/mmfs/samples/mmfs.cfg.sample. This file contains the default values used to configure GPFS if none are specified, either through the fields in the SMIT panel or in another file. The use of another file to set GPFS options may appeal to more experienced users, or to those who want to configure multiple GPFS domains with the same parameters. Simply copy this file (/usr/lpp/mmfs/samples/mmfs.cfg.sample) to a different file, make the changes according to your specifications, propagate it out to the nodes, and configure using SMIT or the `mmconfig` command.

Further information, including details regarding the values to set for pagepool and mallocsize, is in *IBM General Parallel File System for AIX: Installation and Administration Guide*, SA22-7278.

Once GPFS has been configured, the mmfsd daemon has to be started on the GPFS nodes before a file system can be created. The steps to do this are as follows:

1. Set the WCOLL environment variable to target all GPFS nodes for the `dsh` command. *IBM Parallel Systems Support Programs: Administration Guide*, SA22-7348, *IBM Parallel Systems Support Programs: Command and Technical Reference*, SA22-7351, and *IBM RS/6000 SP Management, Easy, Lean, and Mean*, GG24-2563, all contain information on the WCOLL environment variable.
2. Designate each of the nodes in the GPFS domain as an IBM VSD node.
3. Ensure that the rvsd and hc daemons are active on the GPFS nodes. **Note**: rvsd and hc do not start unless they detect the presence of one VSD defined for the GPFS nodes. This VSD may or may not be used in the GPFS file system.
4. Start the mmfsd daemon by running the following command on one GPFS node:

   ```
   dsh startsrc -s mmfs
   ```

   The mmfsd starts on all the nodes specified in the /etc/cluster.nodes file. If the startup is successful, the file /var/adm/ras/mmfs.log* looks like Figure 178 on page 420.



*Figure 178. Sample Output of /var/adm/ras/mmfs.log**

VSDs

Before the file system can be created, the underlying VSDs must be set up. The nodes with the VSDs configured may be strictly VSD server nodes, or they can also be GPFS nodes. Consider the application to decide whether to include VSD server-only nodes in the GPFS domain.

You must also decide the level of redundancy needed to guard against failures. Should the VSDs be mirrored? Should they run with a RAID subsystem on top? Should RVSD be used in case of node failures? Again, this depends on the application, but it also depends on your comfort and preferences for dealing with risk.

In addition to these options, GPFS provides two further recovery strategies at the VSD disk level. GPFS organizes disks into a number of failure groups. A failure group is simply a set of disks that share a common point of failure. A common point of failure is defined as that which, if it goes down, causes the set of disks to become simultaneously unavailable. For example, if a VSD

spans two physical disks within one node, the two disks can be considered a failure group because if the node goes down, both disks become unavailable.

Recall that there are two types of data that GPFS handles: metadata and the data itself. GPFS can decide what is stored on each VSD: metadata only, data only, or data and metadata. It is possible to separate metadata and data, to ensure that data corruption does not affect the metadata, and vice versa.

Further, the separation of data and metadata can even enhance performance. This is best seen if RAID is involved. RAID devices are not suited for handling metadata because metadata is small in size and can be handled using small I/O block sizes. RAID is most effective at handling large I/O block sizes. metadata can therefore be stored in a non-RAID environment, such as mirrored disks, while the data can be stored in a RAID disk. This both protects data and metadata from each other, and maximizes the performance given that RAID is chosen.

Once you adopt the redundancy strategy, there are two ways to create VSDs: have GPFS do it for you, or manually create them. For either method, this is done through the use of a Disk Descriptor file. This file can be set up manually or through the use of SMIT panels. If using SMIT, run `smit gpfs` and then select the **Prepare Disk Descriptor File** option. Figure 179 on page 422 shows the SMIT panel for our example.

In this case, the VSD vsd1n15 has already been created on node 15 (sp3n15). *Do not* specify a name for the server node because the system already has all the information it needs from the configuration files in the SDR. In addition, the VSD(s) must be in the Active state on the VSD server node and on all the GPFS nodes prior to the file system creation.

If the VSDs have not been created, specify the name of the disk (such as hdisk3) in the disk name field, instead of vsd1n15, and specify the server where this hdisk is connected. GPFS then creates the necessary VSDs to create the file system.

The failure group number may be system generated or user specified. In this case, a number of 1 is specified. If no number is specified, the system provides a default number that is equal to the VSD server node number plus 4000.

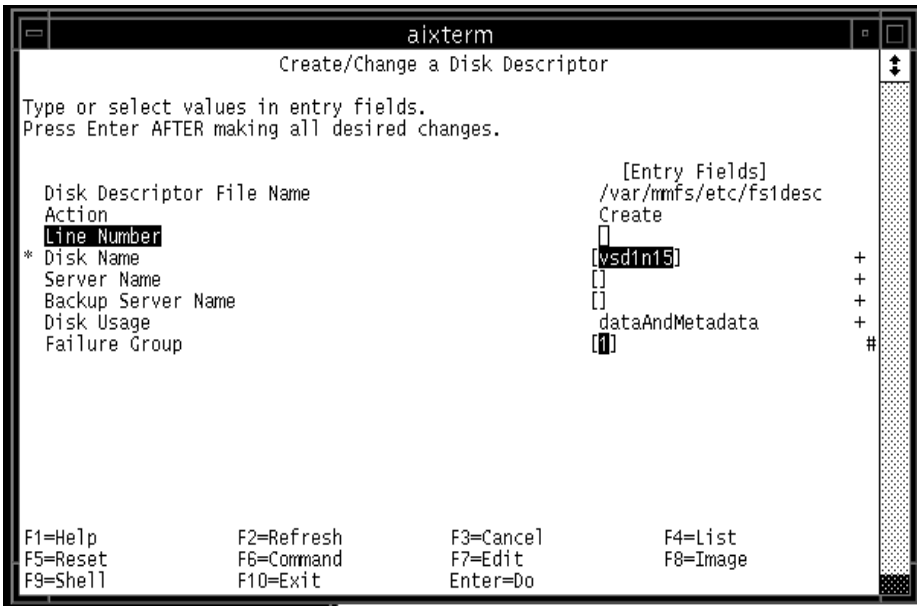


Figure 179. SMIT Panel for Creating Disk Descriptor File

File System

There are two ways to create a GPFS file system: using SMIT panels or the `mmcrfs` command. Figure 180 on page 423 shows the SMIT panel. This is accessed by running `smit gpfs` and then selecting the **Create File System** option. Details on `mmcrfs` can be found in *IBM General Parallel File System for AIX: Installation and Administration Guide*, SA22-7278.

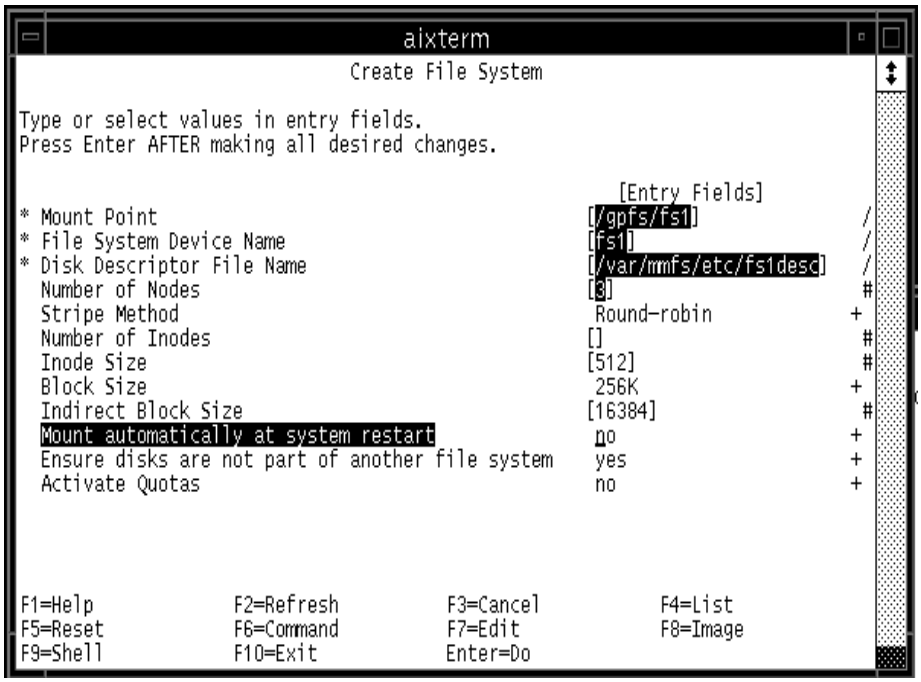


Figure 180. SMIT Panel for Creating a GPFS File System

Four issues must be considered before the file system is created, as follows:

1. How to structure the data in the file system

   There are three factors to consider in structuring the data in the file system: block size, i-node size, and indirect block size.

   GPFS offers a choice of three block sizes for I/O to and from the file system: 16 KB, 64 KB, or 256 KB. Consider the applications running on your system to determine which block size to use. If the applications handle large amounts of data in a single read/write operation, then a large block size may be best. If the size of the files handled by the applications is small, a smaller block size may be more suitable. The default is 256 KB.

   GPFS further divides each block of I/O into 32 sub blocks. If the block size is the largest amount of data that can be accessed in a single I/O operation, the sub block is the smallest unit of disk space that can be allocated to a file. For a block size of 256 KB, GPFS reads as much as 256 KB of data in a single I/O operation, and a small file can occupy as little as 8 KB of disk space.

Files smaller than one block size are stored in fragments, which are made up of one or more sub-blocks. Large files are therefore often stored in a number of full blocks, plus one or more fragments to hold the data at the end of the file.

The i-node is also known as the file index. It is the internal structure that describes an individual file to AIX, holding such information as file size and the time of the last modification to the file. In addition, an i-node points to the location of the file on the hard disk. If the file is small, the i-node stores the addresses of all the disk blocks containing the file data. If the file is large, i-nodes point to indirect blocks which point to the disk blocks storing the file data (indirect blocks are set aside to specifically hold only data block addresses).

The default size of an i-node is 512 bytes. This number can increase to 4 KB, depending on the size of the files the application uses.

An indirect block can be as small as a single sub-block or as large as a full block (up to an absolute maximum of 32 KB). The only additional requirement is that the value of an indirect block is a multiple of the size of a sub-block.

It is also possible to specify the number of i-nodes, which limits the maximum number of files that can be created in the file system. In older versions of GPFS, the maximum number of i-nodes is set at GPFS file system creation time and cannot be changed after. With GPFS 1.2, it is now possible to set a limit at file system creation time, and if it proves necessary, change this upper limit. The upper limit is changed by the `mmchfs` command; the exact syntax can be found in *IBM General Parallel File System for AIX: Installation and Administration Guide*, SA22-7278.

2. Striping method

   GPFS automatically stripes data across VSDs to increase performance and balance disk I/O. There are three possible striping algorithms GPFS can implement: round Robin, balanced Random, and Random. A striping algorithm may be set when a GPFS file system is first created, or can be modified as a file system parameter later on.

   The three algorithms are defined as follows:

   - round Robin

     This is the default option the system specifies. Data blocks are written to one VSD at a time until all the VSDs in the file system have received a data block. The next round of writes will then write another block to each VSD *in exactly the same order.*

This method yields the best write performance. There is, however, a penalty when a disk is added or removed from the file system. When a disk is added or removed from the file system, a restriping occurs. The round Robin method takes the longest amount of time among the three algorithms to handle this restriping.

- balanced Random

  This method is similar to round Robin. When data blocks are written, one block is written to each VSD. When all the VSDs have received one block of data, the round begins. However, in balanced Random, the order in the second round is not the same as the first round. Subsequent rounds are similarly written to all VSDs, but in an order different from that of the previous round.

- Random

  As its name implies, there is no set algorithm for handling writes. Each data block is written to a VSD according to a random function. If data replication is required, GPFS does ensure that both copies of the data are not written to the same disk.

3. Whether or not to use GPFS quotas

   GPFS quotas define the amount of space in the file system that a user or a group of users is allowed to use. There are three parameters with which quotas operate: hard limit, soft limit, and grace period.

   The hard limit is the maximum disk space and files that a user or group can accumulate. Soft limits are the levels below which a user or group can safely operate. A grace period is only used for soft limits; it defines a period of time in which a user or group can exceed the soft limit.

   The usage and limits data are stored in the quota.user and quota.group files that reside in the root directories of GPFS file systems.

   In a quota-enabled configuration, one node is automatically nominated as the quota manager whenever GPFS is started. The quota manager allocates disk blocks to the other nodes writing to the file system, and compares the allocated space to the quota limits at regular intervals. In order to reduce the need for frequent space requests from nodes writing to the file system, the quota manager allocates more disk blocks than requested.

   Quotas can be turned on by switching the Activate Quotas entry (shown Figure 180 on page 423) to Yes, or by specifying the -Q yes flag for the `mmcrfs` command.

   Quotas are further discussed in *IBM General Parallel File System for AIX: Installation and Administration Guide*, SA22-7278.

4. Whether or not to replicate the files

   At the file system level, GPFS provides an option to have additional copies of data and metadata stored on the VSDs. This is above and beyond disk mirroring. Therefore, with both replication and mirroring turned on, it is possible to have a maximum of four copies of data being written.

   It is possible to replicate metadata, data, or both. The parameters for this are Max Meta Data Replicas and Max Data Replicas, which control the maximum factors of replication of metadata and data respectively, and Default Meta Data Replica and Default Data Replicas, the actual factors of replication. Acceptable values are one or two. One is the default and means no replication (only one copy) and two means replication is turned on (two copies). The Default values must be less than or equal to the Max values. In other words, the Max values grant permission for replication, while the Default values turn the replication on or off.

   Replication can be set at file system creation time and *cannot* be set via SMIT panels. The only way to turn on replication is with the command `mmcrfs` and the flags -M for Max Metadata Replicas, -m for Default Metadata Replicas, -R for Max Data Replicas and -r for Default Data Replicas. Using the same example in Figure 180 on page 423, we can create a file system with both metadata and data replication turned on:

   ```
   mmcrfs /gpfs/fs1 fs1 -F /var/mmfs/etc/fs1desc -A yes -B 256K -i 512 -I
   16K -M 2 -m 2 -n 3 -R 2 -r 2 -v yes
   ```

   More information on these flags and the `mmcrfs` command can be found in *IBM General Parallel File System for AIX: Installation and Administration Guide*, SA22-7278.

Once a GPFS file system has been set up, it can be mounted or unmounted on the GPFS nodes using the AIX `mount` and `umount` commands. Or, you can use the SMIT panel by running `smit fs` and then selecting **Mount File System**. Figure 181 on page 427 shows the SMIT panel for mounting a file system.

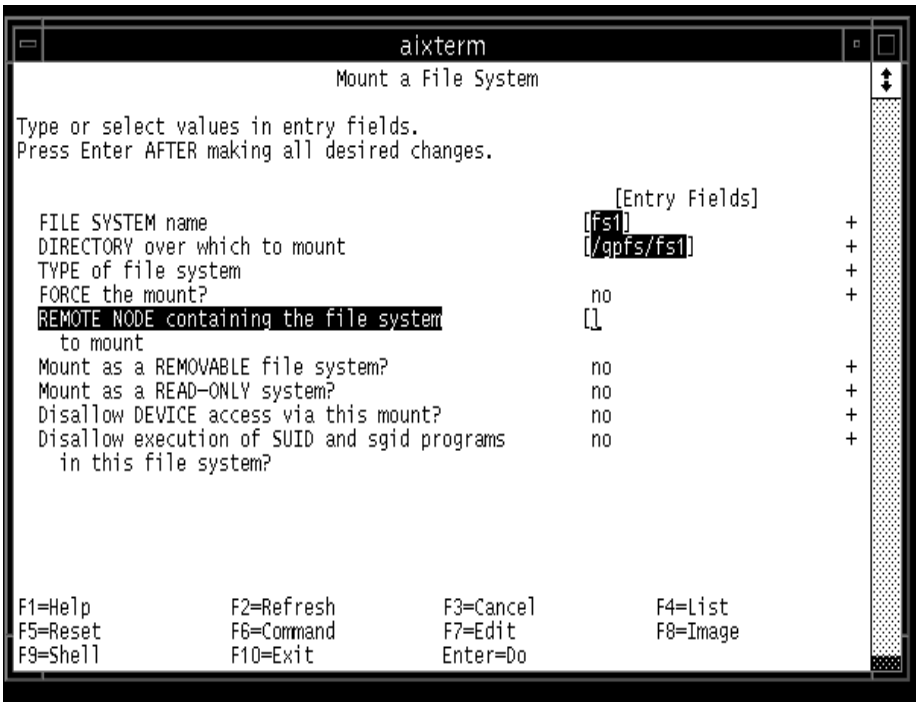


Figure 181. SMIT Panel for Mounting a File System

14.4.3 Managing GPFS

Once a GPFS file system has been set up, a number of tasks can be performed to manage it. Some of the tasks and the commands to execute them are discussed here. Note that SMIT panels are also available to execute the commands. The commands and the SMIT panels are further described in the manual *IBM General Parallel File System for AIX: Installation and Administration Guide*, SA22-7278.

Changing the GPFS Configuration

It is possible to change the configuration of GPFS for performance tuning purposes. The command `mmchconfig` and it is capable of changing the following attributes:

- pagepool
- data Structure Dump
- mallocsize

- maxFiles To Cache
- priority
- autoload

Changes to pagepool may take effect immediately if the -i option is chosen, otherwise the changes will take effect the next time GPFS is started. Changes to data Structure Dump, mallocsize, maxFiles To Cache, and priority require restart of GPFS. Changes to autoload require a reboot of the nodes where this is affected.

For example, to immediately change the size of pagepool to 60 MB, run

```
mmchconfig pagepool=60M -i
```

It is also possible to add and delete nodes from a GPFS configuration. The commands to do so are `mmaddnode` and `mmdelnode`. However, use caution when adding or subtracting nodes from a GPFS configuration, because GPFS uses quorum to determine if a GPFS file system stays mounted or not, and it is easy to break the quorum requirement when adding or deleting nodes.

Note that adding or deleting nodes automatically configures them for GPFS usage. Newly added nodes are considered GPFS nodes in a down state, and are not recognized until a restart of GPFS. By maintaining quorum, you ensure that you can schedule a good time to refresh GPFS on the nodes.

For example, consider a GPFS configuration of four nodes. The quorum is three. With all four nodes running, we can add or delete one node and the quorum requirement is still satisfied. We can add up to three nodes into the GPFS group, as long as all four current nodes stay up. If we try to add four nodes, the GPFS group consists then of eight nodes, with a quorum requirement of five. However, at that point, GPFS can only see four nodes up (configured) and exits on all the current nodes.

Deleting a File System

The command to delete a file system is `mmdelfs`. Before you delete a GPFS file system, however, you must unmount it from all GPFS nodes.

For example, if you want to delete fs1 (shown in Figure 180 on page 423), you can run `umount fs1` on all GPFS nodes, then run `mmdelfs fs1`.

Checking and Repairing a File System

If a file system cannot be mounted or if messages are received saying that a file cannot be read, it is possible to have GPFS check and repair any

repairable damages to the file system. The file system has to be in the unmounted state for GPFS to check it.

The command `mmfsck` checks for and repairs the following file inconsistencies:

- Blocks marked allocated that do not belong to any file. The blocks are marked free.
- Files for which an i-node is allocated, but no directory entry exists. `mmfsck` either creates a directory entry for the file in the /lost+found directory, or it destroys the file.
- Directory entries pointing to an i-node that is not allocated. `mmfsck` removes the entries.
- Ill-formed directory entries. They are removed.
- Incorrect link counts on files and directories. They are updated with the accurate counts.
- Cycles in the directory structure. Any detected cycles are broken. If the cycle is a disconnected one, the new top level directory is moved to the /lost+found directory.

File System Attributes

File system attributes can be listed with the `mmlsfs` command. If no flags are specified, all attributes are listed. For example, to list all the attributes of fs1, run:

```
mmlsfs fs1
```

To change file system attributes, use the `mmchfs` command. The following eight attributes can be changed:

- Automatic mount of file system at GPFS startup
- Maximum number of files
- Default Metadata Replication
- Quota Enforcement
- Default Data Replication
- Stripe Method
- Mount point
- Migrate file system

For example, to change the file system to permit data replication, run:

```
mmchfs -r 2
```

Querying and Changing File Replication Attributes

The command `mmlsattr` shows the replication factors for one or more files. If it is necessary to change this, use the `mmchattr` command.

For example, to list the replication factors for a file /gpfs/fs1/test.file, run

```
mmlsattr /gpfs/fs1/test.file
```

Say the value turns out to be 1 for data replication and you want to change this to 2, run:

```
mmchattr -r 2 /gpfs/fs1/test.file
```

Re striping a GPFS File System

If disks have been added to a GPFS, you may want to restripe the file system data across all the disks to improve system performance. This is particularly useful if the file system is seldom updated, for the data has not had a chance to propagate out to the new disk(s). To do this, run:

```
mmrestripefs
```

There are three options with this command; any one of the three must be chosen. The -b flag stands for rebalancing. This is used when you simply want to restripe the files across the disks in the file system. The -m flag stands for migration. This option moves all critical data from any suspended disk in the file system. Critical data is all data that would be lost if the currently suspended disk(s) are removed. The -r flag stands for replication. This migrates all data from a suspended disk and restores all replicated files in the file system according to their replication factor.

For example, when a disk has been added to fs1 and you are ready to restripe the data onto this new disk, run:

```
mmrestripefs fs1 -b
```

Query File System Space

The AIX command `df` shows the amount of free space left in a file system. This can also be run on a GPFS file system. However, to obtain information

Ex.1010 / Page 454 of 572

regarding how balanced the GPFS file system is, the command to use is `mmdf`. This command is run against a specific GPFS file system and shows the VSDs which make up this file system and the amount of free space within each VSD.

For example, to check on the GPFS file system fs1 and the amount of free space within each VSD which houses it, run:

```
mmdf fs1
```

14.4.4 Migration and Coexistence

The improvements in GPFS 1.2 make it necessary that all nodes in a GPFS domain be at the same level of GPFS code. That is, in a GPFS domain, you cannot run both GPFS 1.1 and 1.2.

It is, however, possible to run multiple levels of GPFS codes, provided that each level is in its own group, within one system partition.

There are two possible scenarios to migrate to GPFS 1.2 from previous versions: full and staged. As its name implies, a full migration means that all the GPFS nodes within a system are installed with GPFS 1.2. A staged migration means that certain nodes are selected to form a GPFS group with GPFS 1.2 installed. Once you are convinced by this test group that it is safe to do, you can migrate the rest of your system.

Migration and coexistence are further described in *PSSP 3.1 Announcement*, SG24-5332, and *IBM General Parallel File System for AIX: Installation and Migration Guide*, SA22-7278.

Chapter 15. High Availability

The RS/6000 SP system is a popular choice for businesses because of its scalability, flexibility and manageability. Its potential for high uptime also contributes to its value in the marketplace. Two software products that contribute to high system uptime are available on the RS/6000 SP: High Availability Cluster Multiprocessing Enhanced Scalability (HACMP/ES) and High Availability Cluster Multiprocessing Enhanced Scalability Concurrent Resource Manager (HACMP/ESCRM).

While fault-tolerant systems provide for high availability through expensive hardware redundancy, IBM's HACMP/ES and HACMP/ESCRM software provide a low-cost commercial computing environment that ensures mission-critical applications can recover quickly from hardware and software failure. This is a significant advantage over the fault-tolerant systems where software failures are not taken care of. Moreover, in fault-tolerant systems, the redundant hardware components do no processing in normal circumstances. While fault-tolerant systems surpass high availability systems in terms of service interruptions, high availability stands out for its low-cost, minimal service interruptions and flexibility of configuration. Shared resources, such as shared IP addresses for high network availability and shared volume groups for data availability, provide the foundation of the high availability features of the HACMP family of products.

HACMP/ES supports up to 32 SP nodes or RS/6000 systems. Development is underway to enhance the scalability to support up to 128 nodes. HACMP/ESCRM adds on to HACMP/ES, providing concurrent shared-access management for supported RAID and SSA subsystems. It supports up to 8 SP nodes or RS/6000 systems.

15.1 Components and Relationships

HACMP/ES and HACMP/ESCRM rely on the RS/6000 Cluster Technology (RSCT) for event detection and heartbeat. Initially part of the IBM PSSP for AIX Availability Services, RSCT is now an integral part of the HACMP/ES software. Refer to Chapter 8, "RS/6000 Cluster Technology" on page 185 for details on RSCT.

With this enhancement, HACMP/ES and HACMP/ESCRM can support these configurations:

- A cluster of standalone RS/6000 systems
- A mixed cluster of RS/6000 systems and RS/6000 SP nodes

© Copyright IBM Corp. 1999

- A cluster of RS/6000 SP nodes in different system partitions or different RS/6000 SPs

15.2 Modes of Operation

HACMP/ES supports two types of configurations: *cascading* and *rotating.* In these configurations, each shared volume group can belong to any one system, but not to more than one at a time. It comes with a Cluster Single-Point-of-Control (C-SPOC) facility that does dynamic reconfiguration of shared volume groups to ensure consistency of information across the cluster nodes. With this new introduction, common cluster administration tasks can be performed from any node in the cluster.

Another useful feature of HACMP/ES is the Dynamic Automatic Reconfiguration Event (DARE). This utility improves cluster management by allowing the placement alteration of resource groups to specific cluster nodes using the `cldare` command.

In a cascading configuration, priority is given to each resource group. The node with the highest priority for a given resource group will take precedence over that group. If the node fails, the node with the next highest priority will take over that resource group. When a node with a higher priority rejoins the cluster, it takes over the resource group.

In a rotating configuration, priority is given on a first-come-first-served basis. The node that joins the cluster first will own those resource groups to which it has access. It releases those resource groups only when it fails or leaves the cluster with the takeover option turned on. The next node in line will then take control of the resource group. Upon rejoining, the original node that owned the resource group does not attempt to take back the resource group. So, in a rotating configuration, a resource group does not have a fixed node it attaches to.

HACMP/ESCRM adds to HACMP/ES by providing *concurrent* access to shared volume groups. This is to say more than one system can access a shared volume group at the same time. A distributed locking mechanism must be in place to prevent data contention. One of the restrictions on the use of concurrent volume groups is that it handles only raw logical volumes. With this restriction in place, the type of disk subsystems becomes limited. Supported IBM devices include:

- 7133 and 7131-405 SSA disk subsystems in non-RAID configurations
- 9333 disk subsystems

- 7135-110 and 7135-210 RAIDiant arrays
- 7137 Disk Arrays

15.3 Planning Considerations

When configuring HACMP/ES or HACMP/ESCRM, allow an adequate amount of time for planning. Proper planning ensures easier installation and administration, higher availability, better performance, and less interruption to the cluster. This section discusses the various areas of consideration and provides suggestions on how to properly configure a HACMP/ES or HACMP/ESCRM environment. The rest of this section will use HACMP/ES as a reference to both HACMP/ES and HACMP/ESCRM unless otherwise stated.

15.3.1 Cluster

With HACMP/ES, you can include up to 32 nodes in a single SP partition. Alternatively, you can define clusters that are not contained within a single SP partition. Consider the following points when planning for cluster nodes:

- Nodes that have entirely separate functions and do not share resources should not be combined in a single cluster. Instead, create several smaller clusters on the SP. Smaller clusters are easier to design, implement, and maintain.
- For performance reasons, it may be desirable to use multiple nodes to support the same application. To provide mutual takeover services, the application must be designed in a manner that allows multiple instances of the application to run on the same node.
- In certain configurations, including additional nodes in the cluster design can increase the level of availability provided by the cluster; it also gives you more flexibility in planning node fallover and reintegration.

15.3.2 Application

Application availability is the key item to look at when considering the purchase of high availability products. Plan carefully in this area before implementing HACMP/ES to take full advantage of its capabilities. Some considerations that must be considered include:

- The application and its data should be laid out such that only the data resides on shared external disks while the application resides on each node that is capable of running it. This arrangement prevents software license violations and simplifies failure recovery. Ensure each node has

the required number of licenses available to run the application. However, if the application is not license-bound (for example, in-house developed applications), then having it on the shared external disks will be a better choice from an administrative point of view because the administrator will then need to contend with only one copy of the application.

- Start and stop scripts should be robust enough to handle the application on all related nodes. They should be able to handle abnormal termination of the application, such as node crash, providing speedy recovery. These scripts must be able to handle unsuccessful starting or stopping of applications. Failure to exit completely from a script constitutes an event error in HACMP/ES.

  Certain event errors will cause HACMP/ES to hang in a limbo state and refuse to carry out all other tasks that follow. We recommend that you not include interactive statements in these scripts, as there is no facility for user inputs. HACMP/ES start, stop and takeover should be automatic and should be as transparent to users as possible.

  A good practice when implementing such scripts is to test them fully on a node and then on another node without having HACMP/ES software involvement. HACMP/ES software merely kicks off the scripts in the order you tell it to; it does not carry out any recovery procedure on its own. Successful implementation of these scripts depends on how well they are written.

- Modularize the start and stop scripts. In debugging, this helps to determine where a problem lies. Whenever possible, let HACMP/ES call one start and one stop script. Within the start and stop scripts, each will call function statements to start or stop particular applications. Having one start and one stop script simplifies administrative tasks. Having modules, or functions, within the scripts make debugging and changes easier.

- Consider performance when deciding which nodes to use for takeover functions. An overloaded node will not be able to handle additional workload from another failed node. Forcing it to take over more workload will affect performance not only of applications from the failed node but also of its own. Where possible, spread out the load between processors, keeping in mind the amount of load each has to take when it takes over other nodes' responsibilities. Hardware upgrades may be required to boost performance.

15.3.3 Network

In a cluster environment, each node must have network connectivity to other nodes. This is required for communications between nodes to update each

other's status. Client machines are served through these network channels to access the applications. Keep in mind the following considerations when planning for your network layout:

- Eliminate the network as a single point of failure.

  Having more than one network available to client machines insures against inaccessibility of nodes due to network failure. This, however, means added cost to network hardware. Supported TCP/IP-based networks include:

  - Ethernet (802.3 Ethernet not supported)
  - Token-Ring
  - Fiber Distributed Data Interchange (FDDI)
  - ATM and ATM LAN Emulation

  SOCC or SLIP networks are not supported in HACMP/ES cluster.

  On the SP, there are 3 SP-specific networks available:

  - SP Ethernet

    This network is used for software installation and administrative purposes only and should not be included in the HACMP/ES for external access by clients. However, it can be included in HACMP/ES for use as a heartbeat network

  - SP Switch

    This high speed network connects all SP nodes and is generally used for large data transfers. It is configurable in HACMP/ES for IP Address Takeover (IPAT).

  - SP Serial Network

    Spanning across all nodes and the CWS is a serial network. It is used for administrative purposes only and is not configurable under HACMP/ES.

- Eliminate the TCP/IP subsystem as a single point of failure.

  It is important that nodes within a cluster are aware of each other's status. They keep each other updated through all the network channels. Networks that rely on TCP/IP for communication run a risk of failure on the TCP/IP subsystem. A cluster that is totally reliant on TCP/IP networks for heartbeat transmission will break down once the TCP/IP subsystem fails. To prevent such a failure, a non-IP network should be present in the cluster. Currently, support for non-IP networks includes target mode SCSI (tmscsi), target mode SSA (tmssa) and serial (rs232) networks.

- Eliminate Network Adapters as a single point of failure.

  Having multiple networks guards against network failures. However, where cost is a concern, there may only be one network available for access by clients. In such a case, we can at least protect ourselves from network adapter failures by having more than just one network adapter. In an IP Take Over (IPAT) environment, for each service adapter, there has to be an equivalent standby adapter on the same node. The SP Switch network is an exception where IPAT is concerned. It cannot have a standby adapter. Failure on an SP Switch adapter can be promoted to a node failure through the use of the AIX error notification facility depending on how important the SP Switch is.

15.3.4 Data Loss Protection

In order to protect against data loss, we have to decide on a mechanism to eliminate data loss as a single point of failure. Two ways to ensure against data loss are as follows:

- Logical Volume Manager Mirroring

  The logical volume manager (LVM) in AIX provides a means of mirroring all logical volumes. *Mirroring* means having more than one copy of a physical partition associated with a logical partition. These copies can either reside on the same hard disk or they can reside on different hard disks. It is advisable that mirrored copies reside on different hard disks to prevent loss of data in the event of a hard disk crash. With mirroring in place, should an error occur in one copy of the physical partition, the remaining copy can still function. This means uninterrupted access to the data. Mirroring applies to both logical volumes as well as journal logs.

  In a *non-concurrent* configuration, we advise you to have quorum turned off for volume groups with mirroring. This provides continuous access to data even when some copies of data are bad. On restarting HACMP/ES, it also prevents volume groups with missing disks from being activated (varyonvg).

  In a *concurrent* configuration, quorum must be enabled. Disabling quorum may result in data corruption. Multiple failures, that result in no common shared disk between cluster nodes, have the potential for data corruption or inconsistency because each node may be accessing different copies of the same data and thus updating them differently.

Ex.1010 / Page 462 of 572

- Redundant Array of Independent Disks (RAID)

  IBM 7135 models 110 and 210 provide supported RAIDiant technology in an HACMP/ES environment. It contains a group of hard disks that work together to provide large storage capacity and high I/O rates.

  In a RAID configuration, LVM mirroring must not be used to mirror the same set of data. There are four levels in which it can operate:

  - RAID level 0

    Data is striped across a bank of disks in the array to improve throughput. It does not provide data redundancy and is not recommended for use in HACMP/ES clusters.

  - RAID level 1

    Data redundancy is maintained in multiple copies of the data on separate disks. This is similar to LVM mirroring, but is done at the hardware level.

  - RAID level 3

    This is used with raw disks only and is seldom used.

  - RAID level 5

    Data redundancy is maintained through the use of parity bits that allow data on a particular failed drive to be reconstructed from existing drives. This is the most commonly used mode. However, with this mode, about 20 to 30 percent storage overhead is expected in maintaining parity information.

To avoid having adapters be a single point of failure, we recommend that original and mirrored hard disks be connected to separate adapters on the same system. This is illustrated in Figure 182 on page 440.

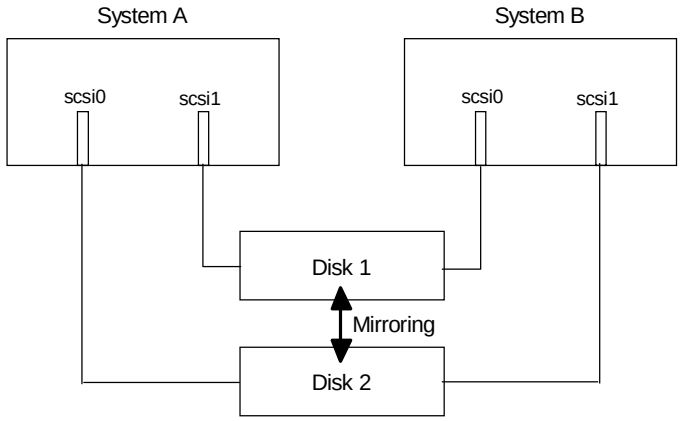


Figure 182. Avoiding Adapters and Disks as Single Point of Failure

15.3.5 Resource Group

HACMP/ES needs to know how to manage each of the resources that is configured. Resources refer to volume groups, service IP addresses, file systems, network file systems, and application servers (including start and stop scripts). Resource groups are created to indicate to HACMP/ES where each resource belongs. While HACMP/ES does not dictate the number of resources that go into a resource group or the number of resource groups in a cluster, it is advisable to keep the design simple. Following are a few general considerations to keep in mind:

- Every cluster resource must be part of one (and only one) resource group.
- You should group resources that cater to the same application in the same resource group.
- A rotating resource group must have a service IP label defined for it.
- A cascading resource group need not necessarily contain a service IP label. If IPAT is required, then the service IP label must be included in one of the resource groups.
- If a node belongs to more than one resource chain, it must have the ability to manage all resource groups simultaneously.
- Cascading or rotating resource groups cannot contain concurrent volume groups.
- Concurrent resource groups contain only application servers and concurrent volume groups.

15.4 Cluster Configuration

With proper planning in place, we can proceed with setting up the HACMP/ES cluster. Users familiar with *HACMP for AIX* will have no problem understanding the steps that follow. Along the way, features associated with HACMP/ES are discussed. The information provided here is not meant to be exhaustive. For details on setup procedure, refer to *HACMP for AIX: Enhanced Scalability Installation and Administration Guide*, SC23-4284.

15.4.1 Server

This section explains some of the terms used in HACMP/ES as well as the general guidelines you have to follow during configuration of the servers.

15.4.1.1 Networks

There are three types of networks to consider when configuring HACMP/ES.

Private Network

A private network is considered a network that is inaccessible by outside hosts. For the SP complex, this refers to the SP Administrative Ethernet and the SP Switch.

- SP Administrative Ethernet

  It is advised that the SP Administrative Ethernet be used only as a private network within the SP complex. In HACMP/ES, it is configured as a private network providing a heartbeat channel for keepalive packets.

- SP Switch

  The SP Switch is also classified as a private network in HACMP/ES with connections to SP nodes. Unlike the old High Performance Switch (HiPS) where HACMP/ES has to manage the Eprimary takeover, the SP Switch does not allow HACMP/ES to control the Eprimary.

  When using SP Switch in HACMP/ES, Address Resolution Protocol (ARP) must be enabled. To find out if a node has ARP turned on, use the following command:

  ```
  # dsh -w <host> "/usr/lpp/ssp/css/ifconfig css0"
  ```

  If NOARP appears on the output, it reflects that ARP is turned off for that node and will require steps to turn it on. There are two methods to turn ARP on.

  The first method requires the node to be customized after setting the SDR to enable ARP. Refer to *Parallel System Support Programs for AIX: Administration Guide*, SA22-7348 for the procedure.

The second method requires modifications to the node's ODM to turn it on manually. Do this with the following steps:

> **Important**
>
> The steps listed must be followed exactly and completely. Ensure that the CuAt file is backed up as directed. Without this backup copy, any error made may end up with the re-installation of the node. Mistakes made can be fixed by replacing the CuAt file with the backup copy and then rebooting the node. When in doubt, do not use this method; use the first method to customize the node.

1. Backup the node's CuAt.

   ```
   # dsh -w <host> "cp -p /etc/objrepos/CuAt /etc/objrepos/CuAt.save"
   ```

2. Save css0 ODM information with ARP value "yes" to temporary file.

   ```
   # dsh -w <host> "odmget -q 'name=css and attribute=arp_enabled' \
   CuAt | sed s/no/yes/ > /tmp/arpon.css"
   ```

3. Replace ODM information for css0 with the ARP value set to "yes".

   ```
   # dsh -w <host> "odmchange -q 'name=css and attribute=arp_enabled' \
   -o CuAt /tmp/arpon.css"
   ```

4. Verify that the ODM has the ARP value set to "yes".

   ```
   # dsh -w <host> "odmget -q 'name=css and attribute=arp_enabled' CuAt"
   ```

5. Remove the temporary file.

   ```
   # dsh -w <host> "rm /tmp/arpon.css"
   ```

- Asynchronous Transfer Mode (ATM)

  ATM is considered a private network since it provides a point-to-point connection between two systems and does not do broadcasting. It uses connection-oriented technology as opposed to IP, which is a datagram-oriented technology. As such, configuring IP addresses as on ATM is complicated. With an ATM switch, only Switched Virtual Circuits (SVC) can be used. Hardware address takeover is not supported in IPAT. An ATM ARP server cannot be an HACMP/ES server.

Public Network

Public networks are used by external hosts to access the cluster nodes. They can be classified under the following categories:

- Ethernet, Token-Ring, and FDDI

Ex.1010 / Page 466 of 572

These adapters are normally used for external connectivity to client machines. When configured for IPAT, each service adapter will be accompanied by at least one similar standby adapter. Up to seven standby adapters can be configured for each network.

- ATM LAN Emulation

  With ATM configured, ATM LAN emulation can be configured to bridge the ATM network with existing Ethernet or Token-Ring networks. Each adapter configured can take the form of either ethernet or token-ring. Like the classic ATM, hardware address takeover is not supported. The network configured is classified as public under HACMP/ES.

Serial Network

Serial networks do not rely on TCP/IP for communication. As such, they provide a reliable means of connection between cluster nodes for heartbeat transmissions.

- RS232

  This is the most commonly used non-IP network, as it is cheap and reliable. It is a crossed (null modem) cable connected to a serial port on the node. Earlier model nodes like the 62MHz Thin (7012-370), 66MHz Thin (7012-390), 66MHz Wide (7013-590), 66MHz Thin2 (7012-39H), 77MHz Wide (7013-591) and 66MHz Wide (7013-59H) do not have supported native serial ports and therefore require the use of the 8-port or 16-port asynchronous adapters. Extension nodes S70 and S7A will require PCI multi-port asynchronous adapters to support this feature.

- TMSCSI

  Like the RS232 connection, TMSCSI is non-IP reliant. It is configured on the SCSI-2 Differential bus to provide HACMP/ES clusters with a reliable network for heartbeat transmission.

- TMSSA

  The TMSSA is a new feature of HACMP/ES, which allows SSA Multi-Initiator RAID adapters to provide another channel for keepalive packets. This feature is supported only when PTF2 or greater is installed.

15.4.1.2 Shared Devices and LVM Components

HACMP/ES requires that shared volume groups be configured on external disks accessible by cluster nodes sharing the same volume group information. Supported disk subsystems include:

- Small Computer System Interface (SCSI)
- Redundant Array of Independent Disks (RAID)

- Serial Storage Architecture (SSA)
- Versatile Storage Server (VSS)

Take note that single-ended SCSI disks cannot be used to configure shared volume groups. When configuring for twin-tail access of SCSI-2 Differential disks, be careful to avoid having a SCSI address clash on the same SCSI bus.

Consider the following when configuring shared volume groups and LVM components for Non-Concurrent Access:

- All shared volume groups must have the auto-varyon feature turned off.
- The major number for a shared volume group must be the same on all cluster nodes using NFS mounted file systems. However, when using NFS, it is also recommended that the major number be kept the same for easier management and control. The `lvlstmajor` command can be used to check for available major numbers on each node.
- All journaled file system logs, logical volume names and file system mount points must be unique throughout the cluster nodes.
- All file systems must have the auto-mount feature turned off.
- For shared volume groups with mirroring, we recommend that quorum is turned off.

LVM components for Concurrent Access cannot contain journaled file systems, since concurrent mode supports only raw logical volumes. In this mode, major number for the shared volume group is not important.

The procedure for making concurrent capable volume groups on serial disk subsystems differs from that on RAID disk subsystems. Refer to *HACMP for AIX: Enhanced Scalability Installation and Administration Guide*, SC23-4284 for details on how to configure the different disk subsystems for concurrent access.

15.4.1.3 Preparing AIX

In order for HACMP/ES to be configured properly, the following components in AIX must be set correctly:

- Entries in /.rhosts must include all cluster nodes. This is required to allow synchronization of HACMP/ES configuration information between cluster nodes. During synchronization, the /usr/es/sbin/cluster/godm daemon is used. HACMP/ES is equipped to take advantage of the Kerberos authentication if configured on the systems. In such a setup, the /.rhosts file is not required.

- User and Group IDs must be the same on all nodes. This is to allow users to be able to login on the takeover system in case their main server fails. With the RS/6000 SP, user management can be used to enforce this. In HACMP/ES, groups and users management is made easy through the use of these commands: `cl_mkuser, cl_lsuser, cl_chuser, cl_rmuser, cl_mkgroup, cl_lsgroup, cl_chgroup, cl_rmgroup`.

  They can be accessed via the SMIT fastpath:

  ```
  # smitty cl_usergroup
  ```

  Ensure that all users' home directories are created on all nodes if they are not located on the shared disks.

- All hostnames must be resolvable. Either the /etc/hosts file or the nameserver must contain all IP addresses referenced by HACMP/ES.
- The following network options must be set to 1: *nonlocsrcroute, bcastping, ipsrcroutesend, ipsrcrouterecv, ipsrcrouteforward.*
- I/O pacing needs to be turned on for a HACMP/ES cluster to work properly during large disk writes. A high water mark of 33 and a low water mark of 24 is recommended for a start. Failure to turn on I/O pacing may result in nodes crashing during large disk write activities. To do so, use the following command:

  ```
  # chdev -l sys0 -a maxpout=33 -a minpout=24
  ```

- A syncd value of 10 is recommended. Syncd has the function of flushing the contents of the RAM and writes to disk what has been updated. In many cluster setups, nodes crash due to syncd not releasing the jfs_sync_lock during high node activities, thus preventing clstrmgr from processing heartbeats. To make the necessary change, search for the following sentence in /sbin/rc.boot and modify the default value of 60 to 10.

  ```
  nohup /usr/sbin/syncd 10 > /dev/null 2>&1 &
  ```

15.4.1.4 Installing HACMP/ES

When installing the software, ensure free space of approximately 52 MB in /usr and about 500 KB in / (root) file systems. The following AIX filesets are prerequisites for HACMP/ES:

- bos.adt.lib
- bos.adt.libm
- bos.adt.syscalls
- bos.net.tcp.client
- bos.net.tcp.server

- bos.rte.SRC
- bos.rte.libc
- bos.rte.libcfg
- bos.rte.libcur
- bos.rte.libpthreads
- bos.rte.odm
- bos.rte.lvm.usr 4.3.2 or higher (for CRM)

Install the components that you require for your nodes, bearing in mind the minimum required filesets are:

- rsct.basic.*
- rsct.clients.*
- cluster.es.*
- cluster.cspoc.* (automatically installed when cluster.es.* is being installed)
- cluster.clvm.* (for CRM only)
- cluster.hc.* (for CRM only)

The latest Program Temporary Fixes (PTFs) must be installed together with the base filesets. Without PTFs, certain functionality mentioned in this book may not be applicable. One example is the support of TMSSA.

15.4.1.5 Cluster Information Program Clinfo

The clinfo daemon provides status information about the cluster to cluster nodes and clients. Each node has a copy of /usr/sbin/cluster/etc/clhosts file. This file is required by the clinfo daemon to know which nodes or clients can be communicated. It contains resolvable hostnames or IP addresses of all boot and service labels of servers and clients. Take care to not leave this file empty or include standby addresses, as this may affect the way clinfo or clstat work.

When an event occurs in the cluster, the /usr/sbin/cluster/etc/clinfo.rc script is executed to update the system's ARP cache. Include all client IP addresses or hostnames to the PING_CLIENT_LIST variable. This is especially important where no hardware address swap is configured.

15.4.1.6 Configure Cluster

Configuring HACMP/ES is very similar to configuring HACMP for AIX. The following steps provide a general description of how to configure HACMP/ES.

Ex.1010 / Page 470 of 572

You have to be a root user to do the configurations. Configuration of a cluster must be done on a single node in the cluster. The information will then be propagated to the rest of the nodes with the synchronization features built into the HACMP/ES software.

Define Cluster Topology

To define the cluster topology, the following steps are taken. All the steps are done on one node.

- Define a cluster by providing a cluster ID and cluster name. Note that in any network where there are multiple clusters, each cluster IDs must be unique. Failure to conform to this rule can cause nodes to crash unexpectedly.

  The SMIT fastpath is:

  ```
  # smitty cm_config_cluster
  ```

- Define all the nodes in the cluster. It is advisable to use hostnames as the node names for easy reference.

  The SMIT fastpath is:

  ```
  # smitty cm_config_nodes
  ```

- Define all network adapters. This includes the serial network, service IP labels, boot IP labels and standby IP labels. Note that the SP Switch is configured under Network Type "hps".

  The SMIT fastpath is:

  ```
  # smitty cm_config_adapters
  ```

- Define a global network. In a cluster where nodes belong to different SP partitions, each SP administrative ethernet will be on its own subnets. In HACMP/ES, you must define these subnets as one global network for heartbeat transmission. For example, in Figure 183 on page 448, we have two frames partitioned into Subnet1 (192.168.3.0) and Subnet2 (192.168.4.0).

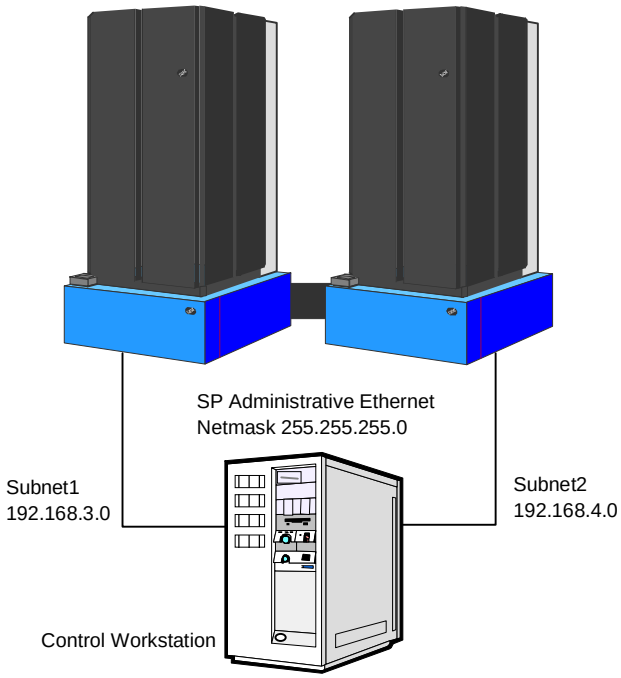


Figure 183. SP Administrative Ethernet on Different Subnets

All routes are defined to ensure that nodes on Subnet1 can communicate to nodes on Subnet2. To ensure heartbeat transmission can flow from nodes on Subnet1 to nodes on Subnet2, we add these 2 subnets to a global network SPGlobal. The following commands show how:

```
# /usr/sbin/cluster/utilities/claddnetwork -u Subnet1:SPGlobal
# /usr/sbin/cluster/utilities/claddnetwork -u Subnet2:SPGlobal
```

The SMIT fastpath is:

```
# smitty cm_config_global_networks.select
```

In case you want to remove Subnet1 from SPGlobal, use this command:

```
# /usr/sbin/cluster/utilities/claddnetwork -u Subnet1
```

- Tune the Network Module. The heartbeat rate can be tuned for all networks defined. If any network is expected to be heavily loaded, the heartbeat rate can be tuned to "slow" to avoid false failure detection over that network. Failure detection is dependent on the *fastest* network in the cluster. For a start, leave the settings as default. Tune only when you experience false failure detections.

- Synchronize the cluster definition to all nodes. You will have to ensure that all hostnames are defined in /.rhosts if you are using the AIX authentication method. If you are using Kerberos authentication across the nodes and want to use this instead of /.rhosts, follow these steps:
  1. Ensure /usr/lpp/ssp/kerberos/bin and /usr/lpp/ssp/rcmd/bin are included in the root user path.
  2. Delete all /.rhosts files.
  3. Set the cluster security mode to Enhanced

     ```
     # /usr/sbin/cluster/utilities/clchclstr -s Enhanced
     ```

     The SMIT fastpath is:

     ```
     # smitty cm_config_security.changeshow
     ```
  4. Run `/usr/sbin/cluster/sbin/cl_setup_kerberos`.
  5. Synchronize cluster definition.
  6. Delete the /usr/sbin/cluster/sbin/cl_krb_service file which contains the Kerberos service principals password which you entered earlier when running the `cl_setup_kerberos` command.

Define Resources

In order to associate all resources with each node, resource groups must be configured. Use the following procedures to set up resources for the cluster:

- Define resource groups. When creating resource groups, it is important to note that when specifying Participating Node Names for cascading or rotating, the priority decreases in the order you specify. For concurrent mode, the order does not matter.
- Define application servers. This will tell HACMP/ES which scripts are used for starting the applications and which to stop.
- Define resources for resource groups. Here, you will input all the related resources which you want HACMP/ES to manage for you. Take note that IPAT is required if you are using the NFS mount option. Also, you must set the value of `Filesystems mounted before IP configured` to `true`.
- Configure run time parameters. If you use NIS or nameserver, set the value to true. Leave the Debug Level as high to ensure that all events are captured.
- Configure cluster events. You can customize the way HACMP/ES handles events by incorporating pre-event, post-event scripts, notification or even recovery scripts. How you want HACMP/ES to react depends on how you configure these events. The default is generally good enough.

- Synchronize the resource definitions.

15.4.2 Client

In order for an RS/6000 system to receive messages about what is happening on the cluster, it must be configured as a client running the clinfo daemon. Use the following steps to do this:

- Install Base Client Software

  Install the following minimum filesets in order for your client system to access any cluster information:

  - rsct.clients.*
  - cluster.es.client.*

- Edit /usr/sbin/cluster/etc/clhosts

  Like the server, this file contains resolvable hostnames or IP addresses of all boot and service labels of servers and clients. The clinfo daemon uses names in this file to communicate with the clsmuxpd process on the servers. This file must *not* be left empty, and you must not include standby addresses in it.

- Edit /usr/sbin/cluster/etc/clinfo.rc

  This file is used the same way as in the servers. Include all client IP addresses or hostnames to the PING_CLIENT_LIST variable. This is especially important where no hardware address swap is configured and where there are clients that do not run the clinfo daemon. Be sure to update the ARP tables on routers and systems that are not running clinfo when IPAT occurs.

- Reboot Clients

  Reboot your clients machines to finish the setup.

15.5 Starting and Stopping Cluster Services

With HACMP/ES, starting and stopping cluster services is made easy with the introduction of CSPOC. Instead of having to start cluster services one node at a time, it is now possible to start some or all nodes from a single node. The command that is used to start cluster services is `/usr/sbin/cluster/sbin/cl_rc.cluster`. This command will call the `/usr/sbin/cluster/etc/rc.cluster` command to start up cluster services on the nodes in a sequential order. The remote execution is basically done through the `rsh` command. The SMIT fastpath is:

Ex.1010 / Page 474 of 572

```
# smitty cl_clstart.dialog
```

The normal `clstart` command to start up one node at a time is still available and is accessible via the SMIT fastpath:

```
# smitty clstart
```

To stop a cluster using CSPOC, the `/usr/sbin/cluster/sbin/cl_clstop` command is used. The SMIT fastpath is:

```
# smitty cl_clstop.dialog
```

To stop the cluster services one node at a time, use the SMIT fastpath:

```
# smitty clstop
```

Note that the **Forced** option has been removed from the SMIT screen and will not be available till later releases of HACMP/ES. The `clstop` script still retains this forced option, but you are advised not to use it.

In order to monitor cluster activity from a client system using the `clstat` command, the clinfo daemon must first be started up. To start it, follow the procedure shown in Figure 184.



Figure 184. Starting Clinfo Daemon on Client System

To start monitoring the cluster status, use the following command:

```
# /usr/sbin/cluster/clstat
```

The output is shown in Figure 185 on page 452.

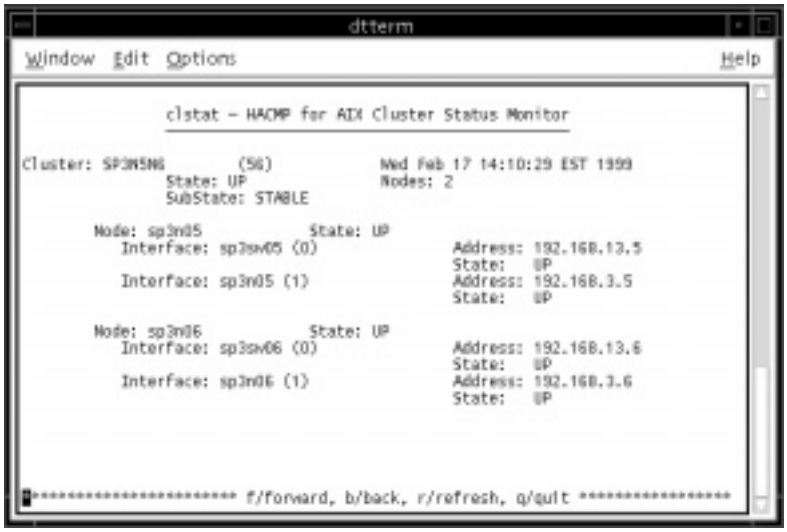


Figure 185. Clstat Output Screen

Chapter 16. Parallel Programming

This chapter describes how the SP can be used for parallel programming. Many of today's scientific and engineering problems can only be solved by using the combined computational power of parallel machines, and some also rely on high performance I/O capabilities. Typically, the applications in this area are highly specialized programs, developed and maintained in universities, government labs, or company research departments.

Parallel Environment for AIX (PE), program number 5765-543, provides the application programmer with the infrastructure for developing, running, debugging, and tuning parallel applications on the SP. In 16.1, "Parallel Operating Environment (POE)" on page 453, we describe the operating environment which allows you to run and control programs in parallel from a single node.

As each SP node is an individual RS/6000 machine with its own memory (invisible to other SP nodes), the programs running on different nodes do not share any data. Therefore, the second core component of PE is the Message Passing Interface (MPI) library, through which programs running on different SP nodes can exchange messages. This enables the otherwise independent programs to work collectively on one common job. MPI is discussed in 16.2, "Message Passing Libraries" on page 463, along with some related subjects.

In addition to explicit message passing by the use of message passing libraries, the data-parallel programming model is also supported on the SP through High Performance Fortran (HPF). We give an overview of HPF and some of the products available on the SP in 16.3, "High Performance Fortran" on page 476.

Finally, the development of parallel programs requires parallel debuggers to localize programming errors, and parallel profiling and tracing tools to find, understand and resolve performance problems. 16.4, "Parallel Programming Tools" on page 480 provides information about these tools.

16.1 Parallel Operating Environment (POE)

The heart of the Parallel Operating Environment (POE) is the `poe` command, which allows you to execute programs in parallel from a single node. POE allocates the nodes on which the application should run (as described in 16.1.4, "POE Resource Management" on page 460), starts the programs on these nodes, and redirects stdin, stdout and stderr between the POE *home*

© Copyright IBM Corp. 1999

node (on which the poe command was invoked) and the *remote nodes* which execute the application.

Before POE can be used, some prerequisites must be met:

- The user must exist with the same (non-root) userid on the home node and all the remote nodes. For security reasons, POE will refuse to run parallel applications as root (UID 0).
- The user must have authorization to run rsh to the remote nodes. On an SP system which is managed and used as a whole, listing all the SP nodes which are used for POE jobs in the /etc/hosts.equiv file is a natural way to ensure this. Using individual users' $HOME/.rhosts files is also possible, of course.
- The desired executable/script must be available on all remote nodes, under the same path name. This can be achieved by starting the application from within a global file system like NFS, DFS, AFS, or even GPFS. Alternatively, the executable can be copied to a local file system on the remote nodes before invoking poe.

Note

Starting POE Jobs on Large Systems: For large SP systems, starting an executable on many nodes from within a shared file system like NFS can cause performance problems. DFS has a better client to server ratio than NFS, and so supports a larger number of nodes that simultaneously read the same executable. As a parallel file system, GPFS is also able to sustain a higher number of clients simultaneously accessing the same executable.

Typically, POE is invoked by specifying the executable/script that should run on the remote nodes as the first argument of the poe command. Alternatively, POE can prompt interactively for the names of the programs to run, or read this information from a commands file. For example, to run the AIX hostname command on four processors and save the resulting output to a file, issue:

```
poe hostname -procs 4   >   ./where_was_i
```

The poe command allocates the remote nodes, and on each of these nodes, it does the following:

- Initializes the local user environment, including a chdir() to the current directory of the home node.
- Runs the command with stdin, stdout and stderr connected to the poe process on the home node.

Ex.1010 / Page 478 of 572

- Passes signals like SIGCONT, SIGINT, SIGQUIT, SIGTERM and SIGHUP to the remote nodes.
- Performs heartbeat checks of the remote nodes (unless the MP_PULSE environment variable is set to zero), and terminates the parallel application if any of the remote nodes are down.
- Gets notifications about abnormal termination of any of the user programs, and in response shuts down the whole parallel application.

The operation of POE can be controlled by a large number of POE environment variables, most of which can also be specified as command line options to poe (following the executable/script name). These options can be freely intermixed with command line options which are interpreted by the application. Before we describe some of these configuration options, an overview about the general POE architecture might be useful.

16.1.1 POE Architecture

POE is a client/server application. The poe command is the client side running on the home node, and communicates with the *partition manager daemon*s (PMDs) running on each of the remote nodes. In PE 2.4, this server component is the pmdv2 daemon. The protocol which POE uses internally is called *Structured Socket Messages* (SSM).

Each partition manager daemon starts and controls the program which runs on its node, and routes that program's stdin, stdout and stderr traffic to the poe command on the home node through SSM. This program may be any executable or (Korn shell) script; it need not be part of a parallel application in which the programs running on different nodes interact with each other.

The POE startup process differs depending on whether the poe command is invoked interactively, or from within LoadLeveler. The interactive case is shown in Figure 186 on page 456.

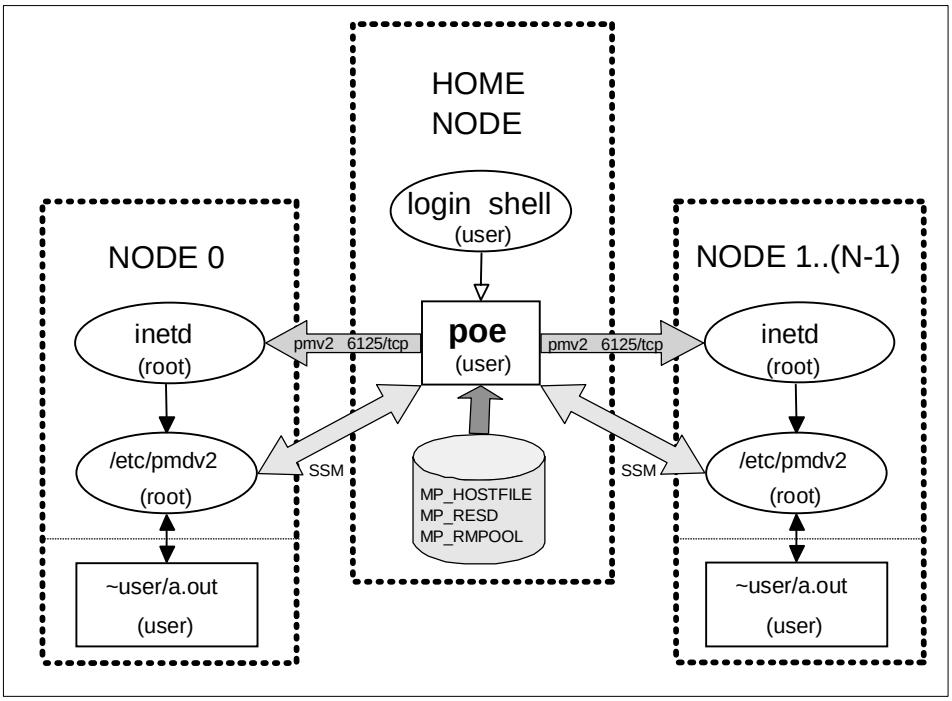


Figure 186. POE Startup for Interactive Use

If poe is called interactively, it determines the remote nodes by the resource allocation methods specified in POE environment variables and command line options. This is explained in 16.1.4, “POE Resource Management” on page 460, and might involve LoadLeveler. However, LoadLeveler is only used to do the resource allocation: the POE application is still interactive.

Note

Home Node versus Node 0: In interactive POE, there is no direct relation between the home node and any of the remote nodes. It is a common misconception that the home node is identical with node 0 of the parallel application. In general, this is not true for interactive POE jobs. The resource allocation mechanism chooses the remote nodes, and the selected set of nodes may or may not include the node on which the poe command was invoked.

The poe command then uses the pmv2 service (normally port 6125/tcp) to start the pmdv2 partition manager daemons on the remote nodes, through

the inetd daemon. The pmdv2 daemon runs under the identity of root, since it must check the user's authentication and set the priorities for the user's task. Finally, pmdv2 starts the application under the user's identity.

If POE is run in a LoadLeveler batch job, it is started differently. The resource allocation as well as the start of the POE application is done by LoadLeveler. On as many batch nodes as requested, LoadLeveler starts a LoadL_starter process under the identity of the user. It does so through the LoadL_master and LoadL_startd processes, which run on the batch nodes under the identity of LoadLeveler.

As shown in Figure 187, the LoadL_starter process on each node starts the pmdv2 partition manager daemon, which in this case runs under the identity of the user since all authentication checks and priority settings have been performed already by the LoadLeveler components. Finally, pmdv2 invokes the user's application as in the interactive case.

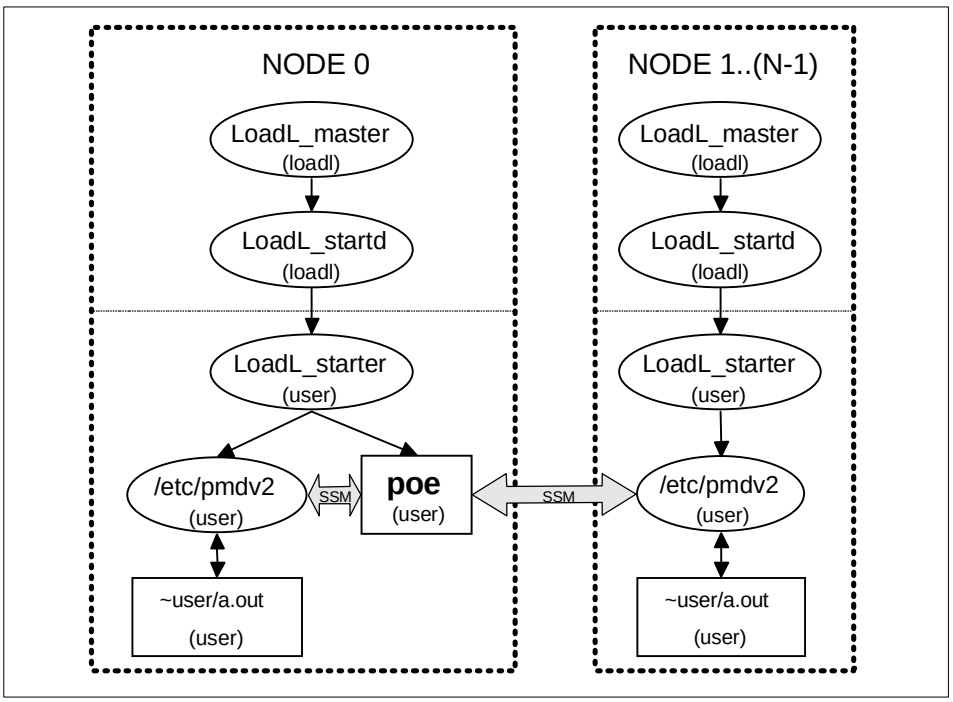


Figure 187. POE Startup under LoadLeveler

When running POE in a batch job, node 0 is a special case. Since there is no poe process already running somewhere, the LoadL_starter process on node 0 spawns two tasks: one which starts the partition manager daemon

and user application, and a second task which represents the home node and runs the poe command. The poe process running on node 0 communicates with the partition manager daemons (on node 0 as well as on the other nodes) through SSM, just as in the interactive case.

16.1.2 Controlling the POE Runtime Environment

The operation of POE can be influenced by more than 40 different environment variables, which are of the form MP_POEOPTION. For many of these options, a corresponding flag for the poe command exists. These flags do not have the “MP_” prefix, and are in lower case. For example, the flag corresponding to MP_POEOPTION would be -poeoption.

POE environment options can be grouped into the following set of functions:

- Partition Manager Control. This is the largest set of POE options. Apart from several timing selections, this group of environment variables also determines how resources (nodes, processors in case of SMP nodes, network adapters) are allocated. See 16.1.4, “POE Resource Management” on page 460 for details.
- Job Specification. MP_CMDFILE specifies a file from which to read node names for allocation. MP_NEWJOB can be used to run multiple job steps on the same set of nodes: if set to yes the partition manager will maintain the partition for future job steps. By using MP_PGMMODEL=mpmd, applications which use the MPMD programming model can be loaded.
- I/O Control. These settings specify how standard I/O which is routed to/from the remote nodes is handled at the home node. The four variables which control this are MP_LABELIO, which can be used to prefix each line of output with the task number where it originated, MP_STDINMODE, MP_HOLD_STDIN, and MP_STDOUTMODE.
- VT Trace Collection. These settings control VT trace generation, discussed in 16.4.2.1, “IBM Visualization Tool (VT)” on page 488. The level of tracing is set by MP_TRACELEVEL, which may take values 0, 1, 2, 3 or 9.
- Generation of Diagnostic Information. The most useful option in this category is MP_INFOLEVEL. It determines the level of message reporting. A value of 0 specifies no messages, 6 specifies maximum messages.
- Message Passing Interface. These environment variables can be used to tune MPI applications. See 16.2.2, “IBM’s Implementation of MPI” on page 466 for details.

See the poe page in *PE for AIX: Operation and Use, Volume 1*, SC28-1979, for a full description of all POE environment variables and command-line flags.

Note

Enforcing Global POE Settings: The /etc/poe.limits file can be used to enforce system-wide values of three POE options. Users cannot override the values in that file by setting the corresponding environment variables to a different value:

- MP_BUFFER_MEM can be used to limit the maximum value to which this variable can be set by users (they can select lower values).
- MP_AUTH=DFS can be used to globally enable DCE credential forwarding. See 16.1.5, “POE Security Integration” on page 461 for details.
- MP_USE_LL=yes causes POE to reject all POE jobs which are not run under LoadLeveler. This can be used to force users to use LoadLeveler.

16.1.3 The POE Compilation Scripts

Parallel applications are compiled by the same compilers which are used for serial programs on regular RS/6000 workstations. The following compilers are supported by Parallel Environment for AIX:

- C for AIX Version 4.3 or later, program number 5765-AAR
- C and C++ Compilers for AIX Version 3.6.4 or later, program number 5801-AAR
- XL Fortran Version 5.1.1 or later, program number 5808-AAR
- XL High Performance Fortran Version 1.3.1 or later, program number 5765-613

Note that C Set++ for AIX is withdrawn from marketing, and the VisualAge C++ Professional for AIX Version 4.0 compiler is not supported by Parallel Environment.

POE provides several compilation scripts which contain all the POE include and library paths, and automatically link the user’s program with the POE and message passing libraries. Four scripts `mpcc`, `mpCC`, `mpxlf` and `mpxlf90` are available which call the `xlc`, `xlC`, and `xlf` compilers. The Fortran compilation scripts can be configured to use XL HPF instead of XL Fortran (if both are installed) by setting the environment variable MP_HPF=yes. These four

scripts also exist in a version which links threaded libraries (suffix _r), and in a version which supports checkpointing (suffix _chkpt). All these scripts do the following:

- Include /usr/lpp/ppe.poe/include
- Link the libraries libmpci.a, libmpi.a, libvtd.a and libppe.a (the Fortran scripts also link libxlf90.a and libxlf.a)
- Link POE's version of libc.a, which has its own exit() routine that allows synchronized profiling and capture of exit code
- Ensure that the POE runtime initialization routine is called before control is passed to the user program

Note

Initializing the POE Runtime: With AIX 4.2 and later, the POE compiler scripts now link the standard AIX /lib/crt0.o, and use the -binitfini:poe_remote_main linker option to initialize the POE runtime environment. With earlier releases of AIX which did not support -binitfini, POE linked a special /usr/lpp/ppe.poe/lib/crt0.o. The call in the executable which actually invokes -binitfini initializations is modinit().

16.1.4 POE Resource Management

The allocation of CPUs and network adapters to parallel jobs is controlled by several variables. The hosts which should execute a parallel program can be specified in a host file listing the machines. The name of that file can be defined by the MP_HOSTFILE variable. This is straightforward, but normally does not balance the load on the system well. Remote nodes can also be determined by LoadLeveler, which should have a better view of the SP's utilization and allocate nodes accordingly. In releases prior to PSSP 3.1, node allocation has been done by the SP Resource Manager (or Job Manager). In PSSP 3.1, these functions have been integrated into LoadLeveler. The MP_RESD and MP_RMPOOL environment variables provide the information if, and from which pool, LoadLeveler (or the SP Resource Manager, in previous releases) should allocate nodes. Exclusive use of CPU resources can be requested by setting MP_CPU_USE=unique.

Through the environment variable MP_EUILIB={ip|us}, you can select if the IP protocol or the User Space (US) protocol should be used. For US communication, the SP Switch adapter (css0) will be used. If IP is specified, the MP_EUIDEVICE environment variable can be set to request a specific network adapter (en0, css0, tr0, and so on). MP_ADAPTER_USE can be set

to shared or dedicated to indicate that the adapter should be exclusively used by the current program. The value of this variable is honored only if LoadLeveler is using to start the job.

The number of tasks can be specified through three environment variables:

- MP_PROCS directly specifies the number of tasks in the parallel program. This is the traditional way to allocate CPUs on uni-processors.
- MP_NODES specifies the number of SP nodes that should be used. SP nodes could be multi-processor SMP nodes, and this option allows you to specify the number of SP nodes rather than the number of processors.
- MP_TASKS_PER_NODE can be used to allocate a fixed number of tasks on all the selected nodes. In conjunction with MP_PROCS or MP_NODES, this value can be used to describe CPU allocation. It is particularly useful on SMP nodes.

Combinations of these three variables can be specified, as long as they do not result in contradictory requirements.

For a more detailed description of the interrelationships of the POE options, refer to Chapter 2, “Executing Parallel Programs” of *Parallel Environment for AIX V2R4 Operation and Use, Volume 1*, SC28-1979.

16.1.5 POE Security Integration

If the SP is integrated into a DCE/DFS or AFS environment, a parallel job needs to establish appropriate user credentials on the remote nodes to be able to access the user’s data (like a DFS or AFS home directory). If POE is run as a LoadLeveler batch job, LoadLeveler will establish these credentials on the batch nodes. For interactive jobs, this has to be done differently.

Chapter 5, “Installation-Related Procedures” of *Parallel Environment for AIX Installation V2R4*, GC28-1981 contains details on how to enable AFS support in Parallel Environment. In particular, /usr/lpp/ppe.poe/samples/afs contains the sample files gettokens.c and settokens.c, which can be linked to the poe command and the pmdv2 daemon to transparently transfer AFS tokens from the home node to the remote nodes.

Unfortunately, this method does not work with DCE credentials because the buffer space used by POE to transfer the credentials to the remote nodes is too small for DCE tickets.

Future releases of Parallel Environment might include facilities to transparently forward DCE credentials in a way similar to the current AFS

solution. In the meantime, the POE command poeauth can be used to explicitly distribute DCE credentials from node 0 (not the home node) to the other remote nodes. To do this, the environment variable MP_AUTH has to be set to indicate that DCE credentials should be forwarded. This is achieved by:

```
export MP_AUTH=DFS
```

There is no command line flag for this option. The default value for this variable is AIX. Note that even with MP_AUTH=DFS, the POE authorization check is still based on the /etc/hosts.equiv or $HOME/.rhosts file.

The poeauth command can then be invoked to copy the DCE credentials, but you have to understand what it does in order to successfully use it:

- Be aware that poeauth expects DCE credentials on node 0. This is not always the case, for example when poe is run from a login node but the nodes allocated for the actual computation come from another pool of nodes. You have to make sure that DCE credentials exist on node 0, either by a suitable hostfile which contains the home node, or by logging in to node 0 before calling poeauth.
- Be aware that poeauth is a normal POE application. POE initializes the user's environment on the remote nodes, which includes a chdir() to the directory in which it was called on the home node. This means that poeauth must not be called when the current directory is in DFS: POE would fail when attempting the chdir() to DFS, before the copying of DCE credentials could take place. You might therefore want to provide a circumvention for this problem, like the following script:

  ```
  #!/bin/ksh
  cd /tmp
  /usr/lpp/ppe.poe/bin/poeauth $@
  cd - >/dev/null
  ```

- Make sure that the parallel application runs on the same nodes to which poeauth has copied the credential files. This can be done by setting the POE environment variable MP_NEWJOB=yes.

If these issues are addressed, poeauth can be used to run interactive POE jobs which have full DCE user credentials on the remote nodes.

Ex.1010 / Page 486 of 572

Note

Order of LPP Installation: If you plan to use Parallel Environment in combination with DCE (and DFS), make sure that the DCE Licensed Program Products (LPPs) are installed before you install Parallel Environment! Parallel Environment has DCE-related parts like the poeauth command for DCE credential forwarding, but the actual installp filesets only contain an object file poeauth.o, no executable. The executable command will be built during installation, but only if the DCE filesets are already installed.

If you install Parallel Environment before DCE, the POE environment will not be completely built for DCE support. The safest solution in such a case is to deinstall POE, and reinstall it after DCE is installed.

16.2 Message Passing Libraries

While POE provides the environment to run programs in parallel, message passing libraries are required if these individual programs need to exchange application data to do their job. The need for message passing libraries is caused by the fact that in a cluster of machines, there is no common address space like in a Symmetric Multiprocessor (SMP), where all processors have shared access to the memory. To access data which resides in the memory of another node, a program needs to call a message passing library procedure which gets that data over the network.

When parallel programming on workstation clusters and parallel computers became popular, many different proprietary message passing systems were developed. The programs written for any one of these systems were normally not portable to other systems. To enhance portability of applications across different systems (and so protect the sometimes huge investments in the development of message passing applications), the Message Passing Interface Forum (MPIF) has (informally) standardized an API for explicit message passing called the Message Passing Interface (MPI). This section describes the MPI industry standard, some details of IBM's implementation of MPI, and a couple of other message passing systems available on the SP.

16.2.1 What is in MPI

The contents of MPI have been defined and published by the Message Passing Interface Forum (MPIF), taking into account the experiences from many different message passing systems. The two relevant documents are:

- *MPI: A Message-Passing Interface Standard* (Version 1.1; June 12, 1995)
- *MPI-2: Extensions to the Message-Passing Interface* (July 18, 1997)

These specifications, and much more information about MPI, can be obtained from the following MPI home pages:

```
http://www.erc.msstate.edu/mpi
http://www.mpi-forum.org
```

MPI Version 1.1 has the following content:

- Point to Point Communication. This includes the basic send and receive calls, in blocking and nonblocking mode. MPI datatypes can be used to describe the contents of messages, including user-defined datatypes which may be used to describe sub-arrays or other data structures.
- Collective Communications. This part contains procedures for barrier synchronization, broadcasts, gather/scatter operations, and reduction operations. MPI provides standard reductions like SUM or MAX, and user-defined reduction operations can be easily added.
- Groups, Contexts, and Communicators. These are used in MPI to describe groups of processes which are in a common communication realm. They provide the infrastructure to develop modular applications (in particular, parallel libraries) which use message passing internally, and guarantee that this communication does not interfere with other messages in the system.
- Process Topologies. Views of a set of processors can be created which suit the application needs. For example, if an algorithm works on a 3D grid, the programmer may prefer to reference the processors also as a 3D grid when sending messages between them. These MPI procedures help to map (physical) processors to (virtual) topologies.
- MPI Environmental Management. This section of the MPI standard has calls to initialize and finalize MPI, procedures to inquire about the environment (number of tasks, local task number), timer routines, and the MPI error handling infrastructure.
- Profiling Interface. MPI defines a simple name-shifting scheme which must be supported by all implementations. This allows the development and use of MPI tools without having access to the library's sources.
- Language bindings. Bindings exist for C and for FORTRAN77.

Note

MPI and Fortran: Unfortunately, the MPI FORTRAN77 bindings do not strictly conform to the ANSI/ISO FORTRAN77 standard. Most importantly, MPI uses one procedure to send user data of all possible datatypes (MPI calls the corresponding procedure argument a *choice* argument). This design is supported with C where data is passed as void*, but the Fortran standard does not allow this (most Fortran compilers do, though). MPI also makes some assumptions about compiler implementation details.

These issues might cause problems when using the MPI FORTRAN77 bindings from Fortran 90 or Fortran 95. Details can be found in MPI-2, and in /usr/lpp/ppe.poe/samples/mpif90/README.mpif90.

MPI Version 2 contains the following extensions to MPI Version 1.1:

- Corrections and Clarifications to MPI Version 1.1. This part of MPI-2 defines the MPI Version 1.2 language level.
- Miscellany. Several additions or modifications which do not fit into one of the other chapters of the MPI-2 document.
- Process Creation and Management. MPI Version 1 does not provide facilities for dynamic process management, like spawning additional processes. MPI-2 provides this functionality.
- One-Sided Communication. Traditional message passing requires a matching receive call on a remote process to send data to it. One-sided communication needs only one visible communication partner. The origin task makes an MPI call, which specifies both origin and target parameters. Target parameters are sent to an asynchronous agent or handler at the remote task and the target activities needed to carry it out. The operation occurs without needing user calls to MPI at the target.
- Extended Collective Operations. Extensions of existing collective operations for intracommunicators to support intercommunicators, some new functions, and support for in-place buffers.
- External Interfaces. Defined to facilitate the development of additions to the core MPI library.
- I/O. This is probably the single most important extension of MPI Version 1. The MPI datatype mechanism can be used to define how parallel processes view a file, and calls similar to the send and receive communication calls can be used to access a file in parallel. There are

three different modes of parallel I/O defined in MPI-2: explicit offset, shared file pointer, and local file pointers.

- Language Bindings. C++ bindings are provided, and some Fortran 90 issues are discussed. As previously mentioned, the MPI Fortran bindings are not completely (Fortran) standard-conforming, so it is difficult to provide a clean binding for Fortran 90 or Fortran 95.

Although the MPI-2 document was published about two years before the time of writing this redbook, there are very few (if any) full implementations of that standard available. This is quite different to MPI Version 1, for which a number of full implementations exist (in the public domain as well as commercial products).

16.2.2 IBM's Implementation of MPI

Parallel Environment 2.4 fully supports MPI Version 1.2. The threads-based version of the MPI library also supports a subset of the MPI-IO chapter of MPI-2. See 16.2.2.3, "Compatibility Issues" on page 470 for details on the signal-based and threads-based MPI libraries, and 16.2.3, "MPI-IO Subset" on page 474 for the MPI-IO subset supported by PE 2.4. IBM has committed to provide full MPI-IO support and expects to continue development of the remainder of the MPI-2 standard in future releases of Parallel Environment for AIX.

In most cases, the details of the IBM implementation of MPI should be transparent. Here we discuss some of the areas which are directly visible to the user.

16.2.2.1 Signal-based and Threads-based MPI

There is one important fact which influences the implementation of MPI: The MPI runtime system has to do a lot of processing in the background (that is, work which is not coupled to a specific MPI procedure invocation). Examples include the handling of nonblocking communication requests, as well as reacting to operating system notifications that messages have arrived through the network. However, the MPI library executes in user mode, not in kernel mode. This means that the MPI runtime system has to share CPU resources with the actual user application. Consequently, there has to be a means to switch between the application (doing calculations) and the MPI runtime (processing communication operations to and from the SP Switch or UDP/IP layers).

Using Signals

Early PE implementations of MPI relied on signals to switch between the application and the POE/MPI runtime. Specifically, there are three groups of signals which are used for different purposes:

- Asynchronous control of the individual parallel processes, especially their termination, is facilitated by POE through the use of the signals SIGQUIT, SIGHUP, SIGPWR, SIGDANGER, SIGTSTP, SIGTERM, SIGHUP and SIGINT.
- The SIGALRM and SIGIO signals are used to manage message traffic. The MPI library has to process communication requests of the application, or deal with data that arrives through the network. These two signals are used to pass control between the application code and the MPI runtime system.
- If User Space (US) communication over the SP Switch is performed, the SIGPIPE signal is used during processing of SP Switch faults. Its main purpose is to defer all MPI processing until the SP Switch has processed the fault and the switch is available again.

Note that with the signal-based MPI library, these signals are not available to the user application; they are used internally by POE/MPI.

This signal-based version of MPI is still provided with PE 2.4 for backward compatibility. It is contained in the libraries libppe.a and libmpi.a. The compiler scripts mpcc, mpCC, mpxlf and mpxlf90 described in 16.1.3, “The POE Compilation Scripts” on page 459 use these signal-based libraries.

Threads-based MPI

The more recent PE implementation of MPI uses threads instead of signals. Threads are normally more efficient than the interrupts which are caused by signals, and they also offer the potential of parallelism if the MPI task runs on an SMP node. The threads-based implementation resides in libppe_r.a and libmpi_r.a. The compiler scripts mpcc_r, mpCC_r, mpxlf_r and mpxlf90_r described in 16.1.3, “The POE Compilation Scripts” on page 459 use these threads-based libraries.

Figure 188 on page 468 shows the control flow in a task which uses the threads-based MPI implementation:

- The Partition Manager Daemon (PMD) uses fork() and exec() to start the POE task. This is the same as for the signal-based implementation.
- Before calling the user's main() program, the POE initialization routine will create a thread called the “POE Asynchronous Control Thread”. This

thread will handle all the asynchronous signals to terminate the process, for example SIGINT.

- The user application itself may create any number of “compute threads”, for example if it exploits thread parallelism by specifying the `-qsmp` compiler option.
- In any of the application's threads, MPI_Init() is called. Note that exactly one thread must call MPI_Init(), not all of them. This will spawn two “service threads”: the Timer Thread (which replaces the SIGALRM signal), and the Asynchronous I/O Thread (which replaces the SIGIO signal).
- All user threads in the process can then use MPI procedures.
- The MPI_Finalize() call must be made from the same thread which called MPI_Init(). This call will terminate all MPI service threads.
- No thread is allowed to call any MPI procedure after this time.
- Any threads created by the application must also be terminated by the application; this is not managed by MPI_Finalize().

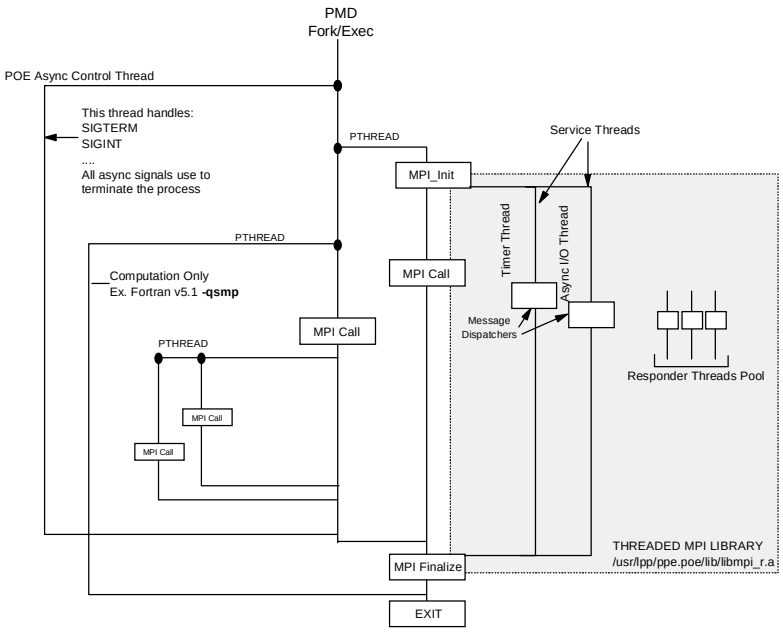


Figure 188. Thread Structure of a MPI-POE Task

The MPI library also creates “responder threads” that handle asynchronous MPI-IO and non-blocking communications. These threads are dynamically

created when needed by an MPI asynchronous operation, but they are not destroyed after the operation ends. This allows a fast reuse of responder threads by future operations. The MPI_Finalize() call will eventually deallocate these resources.

In contrast to the service threads, which are merely a more efficient implementation of the functionality which was already implemented with signals, the responder threads offer significant new capabilities: they can exploit parallelism to process asynchronous communication calls.

The name responder thread derives from the fact that a nonblocking MPCI Irecv can register a handler function. When a message arrives that matches that MPCI Irecv, MPCI responds by putting the registered handler onto the responder queue so it can be executed on a responder thread. Thus, the MPI task “responds” to the arriving message without any application-level participation.

16.2.2.2 The Underlying Communication Subsystem

The IBM implementation of MPI does not directly implement the MPI procedures for a specific networking device. To isolate the MPI library from lower level details, a layered model as shown in Figure 189 on page 470 is used:

- The message-oriented calls of MPI are mapped to an intermediate layer called the Message Passing Client Interface (MPCI). Note that, whereas MPI supports both point-to-point and collective operations, MPCI only supports point-to-point messages. So the mapping of collective communications to point-to-point messages takes place completely within the MPI library.
- MPCI transforms the message-oriented MPI calls to stream-oriented calls, and finally communicates with the device-dependent layer, which is packet oriented.
- On the device-dependent level, two different communication models are supported. Communication can be done over UDP/IP using UNIX sockets, or alternatively the User Space (US) communication protocol of the SP Switch can be used.

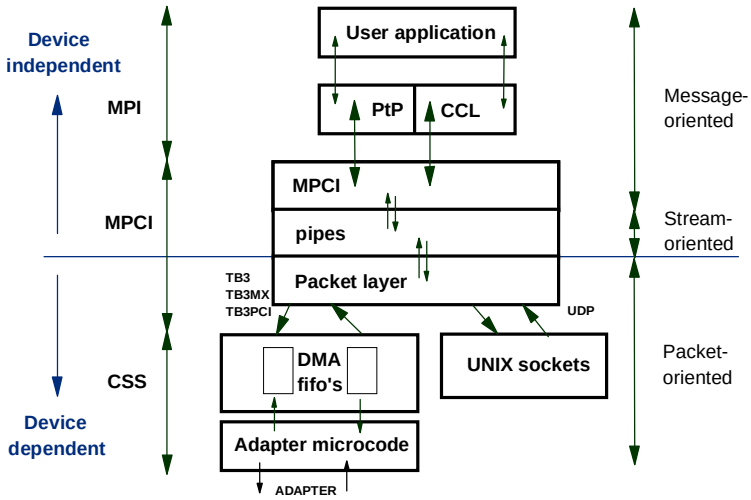


Figure 189. MPI Structure

The MPI library is not aware of the underlying communication subsystem; there is one MPI library regardless of the networking infrastructure. However, different MPCI libraries are available for each communication model, as follows:

```
# lslpp -w "*libmpci*"
  File                                        Fileset              Type
  ----------------------------------------------------------------------------
  /usr/lpp/ssp/css/libip/libmpci_r.a          ssp.css              File
  /usr/lpp/ssp/css/libtb3/libmpci_r.a         ssp.css              File
  /usr/lpp/ssp/css/libip/libmpci.a            ssp.css              File
  /usr/lpp/ssp/css/libtb3/libmpci.a           ssp.css              File
  /usr/lpp/ppe.poe/lib/ip/libmpci.a           ppe.poe              File
  /usr/lpp/ppe.poe/lib/ip/libmpci_r.a         ppe.poe              File
```

The ppe.poe filesets deliver a UDP/IP version of MPCI so Parallel Environment can be used on non-SP workstations. The ssp.css filesets deliver versions for UDP/IP and for US over the SP Switch. The libraries which have the _r suffix are used with the threads-based implementation of MPI.

16.2.2.3 Compatibility Issues

Internals of both AIX and PSSP/PE regularly change from release to release. Normally, binary compatibility is kept between such changes and old applications should run with new software levels without modification.

Nevertheless, there are always some exceptions, and we recommend that you recompile and relink your applications whenever a significant system

upgrade has been performed (for example, moving from AIX 4.2 to AIX 4.3 and from PE 2.3 to PE 2.4).

Since recompilation of the entire application might not be possible or desirable in all cases, this section summarizes two important compatibility issues to which you should pay attention if you use the threaded MPI library.

POSIX Threads Version

Threads for AIX 4.3.0 and earlier are based on Draft 7 of the IEEE POSIX Thread Standard. AIX 4.3.1 and later is based on the IEEE POSIX 1003.1-1996 Thread Standard, which is the final standard, and different from Draft 7.

This means that the AIX threads library (libpthreads.a) has changed in AIX 4.3.1 and later, but AIX maintains binary compatibility across versions. AIX does that by having multiple shared objects in the library itself. Executables built in AIX 4.3.0 or earlier (which link the threads library) reference the shr.o shared objects. Executables compiled in AIX 4.3.1 and later reference shr_xpg5.o shared objects. However, AIX 4.3.1 and later maintains both shared objects in the library, so “old” applications referencing shr.o objects will run without problems. This is shown in Figure 190.

The threaded MPI library in Parallel Environment 2.4 is compiled in AIX 4.3.1, which means it uses the shr_xpg5.o shared objects. Earlier versions of the threaded MPI library reference shr.o objects, since they have been built with AIX 4.2.1 or earlier. Applications compiled with previous versions of AIX and Parallel Environment will run, as long as mutexes (locks) and thread condition structures (signaling structures) are not shared.

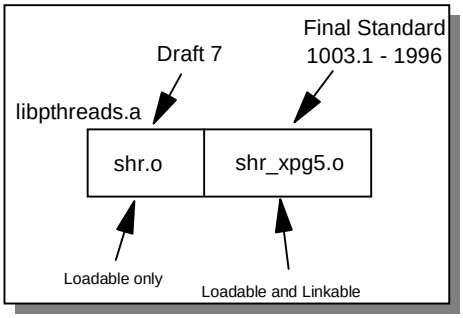


Figure 190. Structure of AIX Thread Library

The interesting case comes when part of an old application has to be recompiled under AIX 4.3.1 and later. In that case, part of the application will

use shr.o shared objects, and part will use shr_xpg5.o shared objects. This is not guaranteed to run, mainly because it is likely that these recompiled parts of the application will share locks or signaling structures with “old” parts; this is not supported across these shared objects (shr.o and shr_xpg5.o).

In order to safely recompile part of an “old” application, Parallel Environment provides a flag (-d7) in the mpcc_r compiler script to link the libpthreads_compat.a library, which contains only the shr.o shared objects.

16.2.2.4 AIX Thread Structure

AIX 4.3.1 and later has changed the default thread structure. In AIX versions prior to AIX 4.3.1, the mapping between user threads (pthreads) and kernel threads (kthreads) is 1:1 (also called *System Contention Scope*). This means that each user thread has a kernel thread allocated to it. The AIX Kernel Dispatcher (sometime called Scheduler) allocates a kernel thread each time a user or process thread is created.

In AIX 4.3.1 and later, this thread structure has changed. The default is now M:N or *Process Contention Scope*, which means that the Kernel Dispatcher has a “pool” of kernel threads which are dynamically allocated to user or process threads when a user scheduling thread switches from one user thread to another, or when a user thread makes blocking system calls.

This new default in AIX brings some problems to multi-threaded parallel applications, because the kernel threads are allocated to process threads only when they are available, which does not guarantee that your application will be able to execute all the threads you (or the MPI runtime library) may want. However, in System Contention Scope, your threads have each allocated a kernel thread no matter what they are doing.

You can change this default to System Contention Scope by setting an environment variable called AIXTHREAD_SCOPE to S. For applications compiled with Parallel Environment 2.4, this variable is not required because the MPI library changes each process scope thread which makes MPI calls into system scope. However, it will not change the scope of a thread which does not call MPI. Applications built with previous versions of Parallel Environment under AIX 4.3.1 or later need to set this variable to S (System Contention Scope), since the previous versions of the MPI library do not change this.

16.2.2.5 Performance Tuning

Similar to POE, the behavior of the MPI runtime and its cooperation with the communication subsystem can be controlled through environment variables. These environment variables are described in Table 13 of *Parallel*

Environment for AIX 2.4: Operation and Use, Vol.1, GC23-3894. The following options have proved to be particularly useful for a broad range of applications:

MP_EAGER_LIMIT — This is a threshold for MPI message sizes. It has a default value of 4 KB and a maximum of 64 KB. MP_EAGER_LIMIT specifies the maximum message size for which the MPI runtime can use a faster internal protocol, because it is expected that the partner of the communication has enough buffer space to receive the messages. Messages above that threshold have to be transmitted using a slower protocol. Consult the Parallel Environment documentation for the interrelation of the number of MPI tasks, MP_EAGER_LIMIT and MP_BUFFER_MEM.

MP_BUFFER_MEM — This can be used to set the size of MPI internal working space. For US communication, this value defaults to its maximum (64 MB), but for IP communication, the default is slightly less than 3 MB. You might want to increase that value if you are sending large messages over IP, or are developing an application in IP mode which should later run with US.

MP_CSS_INTERRUPT — Setting this variable to yes (and maybe adjusting MP_INTRDELAY) may improve the performance for applications which perform asynchronous I/O and have the potential of overlapping communication and computation. With the default setting of MP_CSS_INTERRUPT=no, no SIGIO interrupt will be generated by the runtime if new data has arrived asynchronously and needs to be processed. This implies that the actual processing may be done as a side effect of other MPI calls, or it may be done on the timer threads, or it may not be done until the matching MPI_Wait() call, which causes unnecessary delays. Enabling MP_CSS_INTERRUPT causes additional overhead to process the SIGIO interrupts, but may improve the overall performance because it can significantly reduce the time spent in MPI_Wait() calls.

MP_SINGLE_THREAD — Set this variable to yes to avoid some overhead in a program which is compiled with the threaded MPI library, but is known to be single threaded.
Note: that programs using MPI-IO may not set this to yes.

In any case, you need to experiment with these settings to find out if and how your application will benefit from changing the default values.

Apart from these environment variables, the behavior controlled by MP_INTRDELAY and MP_CSS_INTERRUPT can also be queried and set from the application by using a set of IBM proprietary procedure calls. This might help you to improve performance in cases where the communication

patterns change during the execution of the program, so that a global setting is not suitable. However, we recommend that you do not use these non-standard procedure calls unless significant performance improvements can be achieved.

16.2.3 MPI-IO Subset

The Message Passing Interface (MPI) standard provides an efficient way to communicate and synchronize multiple tasks. However, MPI Version 1 does not provide support for parallel file I/O. Although applications may use a parallel file system to achieve this, the portability of these applications to other platforms and other file systems is not guaranteed.

Chapter 9 of the MPI-2 standard defines the set of MPI calls that allow parallel file I/O. This set of calls is called MPI-IO and a portion is being implemented as part of the threaded MPI library within Parallel Environment 2.4.

The MPI-IO implementation of the MPI-2 standard in Parallel Environment has been divided in two parts. Parallel Environment 2.4 includes most of the MPI-IO functions although not all. The decision as to which part of the subset needed to be implemented first was heavily based on customer requirements. For detailed information about the current implementation of the MPI-IO subset in Parallel Environment 2.4, refer to *IBM Parallel Environment for AIX: MPI Programming and Subroutine Reference*, GC23-3894.

The MPI-IO subset provides great flexibility for applications to define how they will do their I/O. Tasks within an application can use MPI predefined and derived datatypes to partition the single file in multiple views. This allows applications to partition the data and create their own access patterns based on these basic blocks or datatypes.

PE Version 2.4 supports:

- Basic file manipulation (open, close, delete, sync)
- Get/set of file attributes (MPI view, size, group, mode, info)
- Blocking data access with explicit offsets, both independent and collective
- Nonblocking data access with explicit offsets (independent only)
- Derived datatypes for memory and file mapping

This is an initial release of only a subset of MPI-IO. Emphasis in this release is not on performance, but instead focuses on robustness and correctness. This release has no support for file hints which might improve performance. It

Ex.1010 / Page 498 of 572

only uses standard POSIX calls. Support for MPI-IO will be enhanced in future PE releases.

In PE 2.4, MPI-IO is fully supported only for GPFS. Other file systems may be used only when all tasks of the MPI job are on a single node or workstation. (JFS cannot make a single file visible to multiple nodes. NFS, DFS and AFS can make the same file visible to multiple nodes, but do not provide sufficient file consistency guarantees when multiple nodes access a single file.)

16.2.4 Parallel Virtual Machine

Parallel Virtual Machine (PVM) is a popular message passing system which was developed by Oak Ridge National Laboratory (ORNL) to use workstation clusters for parallel computing. It predates the MPI standard, but is still actively used.

Since the release of Parallel Environment Version 2.3, IBM no longer provides its own implementation of PVM. We recommend that you use MPI as the industry standard API for explicit message passing. If PVM has to be used on the SP (for example to port existing PVM applications), the public domain implementation of PVM can be used. PVM Version 3.3 is widely used, and Version 3.4 is the current release as of this writing. There is additional information about this topic at:

`http://www.epm.ornl.gov/pvm/pvm_home.html`

There are two implementations which run on the SP: an RS6K version and an SP2MPI version. Only the SP2MPI implementation supports User Space communication over the SP Switch, as it is layered on top of IBM's MPI. The disadvantage of this PVM implementation is that it requires one extra processor for each invocation of pvm_spawn/pvmfspawn and tasks created by different spawn calls do not intercommunicate via User Space protocol. Therefore, care must be taken when this PVM implementation is used for applications which call pvm_spawn/pvmfspawn. The RS6K implementation does not have this drawback, but can only use IP communication.

16.2.5 Low-Level API (LAPI)

The LAPI library provides uni-lateral communication on the SP Switch in User Space mode. One process initiates a LAPI operation and no complementary action on any other process is required. LAPI provides the LAPI_Put() and LAPI_Get() functions, as well as a general *active message* function LAPI_Amsend(), which allows programmers to register their own handlers.

LAPI is an IBM proprietary calling interface, and only available for the SP switch in User Space mode. Although broadly similar to the one-sided communications of the MPI-2 standard, the semantics and implementation of LAPI are very different from MPI-2 one-sided communication. In addition, LAPI does not support the versatile data layout of MPI datatypes.

Note

MP_MSG_API: If you want to use LAPI, the POE environment variable MP_MSG_API must be set to include LAPI, since LAPI requires a separate DMA window on the SP Switch adapter, different than the adapter window used by MPI.

Additional information can be found in *PSSP for AIX: Administration Guide*, SA22-7348, and in the Chapter 10, “Overview of LAPI”, of the redbook *Technical Presentation for PSSP Version 2.3*, SG24-2080.

16.2.6 Message Passing Library (MPL)

Message Passing Library (MPL) is IBM’s proprietary application programming interface for explicit message passing, which was developed before the MPI standard. Its calling interface is still provided for backwards compatibility, but we strongly recommend that you use MPI for all new application developments. You should also be aware that MPL is not thread-safe, so the MPL calls are only available in the signal-based version of the MPI library.

MPL and MPI could be used in the same program, but both partners of a communication must always be handled by calls to the same library. For example, an MPI_Send() call cannot be received by an MPL mpc_recv() call.

A concise mapping of MPL calls to corresponding MPI calls can be found in Chapter 5, “Babel Fish” of the *Parallel Environment for AIX: Hitchhiker’s Guide*, GC23-3895.

16.3 High Performance Fortran

Although explicit message passing is the most flexible tool to develop parallel applications on a MIMD parallel computer, this programming model is a long step away from the conventional serial programming languages. It requires the whole serial program be recoded into a message passing program to exploit parallelism. For a limited class of problems, the data-parallel programming model offers an alternative: parallelism can be exploited by performing the same (or similar) operations in parallel on large, regular data

Ex.1010 / Page 500 of 572

sets. Data-parallel programming languages are normally based on a sequential programming language, and add directives and new constructs to exploit this parallelism. Several dialects exist for both Fortran and C/C++, but only High Performance Fortran (HPF) is widely implemented and has become a de facto standard. HPF compilers are available for the SP as a separate IBM Licensed Program Product, and also from several independent software vendors.

16.3.1 What is in HPF

High Performance Fortran Version 1 is an extension of Fortran 90 to support the data-parallel programming model on MIMD and SIMD parallel computers using Fortran. HPF retains the usual programming paradigm: a parallel HPF program like a sequential program sees a global address space, and the control flow through the program is (conceptually) identical to that of a normal Fortran program. The most important extensions are:

- Data distribution directives: The distribution of data on the available processors can be explicitly specified, in order to take the different access speeds for data in local and remote memory into account. For example, arrays are aligned to each other, and distributed to a set of (logical) processors, by the ALIGN, DISTRIBUTE and PROCESSORS directives. These directives are comments to a serial Fortran compiler, and also do not change the semantics of the program when interpreted by an HPF compiler.
- New parallel constructs: The FORALL construct, the INDEPENDENT directive on DO loops and FORALLs, new INTRINSICs and an HPF_LIBRARY module extend the possibilities of Fortran 90 to express parallelism. Most of these features are language extensions which will not be understood by a serial Fortran compiler.
- An interface to other programming models: The EXTRINSIC mechanism can be used to define an interface for routines not written in HPF (not in Fortran and/or not data-parallel), which can then be used like ordinary HPF procedures. For example, computational kernels could be written using MPI and then integrated into an HPF main program. To mention one important application of this mechanism: IBM provides HPF interfaces to the IBM Parallel ESSL subroutine library, which is originally coded as a message passing library.

With this additional information, low-level work like the transformation of global to local indices, managing the communication among processors (by message passing, shared memory mechanisms, and so forth) and optimization of communication patterns in the program can be left to the HPF

compiler. Compared to explicit message passing, this significantly speeds up program development: parallelism is both expressed at a higher level, and can be introduced gradually into existing, serial programs. However, achieving good performance depends heavily on the available compiler technology, which is still evolving.

HPF Version 1 defines a subset HPF language, which requires neither full Fortran 90 support nor all the HPF features. Although many HPF compilers now support the full Fortran 90 language, very few support all of the HPF Version 1 features.

The HPF Version 2 language specification is meant to replace the Version 1 specifications, not to add to them (like in MPI, where Version 2 is an addition to Version 1). HPF Version 2 differs from HPF Version 1.1 in the following ways:

- The base language is Fortran 95, not Fortran 90. This means that the FORALL construct has been removed, as it is included in Fortran 95.
- There is no longer a subset HPF. The standard has been reorganized into a core language (similar contents as the previous subset), and a number of approved extensions. These contain features which are not widely used, difficult to implement, or otherwise unlikely to appear in many HPF compilers.
- The extensions which are defined within HPF Version 2.0 include features for more flexible data mapping, data and task parallelism, asynchronous I/O, as well as enhanced functionality of some intrinsic and library procedures and the HPF EXTRINSIC features.

More details can be found in the HPF Language Specifications:

- *HPF Language Specification Version 2.0* (January 31, 1997)
- *HPF Language Specification Version 1.1* (November 10, 1994)

These documents, as well as more information on the HPF language, compilers, applications and tools can be found on the home page of the High Performance Fortran Forum (HPFF) at:

`http://www.crpc.rice.edu/HPFF/`

16.3.2 IBM XL High Performance Fortran

IBM's product to support HPF is the XL HPF Compiler for AIX, program number 5765-613. XL HPF Version 1.4 supports the HPF Version 1.1 subset, full Fortran 90, and most of the full HPF Version 1.1 language features.

Since XL HPF Version 1.4 is based on the XL Fortran Version 6.1 compiler, program number 5765-D78, it also contains the functionality of that compiler. This includes full Fortran 95 and partial OpenMP support, exploitation of the IBM POWER, POWER2, POWER3, and PowerPC architectures (including SMP nodes), and support for 64-bit applications. Be aware that in XL HPF, some of these features may only be available in non-HPF mode, and that neither PSSP 3.1 nor Parallel Environment 2.4 support 64-bit applications.

For more information about XL HPF, check out the following sources:

- `http://www.software.ibm.com/ad/fortran/xlhpf/`
- *IBM XL High Performance Fortran Language Reference and User's Guide*, SC09-2631
- `http://www.software.ibm.com/ad/fortran/xlfortran/`
- *IBM XL Fortran Language Reference*, SC09-2718
- *IBM XL Fortran User's Guide,* SC09-2719

XL HPF requires PSSP and Parallel Environment for AIX as prerequisites, because it generates intermediate serial Fortran code with embedded MPI message passing calls. This is then linked to an executable which contains the HPF, MPI and POE runtimes. The POE infrastructure is used to actually run the HPF program. To POE, there is no difference between an explicit message passing program which has been built using the Parallel Environment's message passing libraries and compilation scripts, and a data-parallel HPF program which has been built using the `xlhpf` command.

XL HPF is integrated with the Visualization Tool (VT) described in 16.4.2.1, "IBM Visualization Tool (VT)" on page 488. This enables source level tracing and profiling of the original HPF program. There is no fully functional HPF source level debugger available for XL HPF.

16.3.3 Portland Group pgHPF

An important independent software vendor's HPF product is pgHPF from the Portland Group, Inc. (PGI). PGI's home page is:

`http://www.pgroup.com`

The European distributor for pgHPF is Pallas GmbH.

`http://www.pallas.de`

The pgHPF compiler is available for many different platforms, including a version for RS/6000 workstation clusters which uses IP communication and a version for the SP which can also exploit User Space (US) communication

over the SP Switch. Using US communication requires Parallel Environment for AIX, including IBM's implementation of MPI.

The `pghpf` compiler actually uses the machine's native Fortran compiler (XL Fortran on AIX platforms), and supplements it with PGI's own libraries and runtime environment. The compiler produces intermediate serial Fortran code with embedded message passing calls. These calls can be layered on top of various message passing libraries. PGI provides RPM by default, which is a stripped-down version of PVM. On the SP, we recommend that you use IBM's implementation of MPI to achieve the highest performance. After building the executable, running it on the SP is comparable to running a plain MPI program.

In addition to the `pghpf` compiler, PGI provides an HPF profiler `pgprof` which supports line-level profiling at the HPF source level. For debugging, the TotalView parallel debugger can be used. TotalView is described in 16.4.1.2, "The TotalView Debugger" on page 484.

16.4 Parallel Programming Tools

Compilers, message passing libraries, and runtime support are the core components which are required to build and run parallel application programs. However, for the development of parallel applications the programmer also needs other tools. Most importantly, parallel debuggers for all the available languages are necessary to allow the programmer to localize programming errors (including message passing errors). While debuggers help to make parallel programs correct, tuning the applications so that they achieve adequate performance requires tools to analyze the runtime characteristics. Serial profiling tools can (and should) be used to tune the code which runs on a single node. But to understand and tune the parallel performance of an application, parallel profiling and tracing must be done. This section summarizes the parallel tools which are available on the SP, either as IBM products or from independent software vendors.

More information about ongoing efforts in the parallel computing community to build parallel programming tools can be found on the home page of the Parallel Tools Consortium:

`http://www.ptools.org/`

16.4.1 Parallel Debuggers

In addition to the serial debuggers available in AIX, IBM provides two parallel debuggers as part of the Parallel Environment for AIX: the text-based `pdbx`

debugger and a debugger with a graphical user interface called pedb (formerly known as xpdbx). One of their main limitations is the fact that they do not support all the C++ and Fortran 90/95 language features which are supported by the IBM compilers. On the other hand, they offer the possibility of attaching to an already running program, which can be very convenient if problems appear in late stages of a long-running program.

As an alternative to these IBM parallel debuggers, the TotalView Multiprocess Debugger from Etnus, Inc. can be used. Its usage is more intuitive, it has much more complete language support, provides superior functionality, and is available on several platforms. For sites with significant parallel programming activities, we recommend that you install and use the TotalView debugger, which is also the debugger selected by the US Department of Energy (DOE) for the ASCI blue project.

16.4.1.1 IBM Parallel Debuggers pdbx and pedb

The pdbx debugger is a text-based POE application, which runs on the POE home node and controls the execution of the standard AIX debugger dbx on each of the remote nodes. It allows grouping of tasks, making it easier to send the normal dbx commands to all or a user-selected set of tasks of the application.

Since pdbx is a source-level debugger, applications should be compiled with the -g option so that sufficient debugging information is available in the executable. In addition, the program's source files must be available on all nodes, not only the executable. If the source files and executables reside in different directories, the `-qfullpath` option should also be used.

The pdbx debugger can operate in two modes: *normal mode,* which starts the application under control of the debugger; and *attach mode,* which permits the attachment of the debugger to an already running POE application. Attach mode is particularly useful for long-running applications which are suspected of hanging in a late stage. If such applications had to be run completely under control of a debugger, it would probably take too much time to reach the interesting state in the calculation. With pdbx, it is possible to attach to the running program at any time.

To debug a parallel program in normal mode, you basically replace the poe command in your program invocation with the pdbx command. The debugger accepts most of the POE environment variables and command-line arguments. For example:

```
pdbx my_prog [prog_options] [poe_options] -l src_dir1 -l src_dir2
```

The -l option (lowercase L) specifies a directory where program source files reside. If multiple directories are needed, multiple -l options must be specified. After the compute nodes have been selected (by the same mechanisms as when calling poe), pdbx starts the dbx debugger on each of the compute nodes. On all the nodes, the application's executable is then run under control of the local dbx debugger, which is controlled by pdbx on the home node.

To attach to an already running POE application, pdbx must be invoked with the -a option, on the same home node as the POE application to which it should be attached. The application is specified to pdbx by the AIX process ID of the poe command running on the home node. For example:

```
$ poe my_prog [prog_options] [poe_options] &  # start application
[1]       31186
$ sleep 300                                   # run it for a while
$ pdbx -a 31186 -l src_dir1 -l src_dir2       # attach to application
```

Since the POE environment is already set up when the application has been started, it is normally not necessary to specify any program or POE options to pdbx when starting it in attach mode.

The pedb debugger, formerly known as xpdbx, provides a Motif-based graphical user interface. Like pdbx, it executes on the home node and can be run in normal mode, or in attach mode by specifying the -a option. In addition to the options accepted by pdbx, the pedb debugger also accepts X windows options to customize its GUI. As for pdbx, the application should be compiled and linked with the -g option, and the source files must be available on all the nodes. An example of a source file view in pedb is shown in Figure 191 on page 483.

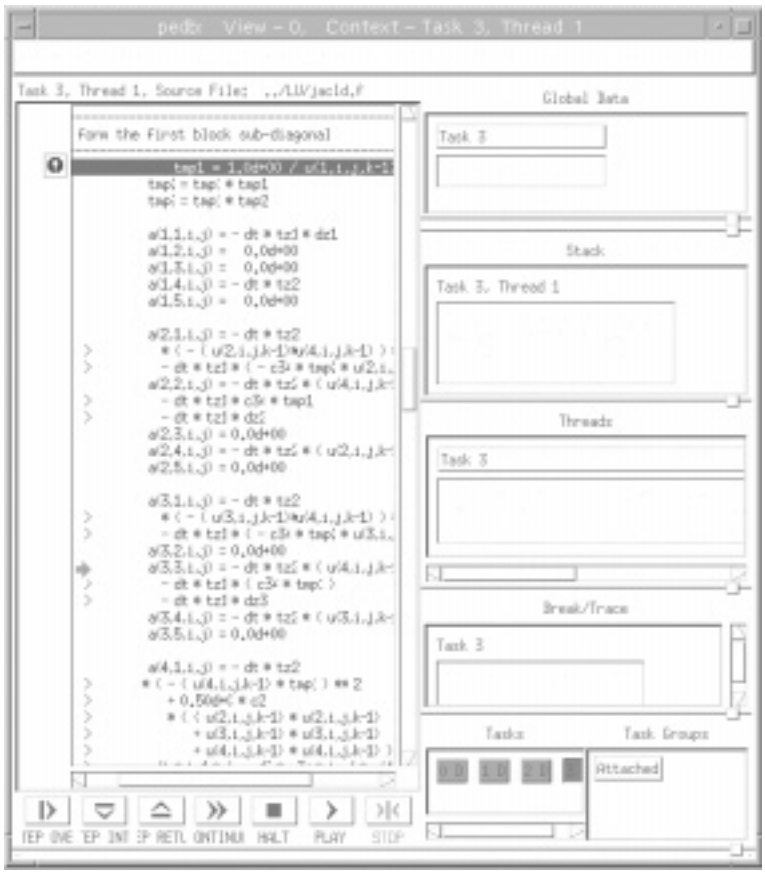


Figure 191. Source File View with pedb

The graphical user interface of pedb makes it more convenient to display data structures in the application. However, there are some limitations concerning the size of the data sets that can be displayed. The pedb debugger also provides a message queue debugger to inspect the MPI messages queued on the nodes.

Note that pedb requires the selection of a fixed set of tasks during startup. This cannot be changed later without detaching from the application and attaching again with a new selection of tasks to be debugged. The tasks that are not selected will not run under debugger control. The maximum number of tasks supported within pedb is 32. For larger applications, only a subset of at most 32 processes can be debugged.

Note

Initial breakpoint: Both pdbx and pedb allow you to set a user-defined initial breakpoint (the place in the program where control is passed to the user for the first time): The POE environment variable MP_DEBUG_INITIAL_STOP can be used to set an alternative file name and source line. By default, the debuggers stop at the first executable statement in the main program.

More details on pdbx and pedb can be found in *IBM Parallel Environment for AIX, Operation and Use, Volume 2, Part 1: Debugging and Visualizing*, SC28-1980.

16.4.1.2 The TotalView Debugger

TotalView is a multiprocess, multithread debugger for applications written in C, C++, FORTRAN 77, Fortran 90, and PGI HPF. It supports multiple parallel programming paradigms including MPI, PVM, and OpenMP. On the SP, TotalView Version 3.8 provides message queue debugging for IBM's implementation of MPI.

The TotalView debugger was formerly available from Dolphin Interconnect Solutions (see `http://www.dolphinics.com/`). It is now marketed by Etnus, Inc. (see `http://www.etnus.com/`). The European distributor for TotalView is Pallas GmbH (see `http://www.pallas.de/`). The Etnus and Pallas home pages also include an on-line presentation of many of the debugger's features.

To use TotalView with an MPI application, the application must be compiled with the `-g` and `-qfullpath` options, and must be linked with the `-bnoobjreorder` option. For example:

```
mpxlf -g -qfullpath -c  myprog.f mysub.f
mpxlf -g -bnoobjreorder myprog.o mysub.o -o myprog
```

The program is then started under TotalView control, for example:

```
totalview -no_stop_all poe -a myprog -procs 4
```

All TotalView options must be specified after the `totalview` and before the poe command. The `-no_stop_all` option specifies that only the task which reaches a breakpoint stops at that breakpoint, by default all other tasks would stop, too. Also note that the `-a` option must be specified between poe and all its arguments.

TotalView then displays its root window, and a *process window* as shown in Figure 192 on page 485. Initially, both windows only show the poe process on the home node.



Figure 192. TotalView Process Window: poe

Type g (lower case g for go) in the poe process window to start the individual tasks of the parallel application. The screen displays progress information for the parallel tasks, and a pop-up which allows you to stop all the tasks before they enter the application's main program. Then a separate process window for the main application's tasks opens, as shown in Figure 193 on page 486. Note that the window title shows the application program's name (heatd2) and the task number (one). The process window for poe can then be closed.

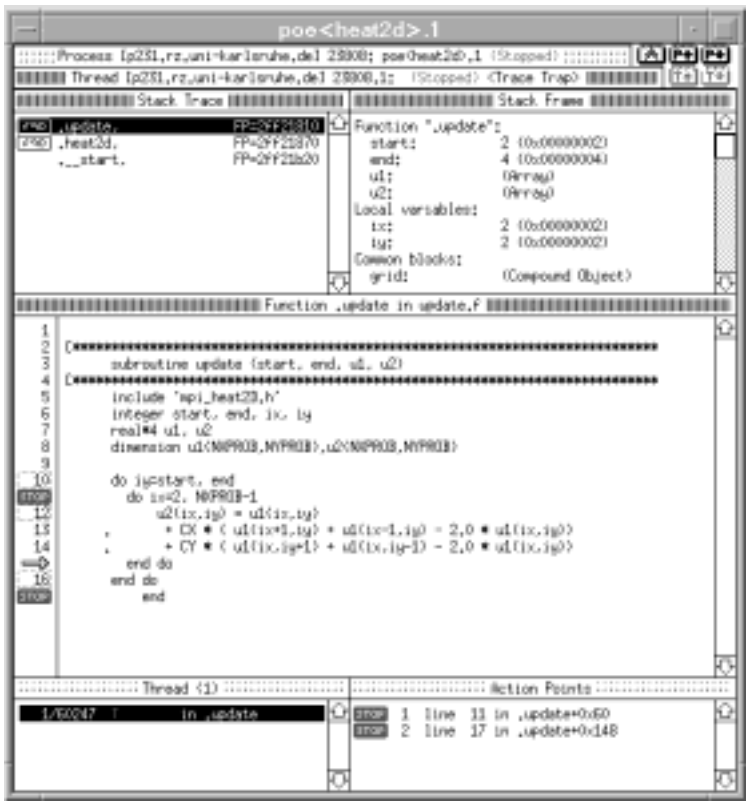


Figure 193. TotalView Process Window: Application Program

The three main areas of the process window are:

- Stack Trace: Displays the current calling hierarchy
- Stack Frame: Displays names and values of (local and global) variables
- Source Code Area: Displays the actual program source

After setting breakpoints, the program can be continued by typing `G` (uppercase G for go group). The root window then displays the status of all tasks, as shown in Figure 194 on page 487. Here `R` means running, `T` means stopped by TotalView, and `B` with a number indicates the breakpoint at which the task is stopped.

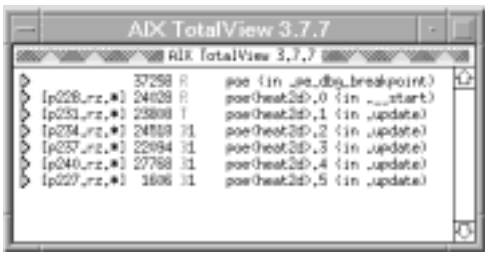


Figure 194. TotalView Root Window: Process Status

Normally, TotalView is controlled using the mouse: the left mouse button selects objects, the middle button displays the currently available functions/commands, and the right button "dives" into the selected object. The right mouse button can be used to display a subroutine's source text, values of variables, or attributes of breakpoints.

To navigate between different tasks, select them in the root window, or use the navigation buttons in the upper right corner of the process window. In general, commands in lower case only apply to one task, whereas upper case commands apply to all tasks of a parallel application.

Note

MP_PULSE: To avoid timeouts during debugging, we recommend that you set the MP_PULSE environment variable to zero before starting TotalView. Otherwise, POE might cancel the application if a task is stopped and does not respond to POE's heartbeat messages.

To use TotalView to debug a serial program, call `totalview` with the name of the executable as argument. For PVM programs, the directory where the TotalView executables reside must be included in the `ep=` clause in the hostfile, and `totalview` must be invoked with the `-pvm` option after the `pvmd` daemon has been started. Debugging PGI HPF programs requires that the program is compiled and linked with the `-g -qfullpath -Mtv -Mmpi` options, and run with the `-tv` runtime option which starts off TotalView.

16.4.2 Profiling and Tracing

For performance analysis and optimization, the programmer needs graphical tools that display the performance information obtained by sample runs of a parallel application. The amount of tracing information (as well as the trace files used to store it) can be overwhelmingly large. The successful

deployment of performance analysis tools depends on presenting this tracing information in an efficient way, and allowing the programmer to quickly navigate and focus on the relevant parts.

Among the tools to do this are IBM's VT and Xprofiler, which are part of the Parallel Environment for AIX. In addition, the Vampir tool from Pallas GmbH is introduced. Vampir has a focus similar to the trace visualization part of VT, but is easier to use and often presents information in a more comprehensive way than VT.

16.4.2.1 IBM Visualization Tool (VT)

Visualization Tool (VT) is IBM's graphical tool for trace visualization of application program execution. Set the POE environment variable MP_TRACELEVEL to a value between 0 (no tracing) and 9 (maximum tracing), to instruct Parallel Environment to generate trace records of a POE application while it is running. This trace data can be analyzed by the `vt` command afterwards. A large number of display options are available, including computation and MPI communication analysis and general system utilization information.

In a second mode of operation, VT is also used for performance monitoring of a set of SP nodes without accessing a particular application's trace files.

More details on VT can be found in *IBM Parallel Environment for AIX, Operation and Use, Volume 2, Part 1: Debugging and Visualizing*, SC28-1980.

16.4.2.2 IBM Xprofiler

AIX supports profiling of CPU usage for serial programs through the standard UNIX `prof` and `gprof` profiling commands. Both `prof` and `gprof` support subprogram level profiling and subprogram call counts. They require that the application be compiled and linked with the `-g` and `-pg` option, respectively. The `gprof` command also provides call graph information.

Parallel Environment takes steps to ensure that each task of a parallel job writes its profile data to a separate file. More details can be found in *IBM Parallel Environment for AIX: Operation and Use, Volume 2*, SC28-1980.

Note

tprof: AIX also provides a `tprof` profiling command. The `tprof` command supports subprogram level profiling without the need to recompile/relink with special options, and line-level profiling for applications which have been compiled and linked with the `-g` option. However, `tprof` provides neither subprogram call counts nor call graph information. There are also some security concerns about using `tprof`: it can be used to profile the AIX kernel, and to do so requires root privileges. If `tprof` is made available to normal users by using SUID root, some security exposures might occur.

The Parallel Environment's Xprofiler is a serial application with a graphical user interface, which provides user-friendly access to `gprof` profiling information for both serial and parallel applications. Like `gprof`, Xprofiler provides only CPU usage information and requires that the application is compiled and linked with the `-pg` option. It is recommended that the `-g` option is also used, since some of the Xprofiler functions also require this flag. Xprofiler is started as follows:

```
xprofiler a.out gmon.out1 gmon.out2 ... gmon.outN [xprof_options]
```

Note that Xprofiler accepts wildcards, so the previous line could be rewritten as follows:

```
xprofiler a.out gmon.out* [xprof_options]
```

As can be seen in this example, for a parallel application all profiling output files must be specified, in addition to the (SPMD) executable.

Note

POE profiled libraries: The AIX operating system and compilers provide a version of their libraries which has been built with the `-pg` compiler options. Typically these are located in /usr/lib/profiled/. When Parallel Environment for AIX is installed, it rebuilds libc.a and stores it in /usr/lpp/ppe.poe/lib/. To support profiling of this modified libc.a, a copy of that library is also built with the `-pg` profiling option, and stored in /usr/lpp/ppe.poe/lib/profiled/. Be sure to include this library in your applications if you want to profile system routines. This can be achieved by setting MP_EUILIBPATH, for example:

```
export MP_EUILIBPATH=/usr/lpp/ppe.poe/lib/profiled:/usr/lib/profiled:/lib/profiled
:/usr/lpp/ppe.poe/lib
```

The Xprofiler can provide the same table reports as `gprof`. In addition, it visualizes the subprogram call tree of the application. Figure 195 on page 490 shows an example of this. Subprograms are represented by boxes whose width and height provide details on the procedure. For example, in Xprofiler's *summary mode*, the height of a subprogram box represents the CPU time spent in itself, while the width also includes the CPU time of its descendants. In *average mode*, the height represents the CPU time in that subprogram averaged over all processes (for which gmon.out files are loaded), while the width gives the standard deviation of that average.

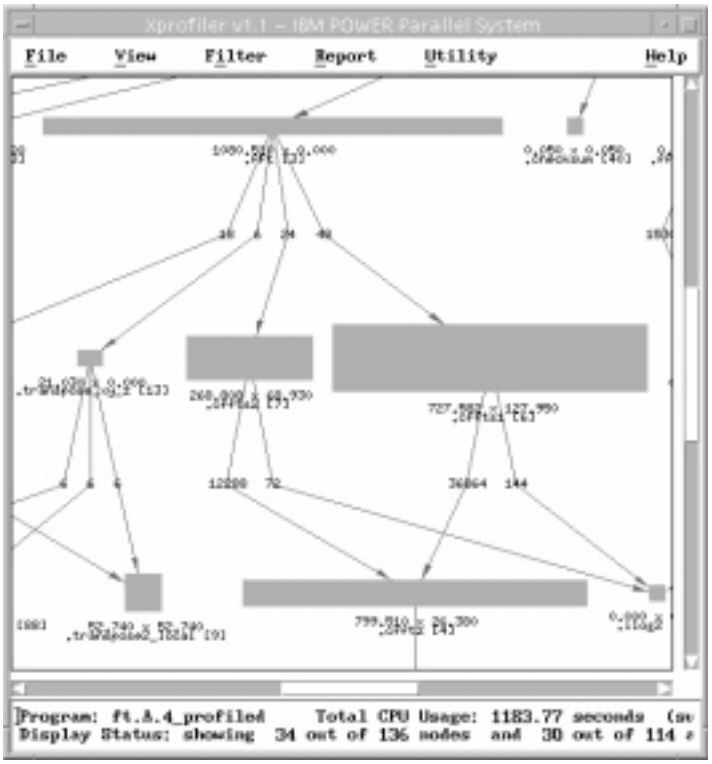


Figure 195. Xprofiler Main Window with Subprogram Call Tree

With Xprofiler, you can easily zoom in and out of interesting parts of the code.

Another important feature is the concept of clustering, which for example allows you to bundle all the subprograms of a library into one box. By default, all the subprograms from a shared library are clustered (or collapsed) into one box. They can be unclustered (or expanded) if details about individual subprograms within that library are required.

More details on Xprofiler can be found in *IBM Parallel Environment for AIX, Operation and Use, Volume 2, Part 2: Profiling*, SC28-1980.

16.4.2.3 Pallas Vampir and VampirTrace

Vampir (Visualization and Analysis of MPI pRograms) is an MPI performance analysis tool. It was initially developed in the German research center Forschungszentrum Juelich (FZJ) and is available as a commercial product from Pallas GmbH:

`http://www.pallas.de`

The tool consists of two parts:

- VampirTrace is a library that collects runtime traces of MPI applications.
- Vampir itself is a graphical visualization tool which can be used to analyze these runtime traces after the program execution.

To instrument an MPI program, it is necessary to link the application with the VampirTrace library. For example, if the library libVT.a resides in /usr/tools/lib/, use the command:

```
mpxlf myprog.f mysub.f -L/usr/tools/lib -lVT -o myprog
```

After setting some environment variables (license manager, configuration file location, and so on) the application can be executed in the usual way, for example by calling:

```
poe myprog -procs 8
```

The VampirTrace library uses the MPI profiling interface to attach to the application, and writes a trace file when the application calls MPI_Finalize. Details can be controlled through a configuration file, or by inserting subroutine calls into the application. For the above example, the default name of the trace file would be myprog.bpv.

The visualization tool Vampir is used to analyze the resulting trace files. It is started by the `vampir` command. Vampir is very easy to use, and has a fast and versatile graphical user interface which allows you to zoom into arbitrary parts of a trace. A variety of graphical displays presents important aspects of the application runtime behavior. Three important classes of displays are:

- Timeline views display application activities and message-passing along the time axis. Figure 196 on page 492 shows an example, zoomed to a very detailed level.

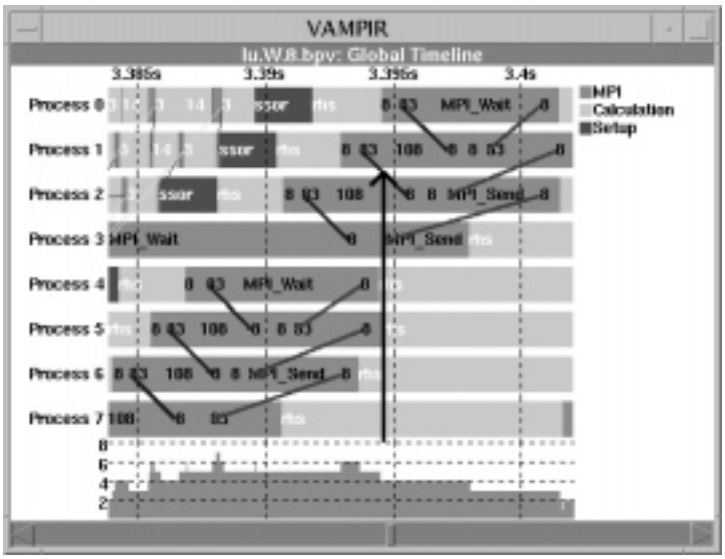


Figure 196. Vampir Global Timeline View

Individual messages can be identified by clicking on them, Figure 197 shows the pop-up window which identifies the message marked by an arrow in Figure 196.



Figure 197. Vampir Message Identification

- Communication statistics present communication metrics for an arbitrary time interval. Figure 198 on page 493 shows a matrix with average message rates between all processes. The striped pattern is typical for nearest-neighbor communication, here for the NAS LU benchmark.

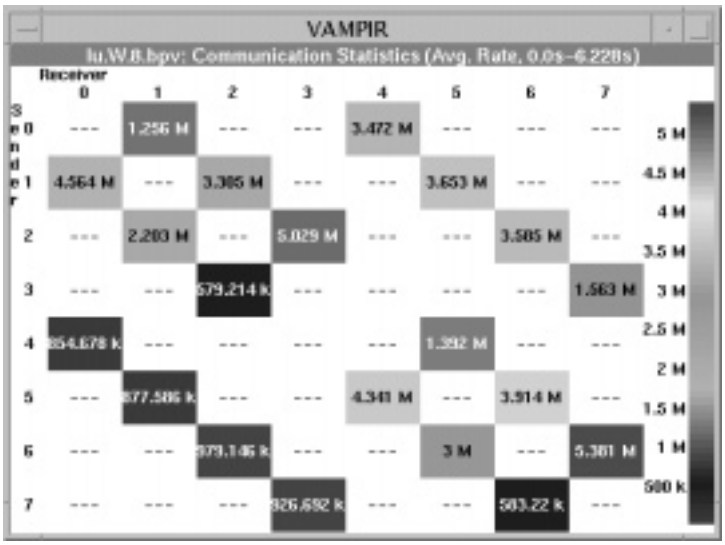


Figure 198. Vampir Average Message Rate Display

Message length distribution histograms are also available, as shown in Figure 199.

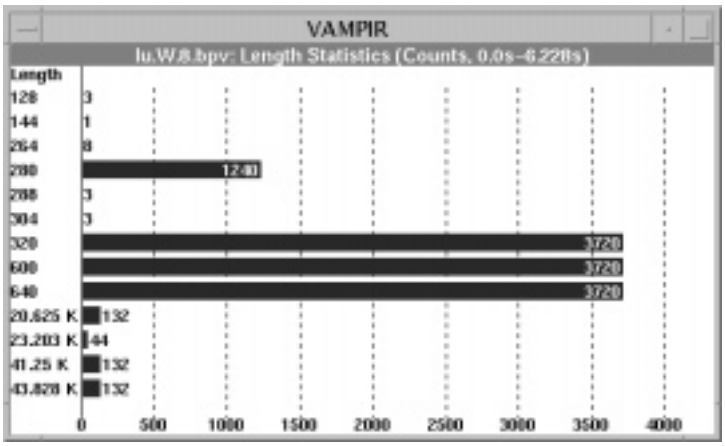


Figure 199. Vampir Message Length Distribution Histogram

- Execution statistics display global (all processes) or local (one process) subroutine execution metrics for an arbitrary time interval. Typical examples are pie charts that present the fraction of time spent in the

application, in message passing, and in profiling as shown in Figure 200. The user can also select specific processes, procedures, and time intervals and display the corresponding activity.

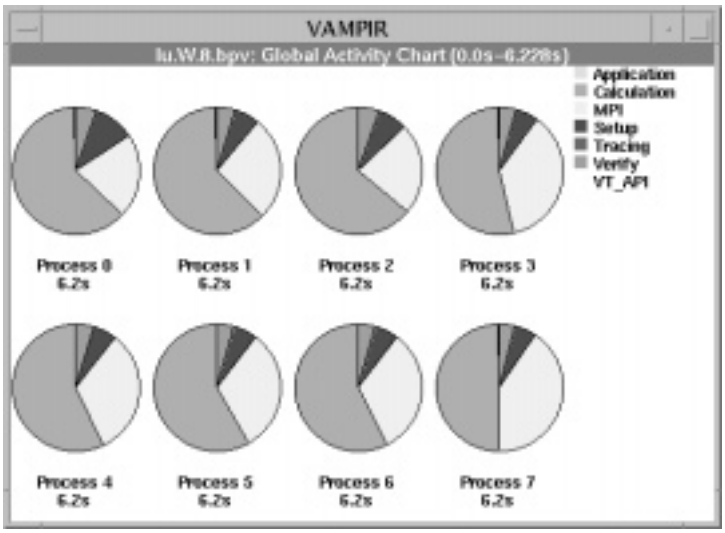


Figure 200. Vampir Global Execution Statistics

Figure 201 on page 495 shows the time spent in selected MPI calls for one of the processes. Call-tree comparison between different program runs is also possible to evaluate the effect of optimizations.

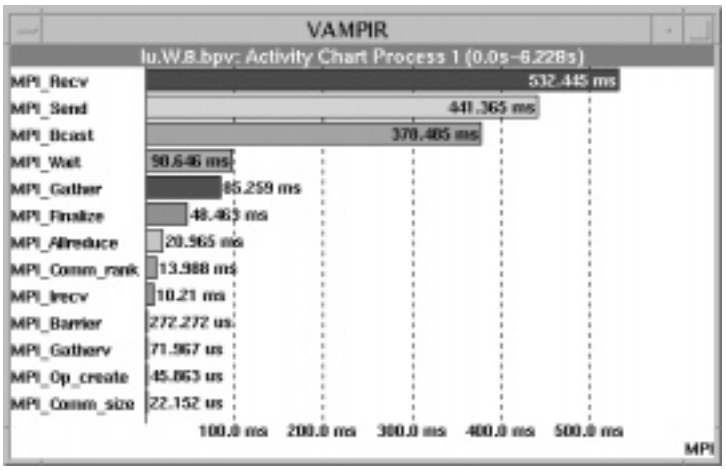


Figure 201. Vampir Local Execution Statistics (User-Selected Routines)

Chapter 17. LoadLeveler

LoadLeveler is a job management system originally developed at the University of Wisconsin. It is offered by IBM to provide the facility for building, submitting and processing batch jobs within a network of machines.

This network of machines may include an SP, other IBM RS/6000s and other types of workstations. The network environment may include Distributed Computing Environment (DCE), AFS, NFS and NQS. All machines in this network that can run LoadLeveler jobs are grouped together and called a cluster.

A batch job consists of a set of job steps. Each job step is an executable that is run on a node or a group of nodes. A job is defined by a Job Command File, which stores the name of the job, the job step(s) involved and other LoadLeveler statements. It is this file that is submitted to LoadLeveler for execution.

A job can either run on one machine (serial) or multiple machines (parallel).

When batch jobs are submitted to LoadLeveler by users, LoadLeveler schedules their execution by matching their requirements with the best available machine resources.

LoadLeveler jobs can be submitted and monitored via commands or the GUI called xloadl.

17.1 Architecture

Once a machine becomes part of a LoadLeveler cluster, it can take on one or more of four roles:

- Central Manager Machine or the Negotiator. A central manager is a node dedicated to examining a submitted job's requirements and finding one or more nodes in the cluster to run the job. There is only one central manager per LoadLeveler cluster; however, it is possible to identify an alternate central manager which becomes the central manager if the primary central manager becomes unavailable.
- Scheduling Machine. When jobs are submitted to LoadLeveler, they get stored in a queue on the scheduling machine. It is this machine's responsibility to contact the central manager and ask it to find an appropriate machine to run the job.
- Executing Machine. This is a node that runs the jobs submitted by users.

© Copyright IBM Corp. 1999

- Submit-only Machine. This type of machine can only submit, query and cancel jobs. A submit-only machine does not carry out any scheduling or execution of jobs. This feature allows machines outside of a LoadLeveler cluster to submit jobs. Submit-only machines have their own GUI which contains a subset of LoadLeveler functions.

Depending on job requirements, a machine in a cluster may take on multiple roles. A machine outside of a cluster can only take part as a submit-only machine.

When a node is part of a LoadLeveler cluster, it may be necessary for the node to stop processing jobs for tasks such as a routine backup. The administrator of a LoadLeveler node therefore can put the node into one of four states:

- Available
- Available only during certain hours
- Available when the keyboard and mouse are not being used interactively
- Not available

There are six daemons that are used by LoadLeveler to process jobs:

- **master** — This daemon runs on every node in a LoadLeveler cluster. It does not, however, run on any submit-only machines. It manages all other LoadLeveler daemons running on the node.
- **schedd** — This daemon receives the submitted jobs and schedules them for execution. The scheduling is based on the machines selected by the negotiator daemon (discussed later in this section) on the central manager. The schedd is started, restarted, signalled and stopped by the master daemon.
- **startd** — This daemon monitors jobs and machine resources on all executing machines in the cluster. It communicates the machine availability to the central manager, and receives a job to be executed from the schedd.
- **starter** — This is spawned by the startd daemon when startd receives the job from schedd. The starter daemon is responsible for running the jobs and reporting the status back to startd.
- **negotiator** — This daemon runs on the central manager and records information regarding the availability of all executing machines to perform jobs. It receives job information from schedd and decides which nodes is are to perform the job. Once a decision has been made, it sends this information to schedd and lets schedd contact the appropriate executing machines.

Ex.1010 / Page 522 of 572

- **kbdd** — This daemon monitors the keyboard and mouse activity on a node. It does not work by using interrupts. Rather, it sleeps for a defined period of time and then determines if there have been any keyboard or mouse activities during the time it slept. If so, it sends this information to startd.

Figure 202 on page 500 shows the steps LoadLeveler uses to handle a job. They are summarized as follows:

1. A job is first submitted by a user using either the GUI or the `llsubmit` command to the schedd daemon on the scheduling machine. The submitting machine can be a submit-only machine or any other machine in the cluster.
2. The schedd daemon on the scheduling machine receives the job and places it into the job queue.
3. The schedd daemon contacts the negotiator daemon on the central manager to inform it that a job has been placed in the queue. The schedd daemon also sends the job description information to the negotiator daemon.
4. The negotiator daemon, which receives updates from the startd daemon on all executing machines, checks for an available executing machine with resources which match the job's requirements. Once found, the negotiator daemon sends a "permit to run" signal to the schedd daemon.
5. The schedd daemon dispatches the job to the startd daemon on the executing machine.
6. The startd daemon spawns the starter daemon which handles the job. When the job is finished, the starter daemon sends a signal back to the startd daemon.
7. The startd daemon sends a signal back to the schedd daemon informing it that the job has been completed.
8. The schedd daemon updates the negotiator daemon that the job has finished.

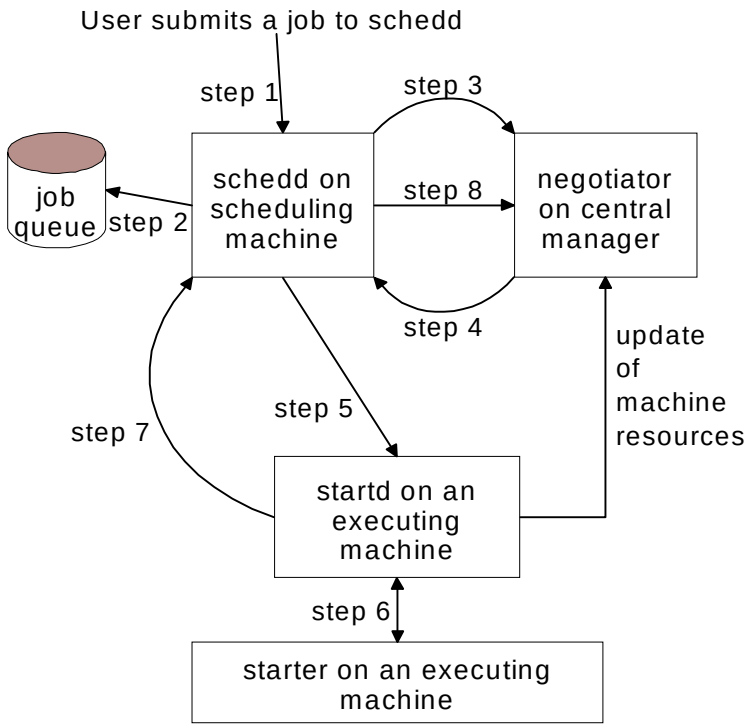


Figure 202. Summary of How LoadLeveler Executes a Job

Note that jobs are not necessarily handled in a first-come, first-served manner. In the job requirements, there are two factors which can influence the scheduling of a job: its class and priority.

A job's class is the user-defined category to which a job belongs, and it is possible to define a specific period when a job may be run. For example, you may want to clean up the contents of a particular file system on a regular basis. This job, however, can only run for a short period of time to minimize outage. You can therefore define a class called small_job, with the restriction that any job belonging to this class can only run for 15 minutes on a particular CPU. Classes are defined in the /home/loadl/LoadL_admin file. An example of a class stanza in this file is as follows:

```
small_job:          type = class              # class for small jobs
                    priority = 100            # ClassSysprio
                    cpu_limit = 15:00      # Limit of 15 minutes
```

The second factor that influences job scheduling is job priority. Jobs' priorities determine which job runs ahead of other jobs. LoadLeveler looks at two priorities: the value the user assigns to the job (user priority), and the value LoadLeveler calculates (SYSPRIO).

The user priority is specified in the job command file. It is a number between 0 and 100, inclusive. The higher the number assigned, the greater the priority given to the job.

LoadLeveler calculates its priority for a job by using a formula defined in its configuration file to come up with the SYSPRIO. The system administrator sets up the formula, which can be based on a number of factors. Details on how to set up the SYSPRIO formula can be found in *IBM LoadLeveler for AIX Using and Administering Version 2 Release 1*, SA22-7311.

LoadLeveler schedules a job based on an adjusted system priority, a number that factors in both the user priority and the SYSPRIO. When jobs are submitted, LoadLeveler does the following:

1. Assigns a SYSPRIO
2. Orders the jobs by SYSPRIO
3. Assigns an adjusted system priority to jobs belonging to the same user and the same class, by taking all jobs with the same SYSPRIO and ordering them according to the user priority

17.2 Configuration of LoadLeveler

The installation and configuration of LoadLeveler is summarized in 10.3, "Installing and Configuring LoadLeveler", in *PSSP 3.1 Announcement*, SG24-5332.

The installation procedure is described in detail in *Installation Memo for IBM LoadLeveler Version 2 Release 1*, GI10-0642.

After installing LoadLeveler, you need to configure it for your system. Global configuration information includes the following:

- LoadLeveler user ID and group ID (default user ID is loadl)
- The configuration directory (default is /home/loadl)
- The global configuration file (default is LoadL_config)

There are three major steps to configure LoadLeveler:

1. Set up the administration file, which lists and defines the machines in a LoadLeveler cluster. The default name of this file is

/home/loadl/LoadL_admin, which assumes that you have given the user id loadl for LoadLeveler.

2. Set up the global configuration file that contains the parameters controlling how LoadLeveler operates in the cluster.
3. Set up local configuration files on machines to define settings for specific nodes. The default file is /home/loadl/<hostname_of_node>/LoadL_config.local.

Configuration of LoadLeveler can be complex. The procedure is discussed in depth in *IBM LoadLeveler for AIX Using and Administering Version 2 Release 1*, SA22-7311.

Once LoadLeveler has been installed and configured, the command to start it is `/usr/lpp/LoadL/full/bin/llctl -g start`. This starts LoadLeveler in all the machines defined to be part of the LoadLeveler cluster.

To check on the status of all the machines in a LoadLeveler cluster, run `/usr/lpp/LoadL/full/bin/llstatus`.

17.3 Serial Jobs

Once LoadLeveler is configured and started, you can begin to build and submit jobs to be run. There are two ways to build the jobs: write your own job command file or use the xloadl GUI. By default, the name of a job command file ends in .cmd.

The following is an example of a job command file; it is called errpt.cmd:

```
#!/bin/ksh
# @ job_type = serial
# @ executable = /usr/bin/errpt
# @ arguments = -a
# @ output = $(Executable).$(Cluster).$(Process).out
# @ error = $(Executable).$(Cluster).$(Process).err
# @ initialdir = /usr/lpp/LoadL/full/bin
# @ notify_user = loadl@sp4en0.msc.itso.ibm.com
# @ notification = always
# @ checkpoint = no
# @ restart = no
# @ requirements = (Arch == "R6000") && (OpSys == "AIX43")
# @ queue
```

All lines beginning with # are treated as comments. LoadLeveler commands begin with #@. The word following the #@ is the LoadLeveler keyword which describes the job to be done. The LoadLeveler keyword is case insensitive.

Ex.1010 / Page 526 of 572

A job command file consists of a number of job steps, or commands to be executed. LoadLeveler stores a job by placing each job step into the queue managed by the schedd daemon to await execution. The keyword that lets LoadLeveler know when a job step has been defined is queue. Each job command file must contain at least one job step (or one queue keyword).

In this example, we ask each node in the cluster to run the command `errpt -a` and store it in a unique filename as defined by the output keyword.

The job_type keyword is optional and defines the type of job, serial or parallel, that we are running.

The executable keyword describes the command that you want to run. This can be a system command or a script you wrote. If the executable keyword is not used, then the job command file itself is treated as the executable. You need to then specify the commands as though the job command file, that is, without a # or a #@.

The arguments keyword specifies the flags that the executable requires.

The output and error keywords define the files to which LoadLeveler will write the respective output and error messages.

The initialdir keyword specifies the path name of the directory to be used as the initial directory during the execution of the job step. The directory specified must exist on both the submitting machine and the machine where the job runs.

The notification and notify_user keywords specify whether LoadLeveler is to send any messages and to which user the messages are to be sent.

The checkpoint keyword specifies whether you want to use checkpointing or not. Checkpointing means that the state of a job is periodically saved for recovery purposes should the node go down. It can be specified via the application program (user initiated) or taken at intervals specified in the LoadLeveler configuration file. There are a number of issues to consider when checkpointing is used. Refer to 10.1.3, "Checkpointing", in *PSSP 3.1 Announcement*, SG24-5332 for a summary and *IBM LoadLeveler for AIX Using and Administering Version 2 Release 1*, SA22-7311 for details.

The restart keyword specifies whether the central manager is to requeue the job should the node go down and come back up. This is different than a restart using checkpoint because this restarts the whole job, whereas a checkpoint specifies a particular point in the program.

The requirements keyword specifies one or a number of requirements which a machine in the LoadLeveler cluster must meet in order to execute the job step.

Once the job command file is built, it must be submitted for LoadLeveler to place on the queue. The command to execute this is `llsubmit`. For our previous example, run `llsubmit errpt.cmd` to submit the job to LoadLeveler. If the command is successful, it returns the message:

```
llsubmit: The job "sp4n05.msc.itso.ibm.com.1" has been submitted.
```

The job name given is the name of the node where the job is submitted, and a job ID that LoadLeveler assigns. In this case, the job ID is 1.

The preceding is just a simple example of a job command file for a serial job. There are many other ways to construct a job command file. For a detailed explanation of the possible structures (including examples) and all the keywords that can be specified in a job command file, refer to *IBM LoadLeveler for AIX Using and Administering Version 2 Release 1*, SA22-7311.

To use the GUI xloadl to handle the chores of building and submitting jobs, run `xloadl` from the command line. This starts the GUI shown in Figure 203 on page 505.

Ex.1010 / Page 528 of 572

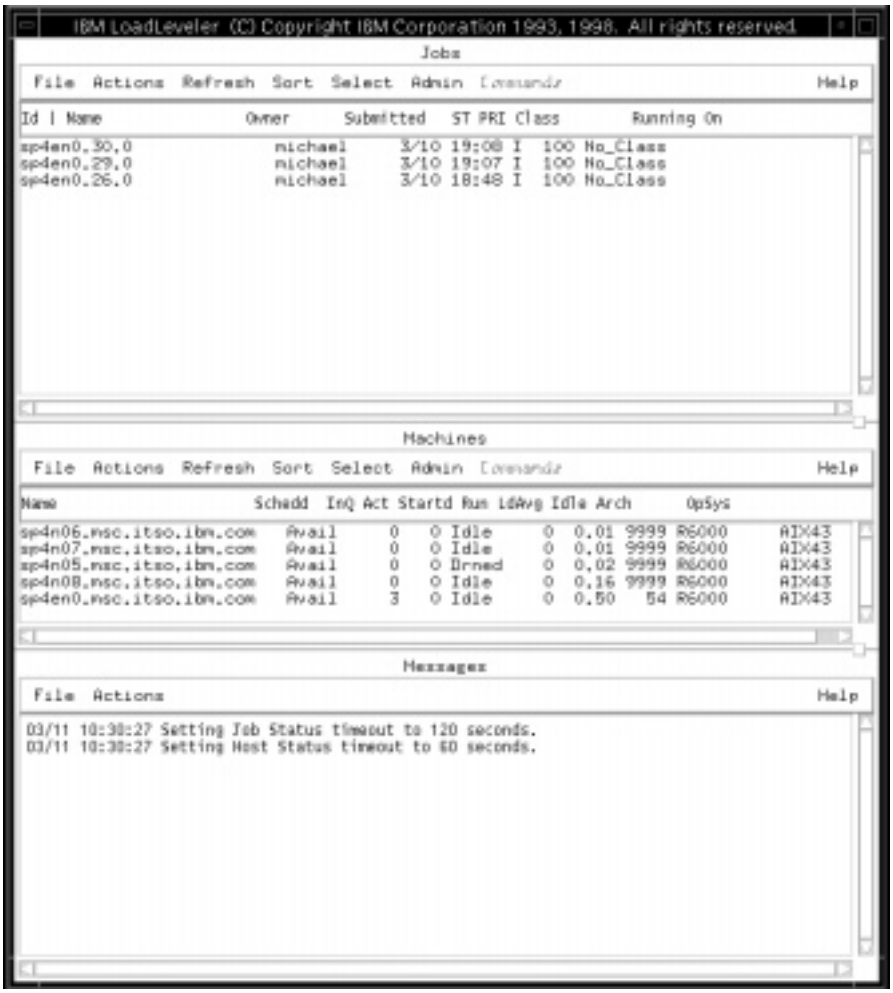


Figure 203. The xloadl GUI

To build a serial job, select **File**, then **Build_File_Job** and finally **File_Build_Job_Serial**. This brings up the dialog box where you can fill in the values for all keywords available in a job command file for serial jobs. Figure 205 on page 507 shows this dialog box.

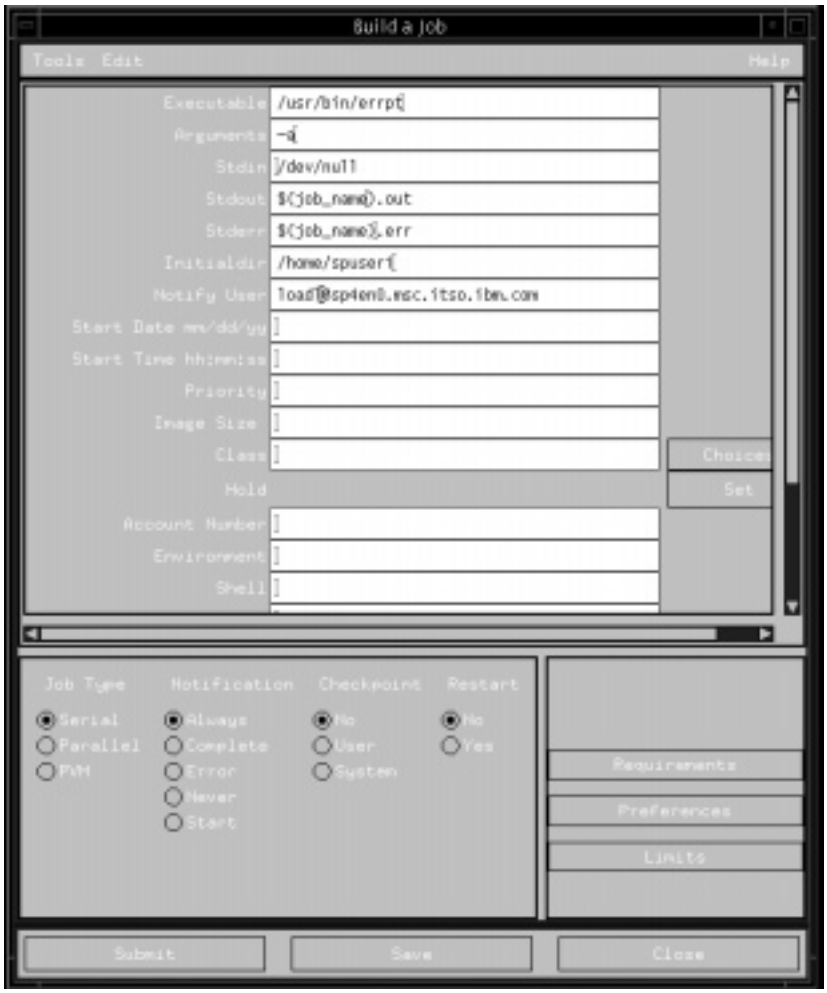


Figure 204. Dialog Box for Writing a Job Command File

Once the file has been built, you have the option to submit the file right away or save it first and submit it later. To submit it immediately, click on the **Submit** button. To save the file first, click on the **Save** button and specify the file name in the dialog box that pops up.

To submit a job command file that has already been built, choose from the main xloadl GUI **File**, **File_Submit_Job** and select the job command file you want to submit in the associated dialog box. This dialog box is shown in Figure 205 on page 507.



Figure 205. Dialog Box for Submitting a Job

After a job command file has been submitted for execution, you can run the command `/usr/lpp/LoadL/full/bin/llq` to check on its status within a queue. The following is a sample output from running `llq`:

```
$ llq
Id                       Owner      Submitted   ST PRI Class        Running On
------------------------ ---------- ----------- -- --- ------------ -----------
sp4en0.21.0              spuser1     3/10 13:02 ST 100 No_Class     sp4n06

1 job steps in queue, 0 waiting, 1 pending, 0 running, 0 held
```

17.4 Parallel Jobs

LoadLeveler allows you to submit parallel jobs that have been written using the following:

- IBM Parallel Environment Library (POE/MPI/LAPI) 2.4.0
- Parallel Virtual Machine (PVM) 3.3 (RS6K architecture)
- Parallel Virtual Machine (PVM) 3.3.11+ (SP2MPI architecture)

After choosing the method to submit parallel jobs, install the appropriate software. Information on parallel programming is found in Chapter 16, “Parallel Programming” on page 453.

You have to make some changes to the LoadLeveler configuration to enable it to handle parallel jobs.

For POE jobs, ensure that all the adapters available to be used on each node have been identified to LoadLeveler. To do this, run `llextSDR` to get the node and adapter information out of the SDR. The following example shows what these entries look like for a node with the hostname sp4n05, one Ethernet adapter and one SP switch adapter:

```
sp4n05:
type = machine
adapter_stanzas = sp4sw05.msc.itso.ibm.com sp4n05.msc.itso.ibm.com
spacct_excluse_enable = false
alias = sp4sw05.msc.itso.ibm.com sp4n05.msc.itso.ibm.com

sp4sw05.msc.itso.ibm.com:
type = adapter
adapter_name = css0
network_type = switch
interface_address = 192.168.14.5
interface_name = sp4sw05.msc.itso.ibm.com
switch_node_number = 4

sp4n05.msc.itso.ibm.com:
type = adapter
adapter_name = en0
network_type = ethernet
interface_address = 192.168.4.5
interface_name = sp4n05.msc.itso.ibm.com
```

Create two adapter stanzas in the LoadL_admin file, one for each adapter, and then add an adapter_stanzas line to the machine stanza. The following example shows these entries:

```
sp4n05: type = machine
        central_manager = false
        adapter_stanzas = sp4n05_en0 sp4n05_css0
        spacct_excluse_enable = false

sp4n05_en0:     type = adapter
                adapter_name = en0
                interface_address = 192.168.4.5
                interface_name = sp4n05.msc.itso.ibm.com
                network_type = ethernet

sp4n05_css0:    type = adapter
                adapter_name = css0
                interface_address = 192.168.14.5
                interface_name = sp4sw05.msc.itso.ibm.com
                network_type = switch
                switch_node_number = 4
```

After adding these entries, ensure that you have selected an appropriate scheduler for your job (for information on scheduling, refer to 17.5, “Scheduling” on page 511).

At this point, you can consider whether you want to set up specific classes in the LoadL_admin file to describe POE job characteristics. Other optional configurable functions include grouping nodes into pools, restricting nodes to handle particular types of jobs (batch, interactive or both), and turning on SP exclusive use accounting.

Once these have been considered and added to the LoadL_admin file, you are ready to start LoadLeveler.

For PVM 3.3 RS6K architecture jobs, you need to set up a path for LoadLeveler to find the PVM software. LoadLeveler by default expects that PVM is installed in ~loadl/pvm3. If the software is installed elsewhere, include this in the machine stanza in the LoadL_admin file. For example, if you have PVM installed in /home/loadl/aix43/pvm3, include the following line in the machine stanza:

```
pvm_root = /home/loadl/aix43/pvm3
```

This entry sets the environment variable $PVM_ROOT, required by PVM.

If you are running PVM 3.3.11+, LoadLeveler does not expect the software to be installed in ~loadl/pvm3. Rather, LoadLeveler expects PVM to be installed in a directory that is accessible to and executable by all nodes in the LoadLeveler cluster.

Both versions of PVM have a restriction that limits each user to only run one instance of PVM on each node. You can ensure that LoadLeveler does not start more than one PVM job per node by setting up a class for PVM jobs. For more information on this topic, refer to Chapter 6, “Administration Tasks for Parallel Jobs”, in *IBM LoadLeveler for AIX Using and Administering Version 2 Release 1*, SA22-7311.

Once LoadLeveler is properly set up, it is possible to build and submit jobs either by the command line or the `xloadl` GUI.

The GUI for building and submitting parallel jobs is the same as the one for serial jobs. The difference is the selection in the **Job Type** field under the **Build a Job** dialog box. You have your choice of either **Serial**, **Parallel** or **PVM**.

Here is an example of a job command file for a parallel job (using POE):

```
# @ job_type = parallel
# @ output = poe_job.out
# @ error = poe_job.err
# @ node = 4, 10
# @ tasks_per_node = 2
# @ network.LAPI = switch,shared,US
# @ network.MPI = switch,shared,US
# @ wall_clock_limit = 1:30, 1:10
# @ executable = /home/spuser1/poe_job
# @ arguments = /poe_parameters -euilib "share"
# @ class = POE
# @ queue
```

We can take a closer look at this job command file.

The job_type obviously must be set to parallel.

The output and error entries remain the same as in the case of a serial job.

The node keyword can be used to specify a minimum and maximum number of nodes that this job can use. The format is node = min, max where min is the minimum number of nodes this job requires and max is the maximum number of nodes required. In this case, the values of 4 and 10 indicate that the job requires at least 4 nodes but no more than 10.

Since the backfill scheduler available with LoadLeveler 2.1 supports multiple tasks being scheduled on a node, you can specify how many jobs are to be scheduled per node. This is done using the tasks_per_node keyword, which is used in conjunction with the node keyword.

The network keyword specifies the communication protocols, adapters and their characteristics to be used for the parallel job. In this instance, we define two protocols, *LAPI* and *MPI*. For each, we are to use the network type of *switch*. The *shared* option means that the adapters can be shared with tasks from other job steps, and *US* refer to the mode of the communication subsystem. (US means User Space; the other option is Internet Protocol, specified by IP.)

The wall_clock_limit sets the time limit for running a job. It is required, and is set either in the job command file by the user or in the class configuration in the LoadL_admin file by the administration when a backfill scheduling algorithm is used. The two numbers for the wall_clock_limit in this example specify a hard limit of one hour and thirty minutes and a soft limit of one hour and ten minutes. The hard limit is the absolute maximum while the soft limit is an upper bound that may be crossed for a short period of time.

The executable and arguments keywords are used just like in a serial file to specify the job to run.

The class is specified to be POE in this example.

The queue keyword is once again required to let the system know when a job step's definitions have been completed and that the schedd can place this job step on the queue to be dispatched by LoadLeveler.

The `llsubmit` command is used to submit the job command file to LoadLeveler. Once submitted, you can view its status either in the GUI or through the use of the `llq` command.

One major change that has been implemented in LoadLeveler 2.1 is the incorporation of certain Resource Manager functions into the program to run parallel batch jobs, since Resource Manager is no longer offered in PSSP 3.1. LoadLeveler 2.1, for example, is able to load and unload the Job Switch Resource Table (JRST) itself using a switch table API. LoadLeveler can also provide the JRST to other programs, such as POE. The JRST is discussed in greater detail in Chapter 11, "Parallel Environment 2.4" in *PSSP 3.1 Announcement*, SG24-5332, while the switch table API is detailed in *IBM Parallel System Support Program for AIX Command and Technical Reference, Version 3 Release 1*, SA22-7351.

17.5 Scheduling

Once jobs have been defined and submitted to LoadLeveler, they are scheduled for execution. LoadLeveler can schedule jobs using one of three possible algorithms: the default LoadLeveler scheduler, the backfill scheduler

and the job control API. The scheduling method is set by manipulating two keywords within the global configuration file: SCHEDULER_API and SCHEDULER_TYPE. Recall from section 17.2, “Configuration of LoadLeveler” on page 501, that the system default name for the global configuration file is /home/loadl/LoadL_config.

The default LoadLeveler scheduler is meant primarily for serial jobs, although it can handle parallel jobs. This algorithm schedules jobs according to a machine’s MACHPRIO. The higher the MACHPRIO, the more available a machine is to handle a job.

The MACHPRIO keyword can set to an expression defined by the system administrator and used by the central manager to measure executing machines’ ability to handle jobs. It is based on a combination of eight factors:

- LoadAvg - the one-minute load average of the machine
- Cpus - the number of processors in the machine
- Speed - the relative speed of the machine as defined in the LoadL_admin file
- Memory - the size of real memory (in MB) on the machine
- VirtualMemory - the size of the available swap space on the machine in KB
- Disk - the amount of free space left in the file system where the executables reside in KB
- CustomMetric - allows you to set a relative priority number for one or more machines
- MasterMachPriority - a value of 1 for this keyword specifies a machine as a master node for a parallel job

The MACHPRIO can only be set in the global configuration file or the configuration file local on the central manager.

The default LoadLeveler scheduler continues to monitor a machine after it has been assigned a job to ensure that the workload is not too heavy. If the workload is deemed to be too heavy for the node, the job may be suspended and resumed at a later time. If this method is used to handle parallel jobs, it uses a reservation method to accumulate the required number of nodes before dispatching the job. The problem with this method is that if a machine is reserved, it cannot handle any other jobs. If a job requires a large number of nodes, it is possible to waste valuable resources because nodes that otherwise can be used to process small jobs are held in reserve for the large job. It is also possible that a large job with a low priority will never get

Ex.1010 / Page 536 of 572

dispatched because LoadLeveler is unable to accumulate the sufficient number of nodes.

The default LoadLeveler scheduler is set by including this expression in the global configuration file:

```
SCHEDULER_API = NO
```

Do not include the SCHEDULER_TYPE keyword because it overwrites the SCHEDULER_API expression. Simply set SCHEDULER_API to NO.

The backfill scheduler can be used for both serial and parallel jobs, although it is designed to handle primarily parallel jobs. The backfill schedule requires that every job sent to schedd has the wall_clock_limit set in its job command file. This defines the maximum amount of time in which the job will complete. Using this information, the backfill algorithm works to ensure that the highest priority jobs are not delayed.

The backfill scheduler supports the scheduling of multiple tasks per machine and the scheduling of multiple user spaces per adapter. This means that if a node is reserved for running a large job, and a small job with high priority is received (one that can be completed before the large job is started), the backfill scheduler algorithm permits LoadLeveler to run this small job on the node.

The backfill scheduler is defined in the configuration file by this line:

```
SCHEDULER_TYPE = BACKFILL
```

Setting SCHEDULER_TYPE overrides any entry you may have in the SCHEDULER_API keyword.

Job control API can be specified if you want to use an external scheduler. This API is intended for those users who want to create a scheduling algorithm for parallel jobs based on specific on-site requirements. It provides a time-based interface, instead of an event-based interface. Further information on the Job Control API is in *IBM LoadLeveler for AIX Using and Administering Version 2 Release 1*, SA22-7311.

To enable the job control API, include this line in the configuration file:

```
SCHEDULER_API = YES
```

Do not include a SCHEDULER_TYPE entry.

17.6 SMP Features

At Version 2 Release 1, LoadLeveler has added the support of multiple tasks being dispatched to nodes and the creation of multiple user spaces per adapter. This feature can improve performance when combined with the backfill scheduling algorithm, as described in section 17.5, “Scheduling” on page 511.

Unfortunately, this does not mean that LoadLeveler is able to schedule jobs on individual CPUs within SMP nodes. LoadLeveler treats all nodes, regardless of the number of CPUs within the node, as one machine and dispatches jobs to the machine to be run. You can, however, specify a job to be run on a particular node. If you have a job that needs to run on an SMP node, you can define that particular job in a class called SMP. Then, you specify in the nodes’ configuration files which node can run the SMP class job.

For example, you can add the following stanza in the administration file:

```
smp: type = class
class_comment = "This is a class that defines a job to require a SMP node"
priority = 90
```

And then add in the local configuration file on the SMP node:

```
Class = SMP
```

You can also add in other jobs that you want the node to handle. Recall that the parameters specified in the local configuration file overwrites the settings in the global configuration file. Therefore, any job classes that you defined as being allowed to run on the SMP node will not run unless you specify them again in the local configuration file.

17.7 DCE Security Integration

For those running the Distributed Computing Environment (DCE) and LoadLeveler, there is one additional factor of which you need to be aware.

When users log in using DCE, they are issued a DCE ticket granting ticket. If they then submit a job using LoadLeveler, LoadLeveler keeps a copy of the DCE ticket granting ticket. If this job has to wait in the queue, there is a chance that the DCE ticket granting ticket may expire when the job is finally dispatched by LoadLeveler. If this is the case, the job will not run because there is insufficient authentication.

We therefore recommend that you exercise caution and set a reasonable expiration date for DCE ticket-granting tickets.

Appendix A. Currently Supported Adapters

These lists are for reference only. A complete list and further documentation can be found at the official Web site for RS/6000 products at the following URL:

`http://www.rs6000.ibm.com`

Table 21. Supported Network Interface Cards for MCA Nodes

Feature Code	Card Type	Description
2972	[8-S]	Auto Token-Ring LAN Streamer Adapter
2980	[2-1]	Ethernet Adapter
2992	[8-U]	Ethernet 10 Mbps AUI/RJ-45 Adapter
2993	[8-V]	Ethernet 10 Mbps BNC Adapter
2994	[9-K]	Ethernet 10/100 Mbps Adapter
2724	[2-R]	FDDI-Fiber Single Ring Adapter
2723	[2-S]	FDDI-Fiber Dual Ring Upgrade Adapter
2989	[9-9]	TURBOWAYS 155 Mbps ATM Adapter
2960	[2-4]	X.25 Interface Co-processor/2
2700	[2-3]	4-Port Communications Controller (SDLC)

Table 22. Supported SCSI and SSA Adapters for MCA Nodes

Feature Code	Card Type	Description
2412	[4-C]	Enhanced SCSI-2 Differential Fast/Wide Adapter
2415	[4-7]	SCSI-2 Fast/Wide Adapter
6216	[4-G]	Enhanced SSA 4-Port Adapter
6219	[4-M]	SSA Multi Initiator/RAID EL Adapter

Table 23. Other Supported Adapters for MCA Nodes

Feature Code	Card Type	Description
2754	[5-3]	S/390 ESCON Channel Emulator

© Copyright IBM Corp. 1999

Feature Code	Card Type	Description
2755	[5-2]	Block Multiplexer Channel Adapter
2756	[5-3]	ESCON Control Unit Adapter
2930	[3-1]	8-Port Asynchronous Adapter (EIA-232)
8128	[3-7]	128-Port Asynchronous Controller
7006		RICP 1 MB Portmaster Adapter
6305	[6-6]	Digital Trunk Dual Adapter

Table 24. Supported Network Interface Cards for PCI Nodes

Feature Code	Card Type	Description
2920	[9-O]	Auto LAN Streamer Token Ring Adapter
2985	[8-Y]	Ethernet 10 Mbps BNC/RJ-45 Adapter
2987	[8-Z]	Ethernet 10 Mbps AUI/RJ-45 Adapter
2968	[9-P]	10/100 Mbps Ethernet Adapter
2969	[9-U]	Gigabit Ethernet adapter
2741		SysKonnect SK-NET FDDI-LP SAS
2742		SysKonnect SK-NET FDDI-LP DAS
2743		SysKonnect SK-NET FDDI-UP SAS
2963	[9-J]	Turboways 155 Mbps UTP ATM Adapter
2988	[9-F]	Turboways 155 Mbps MMF ATM Adapter
2962	[8-L]	2-Port Multiprotocol Adapter

Table 25. Supported SCSI and SSA Adapters for PCI Nodes

Feature Code	Card Type	Description
6206	[4-K]	UltraSCSI Single-Ended Adapter
6207	[4-L]	UltraSCSI Differential Adapter
6208	[4-E]	SCSI-2 Fast/Wide Adapter
6209	[4-F]	SCSI-2 Differential Fast/Wide Adapter

Ex.1010 / Page 542 of 572

Feature Code	Card Type	Description
6215	[4-N]	SSA Multi-Initiator/RAID EL Adapter

Table 26. Other Supported Adapters for PCI Nodes

Feature Code	Card Type	Description
2751	[5-5]	S/390 ESCON Channel Adapter
2943	[3-B]	8-Port Asynchronous Adapter (EIA-232/RS 422)
2934	[3-C]	128-Port Asynchronous Controller
2947	[9-R]	ARTIC960Hx 4-Port Selectable Adapter
6310	[4-E]	ARTIC960RxD Quad Digital Trunk adapter (T1/E1)

Appendix B. Special Notices

This publication is intended to help IBM Customers, Business Partners, IBM System Engineers, and other RS/6000 SP specialists who are involved in Parallel System Support Programs (PSSP) Version 3, Release 1 projects, including the education of RS/60000 SP professionals responsible for installing, configuring, and administering PSSP Version 3, Release 1. The information in this publication is not intended as the specification of any programming interfaces that are provided by Parallel System Support Programs. See the PUBLICATIONS section of the IBM Programming Announcement for PSSP Version 3, Release 1 for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter 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 the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594 USA.

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, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

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

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM ("vendor") products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this

© Copyright IBM Corp. 1999

information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

Any performance data contained in this document was determined in a controlled environment, and therefore, the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

The following document contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples contain 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.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

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

IBM ®	AIX
BookManager	Global Network
ESCON	HACMP/6000
LoadLeveler	OS/390
POWERparallel	RS/6000
S/390	SP

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc. in the United States and/or other countries.

Java and all Java-based trademarks and logos are trademarks or registered

trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

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

PC Direct is a trademark of Ziff Communications Company in the United States and/or other countries and is used by IBM Corporation under license.

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of in the United States and/or other countries. (For a complete list of Intel trademarks see www.intel.com/dradmarx.htm)

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

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

Appendix C. Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

C.1 International Technical Support Organization Publications

For information on ordering these ITSO publications see “How to Get ITSO Redbooks” on page 529.

- *PSSP 3.1 Announcement,* SG24-5332
- *RS/6000 SP PSSP 2.4 Technical Presentation*, SG24-5173
- *RS/6000 SP PSSP 2.3 Technical Presentation*, SG24-2080
- *RS/6000 SP PSSP 2.2 Technical Presentation*, SG24-4868
- *Inside the RS/6000 SP*, SG24-5145
- *SP Perspectives: A New View of Your SP System,* SG24-5180
- *RS/6000 SP Monitoring: Keeping it Alive*, SG24-4873
- *RS/6000 SP: PSSP 2.2 Survival Guide*, SG24-4928
- *Understanding and Using the SP Switch*, SG24-5161
- *RS/6000 SP Performance Tuning*, SG24-5340 (Available in June 1999)
- *RS/6000 Performance Tools in Focus*, SG24-4989
- *Elements of Security: AIX 4.1*, GG24-4433
- *RS/6000 SP High Availability Infrastructure*, SG24-4838
- *GPFS: A Parallel File System*, SG24-5165
- *IBM 9077 SP Switch Router: Get Connected to the SP Switch*, SG24-5157
- *RS/6000 SP: Problem Determination Guide*, SG24-4778
- *SP System Management: Easy, Lean and Mean*, GG24-2563
- *AIX Storage Management*, GG24-4484

C.2 Redbooks on CD-ROMs

Redbooks are also available on the following CD-ROMs:

CD-ROM Title	Collection Kit Number
System/390 Redbooks Collection	SK2T-2177

© Copyright IBM Corp. 1999

CD-ROM Title	Collection Kit Number
Networking and Systems Management Redbooks Collection	SK2T-6022
Transaction Processing and Data Management Redbook	SK2T-8038
Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
AS/400 Redbooks Collection	SK2T-2849
RS/6000 Redbooks Collection (HTML, BkMgr)	SK2T-8040
RS/6000 Redbooks Collection (PostScript)	SK2T-8041
RS/6000 Redbooks Collection (PDF Format)	SK2T-8043
Application Development Redbooks Collection	SK2T-8037

C.3 Other Publications

These publications are also relevant as further information sources:

- *IBM Parallel System Support Programs for AIX: Installation and Migration Guide*, GA22-7347
- *IBM Parallel System Support Programs for AIX: Administration Guide*, SA22-7348
- *IBM Parallel System Support Programs for AIX: Diagnosis Guide*, GA22-7350
- *IBM Parallel System Support Programs for AIX: Messages Reference*, GA22-7352
- *IBM Parallel System Support Programs for AIX: Command and Technical Reference, Volume 1* and Volume 2, SA22-7351
- *IBM Parallel System Support Programs for AIX: Managing Shared Disks*, SA22-7349
- *IBM RS/6000 SP Planning Volume 1, Hardware and Physical Environment*, GA22-7280
- *IBM RS/6000 SP Planning Volume 2, Control Workstation and Software Environment*, GA22-7281
- *RS/6000 Cluster Technology: Event Management Programming Guide and Reference*, SA22-7354
- *RS/6000 Cluster Technology: Group Services Programming Guide and Reference*, SA22-7355
- *IBM Parallel System Support Programs for AIX Performance Monitoring Guide and Reference*, SA22-7353

Ex.1010 / Page 550 of 572

- *IBM General Parallel File System for AIX: Installation and Administration Guide*, SA22-7278
- *HACMP for AIX: Enhanced Scalability Installation and Administration Guide*, SC23-4284
- *Parallel Environment for AIX: Operation and Use, Volume 1*, SC28-1979
- *Parallel Environment for AIX: Installation Version 2 Release 4*, GC28-1981
- *Parallel Environment for AIX: Operation and Use, Volume 2, Part 1 and 2*, SC28-1980
- *Parallel Environment for AIX: Hitchhiker's Guide*, GC23-3895
- *IBM XL High Performance Fortran Language Reference and User's Guide*, SC09-2631
- *IBM XL Fortran Language Reference*, SC09-2718
- *IBM XL Fortran User's Guide*, SC09-2719
- *LoadLeveler for AIX: Using and Administering Version 2 Release 1*, SA22-7311
- *AIX Version 4.3 System User's Guide: Communications and Networks*, SC23-4127
- *Distributed Computing Environment for AIX, Version 2.2: Quick Beginnings*, SC23-4188
- *AIX Version 4.3 Commands Reference*, SC23-4119
- *AIX Version 3.2 and 4 Performance Monitoring and Tuning Guide*, SC23-2365
- *AIX Version 4.3 System Management Guide: Operating System and Devices*, SC23-4126
- *AIX Version 4.3 Network Installation Management Guide and Reference*, SC23-2627

C.4 External Publications

These are non-IBM publications relevant as further information sources:

- *Managing NIS and NFS*, O'Reilly and Associates
- *The Kerberos Network Authentication Service (V5)*, RFC1510
- *Telnet Authentication Option*, RFC1416
- *Generic Security Service API*, RFC1508

How to Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** `http://www.redbooks.ibm.com/`

  Search for, view, download or order hardcopy/CD-ROM redbooks from the redbooks web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this redbooks site.

  Redpieces are redbooks in progress; not all redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

- **E-mail Orders**

  Send orders via e-mail including information from the redbooks fax order form to:

	e-mail address
In United States	usib6fpl@ibmmail.com
Outside North America	Contact information is in the “How to Order” section at this site: `http://www.elink.ibmlink.ibm.com/pbl/pbl/`

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the “How to Order” section at this site: `http://www.elink.ibmlink.ibm.com/pbl/pbl/`

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	Fax phone number is in the “How to Order” section at this site: `http://www.elink.ibmlink.ibm.com/pbl/pbl/`

This information was current at the time of publication, but is continually subject to change. The latest information for customer may be found at `http://www.redbooks.ibm.com/` and for IBM employees at `http://w3.itso.ibm.com/`.

IBM Intranet for Employees

IBM employees may register for information on workshops, residencies, and redbooks by accessing the IBM Intranet Web site at `http://w3.itso.ibm.com/` and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may also view redbook. residency, and workshop announcements at `http://inews.ibm.com/`.

© Copyright IBM Corp. 1999

IBM Redbook Fax Order Form

Please send me the following:

Title	Order Number	Quantity

First name	Last name	
Company		
Address		
City	Postal code	Country
Telephone number	Telefax number	VAT number

☐ Invoice to customer number _____

☐ Credit card number _____

Credit card expiration date	Card issued to	Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

List of Abbreviations

AIX	Advanced Interactive Executive
AMG	Adapter Membership Group
ANS	Abstract Notation Syntax
API	Application Programming Interface
BIS	boot/install server
BSD	Berkeley Software Distribution
BUMP	Bring-Up Microprocessor
CP	Crown Prince
CPU	central processing unit
CSS	communication subsystem
CWS	control workstation
DB	database
EM	Event Management
EMAPI	Event Management Application Programming Interface
EMCDB	Event Management Configuration Database
EMD	Event Manager Daemon
EPROM	Erasable Programmable Read-Only Memory
FIFO	first-in first-out
FS	file system
GB	gigabytes
GL	Group Leader
GPFS	General Parallel File System
GS	Group Services
GSAPI	Group Services Application Programming Interface
GVG	Global Volume Group
HACMP	High Availability Cluster Multiprocessing
HACMP/ES	High Availability Cluster Multiprocessing Enhanced Scalability
hb	heart beat
HiPS	High Performance Switch
hrd	host respond daemon
HSD	Hashed Shared Disk
IBM	International Business Machines Corporation
IP	Internet Protocol
ISB	Intermediate Switch Board
ISC	Intermediate Switch Chip
ITSO	International Technical Support Organization
JFS	Journaled File System
LAN	Local Area Network
LCD	liquid crystal display
LED	light emitter diode
LP	logical partition
LRU	last recently used
LSC	Link Switch Chip
LV	logical volume
LVM	Logical Volume Manager
MB	megabytes

© Copyright IBM Corp. 1999

MIB	Management Information Base
MPI	Message Passing Interface
MPL	Message Passing Library
MPP	Massive Parallel Processors
NFS	Network File System
NIM	Network Installation Management
NSB	Node Switch Board
NSC	Node Switch Chip
OID	object ID
ODM	Object Data Manager
PAIDE	Performance Aide for AIX
PE	Parallel Environment
PID	process ID
POE	Parallel Operating Environment
PP	physical partition
PSSP	Parallel System Support Programs
PTC	prepare to commit
PTPE	Performance Toolbox Parallel Extensions
PTX	Performance Toolbox for AIX
PV	physical volume
RAM	random access memory
RCP	Remote Copy Protocol
RM	Resource Monitor
RMAPI	Resource Monitor Application Programming Interface
RPQ	Request For Product Quotation
RSCT	RS/6000 Cluster Technology
RSI	Remote Statistics Interface
R/VSD	Recoverable/Virtual Shared Disk
RVSD	Recoverable Virtual Shared Disk
SBS	structured byte string
SCSI	Small Computer System Interface
SDR	System Data Repository
SMIT	System Management Interface Tool
SSA	Serial Storage Architecture
VG	volume group
VSD	Virtual Shared Disk

Ex.1010 / Page 556 of 572

Index

Symbols

/etc/ntp.conf 134
/etc/rc.ntp 134
/etc/security/user 147
/etc/sysctl.pman.acl 297
/usr/lpp/ssp/install/bin/spfbcheck 289

Numerics

100BASE-TX 48, 56, 58
10BASE-2 47
10BASE-T 48
8274 57

A

abbreviations 531
Accelerated Strategic Computing Initiative 3
Access Control Lists 164, 171, 180, 182
ACL callback 181
acronyms 531
Administrative Ethernet 47
ADSM 7
Adstar Distributed Storage Manager 7
Agora 5
AIX Error Notification Facility 331
ALIGN 477
alternate rootvg 291
AMD 365
amd_config 369
Andrew File System 461
ARP cache 56
ASCI 3, 6, 481
asymmetric encryption 146, 150
asynchronous I/O 478
Athena project 141, 148
auditing 139
AUTH callback 181
auth_install 143
auth_methods 143, 166
auth_root_rcmd 143, 165
auth1 147
Authentication 140
  methods 143
  Service 149
  System 149
authenticator 153
Authorization 140, 172
authorization callbacks 181
AutoFS 365, 366
automounter 364, 365, 366
autosensing 58

B

Babel Fish 476
backfill scheduler 511
backup 375
barrier synchronization 464
bcastping 445
BNC 47
boot device 280, 376
boot image 376
boot sector 376
boot/install server 42, 51, 52, 258, 279, 285, 289
bootable image 376
bootlist 292
bootp 273, 288
broadcast storm 55
broadcasts 464
BUMP processor 288

C

C2 level of trust 141
cascading 434
CDS server 168
Central Manager 497
Cerberus 148
chauthpar 170
choice arguments 465
client/server model 140
clinfo 446
clstrmgr 445
cluster 463
Cluster Single-Point-of-Control 434
Collective Communications 464
Collective Operations 465
Commands
  3dmon 319
  add_principal 162
  backup 381
  bffcreate 265
  cfgvsd 400
  chauthent 143

© Copyright IBM Corp. 1999

chauthpar 143, 166
chgcss 308
chuser 340
cl_chgroup 445
cl_chuser 445
cl_clstop 451
cl_lsgroup 445
cl_lsuser 445
cl_mkgroup 445
cl_mkuser 445
cl_rc.cluster 450
cl_rmgroup 445
cl_rmuser 445
cl_setup_kerberos 449
clchclstr 449
cldare 434
clstat 451
clstop 451
cpio 381
create_krb_files 167
createvsd 398
css.snap 241, 245, 336
CSS_test 290
cw_restrict_login 357
dbx 481
dce_login 169
dcecp 169
dceunixd 169
defvsd 399
dsh 180
Eannotator 222, 286
Eclock 238, 239, 287
Efence 231, 237
Eprimary 286
Equiesce 238
Estart 217, 225, 238
fencevsd 407
ftp 141
hmadm 116, 173
hmcmds 111, 119, 171
hmmon 111, 116, 171
install_cw 267
k4destroy 159
k4init 159, 171
k4list 159, 171
k5dcelogin 169, 178
kadmin 162
kinit 150
kpasswd 159
ksrvtgt 162
ksrvutil 161
kstash 163
llctl 502
llextSDR 508
llq 507, 511
llstatus 502
llsubmit 511
llsummary 325
lsauthent 143, 177
lsauthpar 143
lsfencevsd 407
lskp 158
lsuser 340
lvlstmajor 444
mkautomap 372
mknimres 285
mksysb 375
mkuser 340
mmconfig 418
mmmkvsd 183
mmremote 183
mpxlf 484, 491
nodecond 111, 119, 171, 287
pdbx 480, 482, 484
pedb 482, 483, 484
perspectives 119
pghpf 480
pgprof 480
poe 481, 484, 491
poeauth 462
post_process 267
pssp_script 260, 289
psspfb_script 289
ptpectrl 318
ptpehier 316, 318
rcmdtgt 179
rcp 173
rdump 381
rexec 142
rmuser 340
rsh 143, 173, 175
rvsdrestrict 406
s1term 111, 114, 119, 171, 288
savevg 376
scp 146
script.cust 289
SDR_test 105, 267
SDRArchive 103, 107, 379

Ex.1010 / Page 558 of 572

SDRChangeAttrValues 104, 106
SDRClearLock 102, 106
SDRGetObjects 105
SDRListClasses 106
SDRRestore 103, 107, 380
SDRWhoHasLock 102, 106
setup_authent 164, 168, 267
setup_server 167, 284
setup_ssp_server 164
slogin 146
smit config_mgmt 103
snap 334
spacctnd 321
spacs_cntrl 346, 347
spbootins 291
spbootlist 291
spchuser 341, 344
spevent 297
sphardware 119, 171
splstdata 103, 282
splsuser 341, 345
spmkuser 341, 343
spmon 111, 117, 171
spmon_itest 267
sprmuser 341, 344
spsetauth 165, 166
spsitenv 133, 321, 341
spvsd 304
ssh 146
startvsd 401
supper 356, 360
sysctl 180
sysctl svcrestart 304
SYSMAN_test 286
syspar_ctrl 279
tar 381
telnet 141
totalview 484, 487
tprof 489
unfencevsd 407
vampir 491
vsd.snap 337
vsdnode 397
vt 488
xauth 144, 145
xhost 144
xlhpf 479
xloadl 504
xmperf 319
xntpdc 134
xpdbx 481
xprofiler 489
Communicators 464
concurrent 433, 434
concurrent access 444
Configuration Manager 412
consensus 133
console 114, 119
control 109
control workstation 42
cookie 144, 145
CSMA/CD 47
customization 283, 289
customize 258
CustomMetric 512
CWS
See control workstation

D

Daemons
automountd 366
clinfo 446
css.summlog 240, 241
cssadm 227, 229, 235, 237
godm 444
haem 279
hags 279
hardmon 110, 267, 270
hats 279
hc 406
hmrmd 173
hr 279
inetd 457
kadmind 162, 164
kbdd 499
kerberos 162, 164
krshd 176, 178
master 498
negotiator 498
nfd 113
pmand 330
pmanrmd 330
pmdv2 456
pvmd 487
rshd 176, 177
rvsd 406
s70d 113

schedd 498
sdrd 91, 98, 99
setup_logd 267
sp_configd 330
spdmapid 315
spdmcold 315
spdmspld 315
splogd 111, 173
startd 498
starter 498
supman 356
sysctld 180
Worm 218, 235
xmservd 312
ypbind 349
yppasswd 349
ypserv 349
ypupdated 349
DARE
See Dynamic Automatic Reconfiguration Event
Data distribution directives 477
Data Encryption Standard 149
dataless 258
data-parallel programming 476
DCE ticket granting ticket 515
dce_export 147
dce_login -f 177
dceunix -t 177
Department of Energy 481
design 257
device database 236
diagnostics mode 118
discretionary security control 141
diskless 258
DISTRIBUTE 477
dog coffins 5
Dolphin Interconnect Solutions 484
Dynamic Automatic Reconfiguration Event 434
dynamic port allocation 394
dynamic reconfiguration 434

E

eavesdropping 148
EMAPI 203
endpoint map 394
Environment Variables
KRBTKFILE 160, 162
MP_AUTH 459, 462
MP_BUFFER_MEM 459
MP_DEBUG_INITIAL_STOP 484
MP_MSG_API 476
MP_NEWJOB 462
MP_PULSE 487
MP_TRACELEVEL 488
MP_USE_LL 459
SP_NAME 98, 100
Envoy 5
Ethernet switch 48, 54
Etnus, Inc. 481, 484
Event Management 200
aixos 208
client 202
daemon 187
EMAPI 187, 204
EMCDB 204, 205
event registration 203
expression 203
ha_em_peers 204
haemd 201, 204
rearm expression 203
resource class 203, 206
resource variable 203
RMAPI 202, 204
Event Perspective 296
/etc/sysctl.pman.acl 297
Condition Pane 302
Conditions Pane 301
Create Condition notebook window 301
Event Condition 301, 302
Event Definition 301, 303
event definition 296
Event Definitions 298
Event Defintion 302
event icon 300
Event Management 296
Event Notification Log 303
event notification log 300
icon colors for event definitions 299
pre-defined events 298
Rearm Expression 302
rearm expression 300
registering events 296, 299
resource elements ID 303
resource variable 302
resource variable class 302
spevent 297
unregistering events 296

Ex.1010 / Page 560 of 572

Executing Machine 497
EXTRINSIC 477

F

Fast Ethernet 56, 57
fault-tolerant 433
File Collections 339, 355
Files
$HOME/.k5login 178
$HOME/.netrc 141, 142
$HOME/.rhosts 142, 177
$HOME/.Xauthority 145
/.k 163, 164
/.k5login 169
/.klogin 165
/etc/auto.master 366
/etc/auto/maps/auto.u 369, 370
/etc/auto_master 366
/etc/bootptab 260
/etc/bootptab.info 274
/etc/firstboot 289
/etc/group 147, 339, 340
/etc/hosts 101
/etc/hosts.equiv 142, 177
/etc/inetd.conf 177
/etc/inittab 100, 164, 235, 241, 289
/etc/krb.conf 157, 164, 167
/etc/krb.realms 157, 164, 167
/etc/krb5.conf 169
/etc/krb-srvtab 161, 162, 165, 167, 171, 172, 179, 181
/etc/ntp.config 134
/etc/passwd 147, 339, 340
/etc/poe.limits 459
/etc/rc.net 263
/etc/rc.ntp 134
/etc/rc.sp 98, 100
/etc/SDR_dest_info 98, 99, 100, 101, 289
/etc/security/ 147
/etc/security/group 340
/etc/security/login.cfg 147
/etc/security/passwd 339, 340
/etc/security/user 147, 346
/etc/services 98, 158
/etc/SP/expected.top 286
/etc/sysctl.acl 182, 304
/etc/sysctl.conf 180
/etc/sysctl.mmcmd.acl 183
/etc/sysctl.pman.acl 183, 297
/etc/sysctl.vsd.acl 304
/etc/sysctl/logmgt.acl 183
/spdata/sys1/install/images 259, 284
/spdata/sys1/sdr/defs/Frame 113
/spdata/sys1/sdr/system/classes/Frame 113
/spdata/sys1/spmon/hmacls 116, 172
/spdata/sys1/spmon/hmthresholds 111
/spdata/sys1/spmon/hwevents 111
/tftpboot/firstboot.cust 284
/tftpboot/-new-srvtab 167
/tftpboot/script.cust 284, 289
/tftpboot/tuning.cust 267, 284, 289
/tmp/tkt 160
/tmp/tkt_hmrmd 173
/tmp/tkt_splogd 173
/usr/lib/security/DCE 147
/usr/lpp/mmfs/bin/mmcmdsysctl 183
/usr/lpp/ssp/bin/nodecond_chrp 287
/usr/lpp/ssp/bin/nodecond_mca 287
/usr/lpp/ssp/install/bin/psspfb_script 289
/usr/lpp/ssp/install/config/tuning.default 267
/usr/lpp/ssp/samples/block_usr_sample 347
/usr/sbin/cluster/etc/clhosts 446
/usr/sbin/cluster/etc/clinfo.rc 446
/usr/sbin/cluster/sbin/cl_krb_service 449
/var/adm/SPlogs/css/out.top 248
/var/adm/SPlogs/css/summlog 240
admin.add 164
admin.get 164
admin.mod 164
cssadm.cfg 227
cssadm.debug 230
LoadL_admin 502
LoadL_config 502
logevnt.out 245
logmgt.cmds 183
passwd_overwrite 148
perfagent.tools 262
pman.cmds 183
principal.dir 163
principal.pag 163
firstboot.cust 167
FORALL 477
Fortran 95 479
frame 20, 111
model frame 21
short expansion frame 21
short model frame 21

SP Switch frame 22
supervisor interface 114
tall expansion frame 21
tall model frame 21
FZJ 491

G

gather/scatter operations 464
General Parallel File System (GPFS) 409
get_auth_method 143, 175, 177, 178
gettokens.c 461
Global file systems 387
global network 447
Global ODM 188
GODM
See Global ODM
GRF 39
Group Services 187
barrier synchronization 199
clients 193
external or user groups 194
group 193
Group State Value 194
Membership List 194
Name 194
Group Leader 197, 198, 199
Group Services Application Programming Interface 195
hagsd 193, 196
hagsglsmd 196
internal components 195
internal groups 194
join request 196
meta-group 196
nameserver 195
namespace 195
Protocols 198
providers 193
Source-target facility 200
subscriber 193
sundered namespace 200
Voting 199
1-phase 199
n phase 199
GSAPI 195

H

HACMP 185
HAI, see also High Availability Infrastructure 185
Half duplex 48
hardmon 110, 172
hardmon principal 171
hardware address 273
Hardware Perspective 120
Controlling hardware 295
icon view 296
Monitoring hardware 295
panes 296
sphardware command 295
system objects 295
hardware_protocol 112
Hashed Shared Disks 404
hatsd 189
HDX
See Half Duplex
heartbeat 447
high availability 433
High Availability Cluster Multiprocessing 185
High Availability Cluster Multiprocessing Enhanced Scalability (HACMP/ES) 433
High Availability Cluster Multiprocessing Enhanced Scalability Concurrent Resource Manager (HACMP/ESCRM) 433
High Availability Control Workstation 45
High Availability Infrastructure 185
High Performance Fortran 476
High Performance Gateway Node 39
High Performance Supercomputer Systems Development Laboratory 4
home directories 387
home node 453
host impersonation 142
host responds 290
host responds daemon 207, 209
HPSSDL 4
HPSSL 5
hrd
See host responds daemon

I

I/O pacing 445
IBM Support Tools 333
css.snap 240, 336
Gathering AIX Service Information 334
Gathering SP-specific Service Information 336
Service Director 333

snap 334
vsd.snap 337
Identification 140
IEEE POSIX 1003.1-1996 471
impersonation 140, 142
implementation 257
INDEPENDENT 477
Initial Program Load 260
in-place buffers 465
insecure networks 148
Install Ethernet, 51
installation 257
integrated DCE login 147
Intermediate Switch Board 22
IPL
See Initial Program Load
ipsrcrouteforward 445
ipsrcrouterecv 445
ipsrcroutesend 445
ISB
See Intermediate Switch Board

J

job command file 502
job control API 512
Job Switch Resource Table (JRST) 511

K

K5MUTE 176
kcmd 176, 178
Kebreros master key 163
Kerberos 141, 142, 329, 380
kerberos port 176
Kerberos principal 304
kerberos4 port 176
Key Distribution Center 169
key mode 119
kshell port 176, 178
kvalid_user 178

L

LAPI_Amsend 475
LAPI_Get 475
LAPI_Put 475
Launch Pad 119, 293, 294
libc.a 176
libspk4rcmd.a 176
libvaliduser.a 178
loadable authentication module 147
LoadAvg 512
LoadL_starter 457
LoadLeveler 497
Accounting 323
Job Resource Data 323
Job Resource Information - User Accounts 325
llsummary 325
parallel jobs 323
serial jobs 323
administration manual 323
cluster 497
scheduler 511
SYSPRIO 501
user priority 501
Low-Level API 475
lppsource 259, 285

M

MACHPRIO 512
Massachusetts Institute of Technology 141, 148
MasterMachPriority 512
Message Passing Interface 463
Message Passing Interface Forum 463
Message Passing Library 476
mirroring 290, 438
MIT-MAGIC-COOKIE-1 145
mksysb 266, 382
monitor 109
Monitor and Control Node (MACN) 110
Monitoring 126
MPI datatypes 464
MPI I/O 465
MPI Version 1.1 464
MPI Version 1.2 465
MPI Version 2 465
multi-homed hosts 157
mutual authentication 155

N

n2 problem 185
National Security Agency 141
Negotiator 497
netboot 287
Netfinity 112, 115, 269
network boot 284, 287

Network Connectivity Table 189
Network File System 259
Network Information System 339, 348
client 349
login control 147
maps 349
Master Server 348
Slave Server 348
Network installation 55
Network Installation Management 258, 284
Network Module 448
Network Option 263
Network Time Protocol 131
ntp_config 133
ntp_server 133
ntp_version 133
peer 132
stratum 132
timemaster 133
nfd 115
NFS
See Network File System
NIM
See Network Installation Management
NIM pull mode 260
NIM push mode 260
NIS
See Network Information System
node
dependent node 38
external node 34
High node 26
Internal Nodes 26
standard node 26
Thin node 26
Wide node 26
node database 272
node supervisor interface 114
non-concurrent 438
none 133
NONE callback 181
nonlocsrcroute 445
NTP
See Network Time Protocol
ntp_config 133, 134
ntp_server 133
Nways LAN RouteSwitch 57

O

Oak Ridge National Laboratory 475
One-Sided Communication 465
OpenMP 479
ORNL 475

P

PAIDE
See Performance Aide for AIX
Pallas GmbH 479, 484, 488, 491
panes 120
Parallel Operating Environment 215
Parallel Tools Consortium 480
Parallel Virtual Machine 215, 475
partition manager daemon 455
Partitioning the SP System
See System Partitioning
passwd_overwrite 148
PDT
See Performance Management Tools
peer 132
perfagent.server 186, 265
perfagent.tools 186, 265
performance 263
Performance Aide for AIX 265
Performance Management 311
AIX utilities 311
PerfPMR 312
Performance Monitoring 305
Performance Toolbox (PTX/6000) 312
Performance Toolbox Parallel Extensions 313
System Performance Measurement Interface 312
Performance Optimization with Enhanced 4
Performance Toolbox (PTX/6000) 265, 312
3dmon 312
Performance Agent 312
Performance Manager 312
Remote Statistics Interface (RSI) 313
xmservd daemon 312
Performance Toolbox Parallel Extensions 206, 313
3dmon 319
Central Coordinator nodes 315
Collection of SP-specific data 314
Collector nodes 315
Data Analysis and Data Relationship Analysis 315
Data Manager nodes 315

Ex.1010 / Page 564 of 572

Installation and Customization 316
monitoring hierarchy 316
ptpe.docs 316
ptpe.program 316
ptpectrl 318
ptpehier 316, 318
SP runtime monitoring 314
spdmapid 315
spdmcold 315
spdmspld 315
ssp.ptpegui 316
xmperf 319
personal password 140
Phoenix 185
PING_CLIENT_LIST 446
plain text passwords 142
planning 257
PMAN Subsystem 329
default events 332
Event Management client 331
pmand 330
pmandefaults script 332
pmanrmd 330
sp_configd 330
ssp.pman 329
subscribed events 332
POE
See Parallel Operating Environment
Point-to-Point Communication 464
poll rate 111
poor passwords 140
Portland Group 479
POWER 4
Power Supplies 23
POWER3 32
PowerPC 30
principal 149
private key 146, 150
Private Network 441
Problem Management 208, 326
AIX Error Logging 327
BSD syslog 327
pmanrmd 208
SP Error Logs 328
Process Creation 465
Process Topologies 464
PROCESSORS 477
PROCLAIM messages 56
Profiling 464
pssp_script 167
PTPE
See Performance Toolbox Parallel Extensions
PTX
See Performance Toolbox (PTX/6000)
PTX/6000
See Performance Toolbox (PTX/6000)
public key 146, 150
Public Network 442
PVM
See Parallel Virtual Machine
pvm_spawn/pvmfspawn 475

Q

quorum 279

R

RAS 8
rcmd 176, 179
rcmd principal 179, 180
r-commands 173
Read-Only Storage 260
realm 157
receive 464
Recoverable Virtual Shared Disk 405
reduction operations 464
registry 147
Reliable Messaging 189
remote execution commands 173
remote node 454
replay attacks 140, 148, 153
resource group 434, 440
Resource Identifier 203
Resource Manager 511
Resource Monitor Application Programming Interface 202
Resource Monitors 201, 205
aixos 208
external 205
aixos 208
CSSLogMon 208
harmld 208
harmpd 207
hmrmd 207
pmanrmd 208
SDR 208
internal 205
Membership 208

Response 208
resource variables 206
observation interval 206
reporting interval 206
Resource Variables
IBM.PSSP.Membership.LANadapter.state 213
IBM.PSSP.Response.Host.state 209
IBM.PSSP.Response.Switch.state 209
restore 375
RFC 1416 143
RFC 1508 143
RFC 1510 142, 148
RJ-45 48
RMAPI
See Resource Monitor Application Programming Interface
root.admin 172
ROS
See Read-Only Storage
rotating 434
routing 50, 53
RPM 480
RS232 443
RVSD Failover 407

S

S70 112, 269
s70d 113
S7A 112, 269
SAMI 112, 269
SCHEDULER_API 512
SCHEDULER_TYPE 512
Scheduling Machine 497
SCSI 280
SDR 190, 192, 204
See also System Data Repository
SDRCreateFile 104
SDRCreateSystemFile 104
SDRDeleteFile 104
SDRReplaceFile 104
SDRRetreiveFile 104
secondary Kerberos servers 157
secret key 146, 149
secret password 143
secure shell 145
Security
ftp 141, 143
rcp 142, 143
rexec 142
rlogin 142
rsh 142, 143, 175
telnet 141, 143
xauth 145
xhost 144
security administration 139
security policy 139
Security Server 168
send 464
Serial Network 443
Service Processor 112
service ticket 149, 160
Services
kerberos 158
kerberos_admin 158
kerberos4 158
kerberos-adm 158
krb_prop 158
pmv2 456
session key 151
set_auth_method 143
settokens.c 461
setup_authent 164
shell port 177
Simple Network Management Protocol 329
site environment 268
SLIM 112, 269
SMPI
See System Performance Measurement Interface
SNMP
See Simple Network Management Protocol
SP access control (SPAC) 340, 346
SP Accounting 319
accounting class identifiers 321
Configuring 320
Node-exclusive-use accounting 320
Parallel job accounting 320
record consolidation 319
spacctnd 321
ssp.sysman fileset 319
user name space 323
SP LAN 47
SP Perspectives 116, 119, 293
Event Perspective 294
Hardware Perspective 293, 295
Launch Pad 293, 294
Performance Monitor Perspective 294

Ex.1010 / Page 566 of 572

SP Resource Center 294
spmon -g 295
system management tasks 293
System Partitioning Aid 294
Virtual Shared Disk Perspective 294
SP Switch 215
Administration and Management 225
autojoin 237
autojoin attribute 231
Automatic Node Unfence 226, 231
automatic unfence 237
Centralized Error Logging 226
Clock Master 238
css.logevnt method 245
css.snap 240, 241, 245, 247
css.summlog 240, 241
cssadm 231, 235, 237
cssadm daemon 227, 228
Eannotator 222, 223
Eclock 238, 239
Eclock topology file 239
Efence 232
Error Logging 240
error logging daemon 240
error notification objects 242
Error recovery 217
Estart 217, 238
Eunfence 233
Fencing nodes 237
Global Shutdown 238
Intermediate Switch Board 218, 239
ISB 220
link 220
log consolidation 241
Log Files 247
logevnt.out 245
management 215
Node Switch Board 218, 239
operation 215
out.top 248
port number 221
primary backup node 217
primary node 217
rc.switch 235
Resource Variables 241
route table 224
Route Table Generator 216, 218, 223
routing table database 218
secondary node 218
Switch Admin Daemon 226
switch chip number 220
Switch Clocks 238
Switch Management 235
switch port number 221
switch scan 232
Switch Topology File 218
Switch-Dependent Application Startup 226, 235
switch-to-switch connection 221
swtlog 241
TBIC
See Trail Blazer Interface Chip
topology file 222
topology filename format 218
Trail Blazer Interface Chip 233
Unfencing nodes 237
user space protocol 215
Worm 216
SP Switch frame 22
SP Switch Router 39
SP System Tuning 305
Adapter Receive Queue size 309
Adapter Transmit Queue size 309
Administrative Ethernet 306
chgcss 308
default transmit queue 310
External Networks 309
Large Configurations 305
Receive Queues 310
rpool and spool 307
rpoolsize and spoolsize parameters 307
SP Daemons 306
SP Switch 307
Subsystems 306
tcp_sendspace, tcp_recvspace 310
Transmit queues 309
Tunable Network Options 306
udp_sendspace, udp_recvspace 310
SP user management (SPUM) 339, 341
SP_ports 111
SP1 5
SP2 6
SP-Attached Server 35
spdata 264
spk4rsh 176, 178
SPMI
See System Performance Measurement Interface

spmon 119
SPOT 260
spsitenv 133
consensus 133
internet 133
none 133
SSA 280
standalone 258
state 111
stratum 132
Stripe Group Manager 413
Structured Socket Messages 455
Submit-only Machine 498
subnet 50
Subsystems
Emonitor 237
Event Management 296
sdrd 91, 98
swtadmd 227
swtlog 241
supercomputer 4
supervisor card 24
supervisor microcode 271
switch clock 287
switch responds 290
switch topology 286
symmetric encryption 146, 149
Symmetric Multiprocessor 463
syncd 445
synchronization 444, 447
Sysctl 180
syslogd 111
SYSTEM 147
SYSTEM callback 181
system characteristics 6
System Data Repository 91, 104
/etc/inittab 100
/etc/rc.sp 100
/etc/SDR_dest_info 98, 100, 101
/spdata/sys1/sdr/system/locks directory 102
Adapter class example 93
Adding AIX Files to the 104
alias IP address 98
Attributes 92
Backup 107
changing an object 104
Classes 92
default partition 94
Locking 102
Manipulating SDR Data 103
multiple SDR daemons 102
Objects 92
Partitioned classes 94
Restore 107
SDR Daemon 98
SDR Data Model 92
SDR directory structure 93, 95
SDR Log File 106
SDR_test 105
SDRArchive 103, 107
SDRChangeAttrValues 104, 106
SDRClearLock 102, 106
sdrd 91, 98, 99
SDRGetObjects 105
SDRListClasses 106
SDRRestore 103, 107
SDRWhoHasLock 102, 106
Shadow Files 103
SP class example 93
SP_NAME 100
Syspar_map class 100
syspar_name 100
System classes 94
System partitioning and the 91
User Interface 102
System Monitor
command line 304
system hardware 304
System Monitoring 293
System Partitioning 95
/etc/inittab 100
/etc/rc.sp 100
/etc/SDR_dest_info 99, 100, 101
alias IP address 98
default IP address 99
default partition 98
Partitioning Rules 96
primary partition 98
sdrd 99
SP_NAME 100
syspar_addr attribute 99
Syspar_map class 99, 100
syspar_name 100
Why Partition? 97
System Performance Measurement Interface 188, 206
System Resource Controller 237

T

Tcl 180
thewall 263
Thin-wire Ethernet 47
ticket cache 154, 173
ticket cache file 159
ticket forwarding 176
Ticket-Granting Service 149, 150
Ticket-Granting Ticket 149, 151, 160, 284
Tivoli 7
TMSCSI 443
TMSSA 443
TOD 236
Topology 253
Topology Services 187
  Adapter Membership 189, 190
  Adapter Membership Group 189, 190, 191, 192
  Crown Prince 190, 191
  DEATH_IN_FAMILY 191
  Group Leader 190, 191
  hatsd 188
  heartbeat 191, 193
    frequency 191
    sensitivity 191
  machines.lst 190, 191, 192
  Mayor 190
  Neighbor 191
  Network Connectivity Table 189, 192
  Node Membership 189
  Prepare To Commit 192
  Proclaim Packet 191
  Reliable Messaging 189, 201
  Singleton group 189, 191
  topology table 192
TP
  See Twisted Pair
Trailblazer 6
transmit queue size 263
trusted third party system 140
trusted third-party 148
Twisted Pair 48

U

UDP 189
UDP/IP 193
UNIX domain sockets 202
UNIX Domain Stream 189
UNIX mode bits 140
Unshielded Twisted Pair 48
uplink 54
user database 140
UTP
  See Unshielded Twisted Pair

V

Vampir 488, 491
VampirTrace 491
Virtual Front Operator Panel 172
Virtual Front Operator Panel (VFOP) 119
Virtual Shared Disk 395
Virtual Shared Disk Perspective 304
  /etc/sysctl.acl 304
  /etc/sysctl.vsd.acl 304
  shared disk management 304
  shared disk management tasks 304
  spvsd command 304
VSD 96, 250
VSD States 403
Vulcan 5

W

wall_clock_limit 511
Wladawsky-Berger, Irving 5
workstation cluster 463
Worm 218

X

X Windows 144
Xauthority 144
XL Fortran 479
XL High Performance Fortran 478
XL HPF 479
xntpd 134
xntpdc 134

ITSO Redbook Evaluation

The RS/6000 SP Inside Out
SG24-5374-00

Your feedback is very important to help us maintain the quality of ITSO redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at http://www.redbooks.ibm.com
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Which of the following best describes you?
_ Customer _ Business Partner _ Solution Developer _ IBM employee
_ None of the above

Please rate your overall satisfaction with this book using the scale:
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction _____

Please answer the following questions:

Was this redbook published in time for your needs? Yes___ No___

If no, please explain:

What other redbooks would you like to see published?

Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK!)

© Copyright IBM Corp. 1999

SG24-5374-00
Printed in the U.S.A.

The RS/6000 SP Inside Out

SG24-5374-00

